

## Features

- Incorporates the ARM920T™ ARM® Thumb® Processor
  - 200 MIPS at 180 MHz, Memory Management Unit
  - 16-KByte Data Cache, 16-KByte Instruction Cache, Write Buffer
  - In-circuit Emulator including Debug Communication Channel
  - Mid-level Implementation Embedded Trace Macrocell (256-ball BGA Package Only)
- Low Power: 30.4 mA on VDDCORE, 3.1 mA in Standby Mode
- Additional Embedded Memories
  - 16K Bytes of SRAM and 128K Bytes of ROM
- External Bus Interface (EBI)
  - Supports SDRAM, Static Memory, Burst Flash, Glueless Connection to CompactFlash®, SmartMedia™ and NAND Flash
- System Peripherals for Enhanced Performance:
  - Enhanced Clock Generator and Power Management Controller
  - Two On-chip Oscillators with Two PLLs
  - Very Slow Clock Operating Mode and Software Power Optimization Capabilities
  - Four Programmable External Clock Signals
  - System Timer Including Periodic Interrupt, Watchdog and Second Counter
  - Real-time Clock with Alarm Interrupt
  - Debug Unit, Two-wire UART and Support for Debug Communication Channel
  - Advanced Interrupt Controller with 8-level Priority, Individually Maskable Vectored Interrupt Sources, Spurious Interrupt Protected
  - Seven External Interrupt Sources and One Fast Interrupt Source
  - Four 32-bit PIO Controllers with Up to 122 Programmable I/O Lines, Input Change Interrupt and Open-drain Capability on Each Line
  - 20-channel Peripheral Data Controller (DMA)
- Ethernet MAC 10/100 Base-T
  - Media Independent Interface (MII) or Reduced Media Independent Interface (RMII)
  - Integrated 28-byte FIFOs and Dedicated DMA Channels for Receive and Transmit
- USB 2.0 Full Speed (12 Mbits per second) Host Double Port
  - Dual On-chip Transceivers (Single Port Only on 208-lead PQFP Package)
  - Integrated FIFOs and Dedicated DMA Channels
- USB 2.0 Full Speed (12 Mbits per second) Device Port
  - On-chip Transceiver, 2-Kbyte Configurable Integrated FIFOs
- Multimedia Card Interface (MCI)
  - Automatic Protocol Control and Fast Automatic Data Transfers
  - MMC and SD Memory Card-compliant, Supports Up to Two SD Memory Cards
- Three Synchronous Serial Controllers (SSC)
  - Independent Clock and Frame Sync Signals for Each Receiver and Transmitter
  - I<sup>2</sup>S Analog Interface Support, Time Division Multiplex Support
  - High-speed Continuous Data Stream Capabilities with 32-bit Data Transfer
- Four Universal Synchronous/Asynchronous Receiver/Transmitters (USART)
  - Support for ISO7816 T0/T1 Smart Card
  - Hardware and Software Handshaking
  - RS485 Support, IrDA Up To 115 Kbps
  - Full Modem Control Lines on USART1
- Master/Slave Serial Peripheral Interface (SPI)
  - 8- to 16-bit Programmable Data Length, 4 External Peripheral Chip Selects
- Two 3-channel, 16-bit Timer/Counters (TC)
  - Three External Clock Inputs, Two Multi-purpose I/O Pins per Channel
  - Double PWM Generation, Capture/Waveform Mode, Up/Down Capability
- Two-wire Interface (TWI)
  - Master Mode Support, All 2-wire Atmel EEPROMs Supported
- IEEE 1149.1 JTAG Boundary Scan on All Digital Pins
- Power Supplies
  - 1.65V to 1.95V for VDDCORE, VDDOSC and VDDPLL
  - 1.65V to 3.6V for VDDIOP (Peripheral I/Os) and for VDDIOM (Memory I/Os)
- Available in a 208-lead PQFP or 256-ball BGA Package



**ARM920T™ -  
based  
Microcontroller**

**AT91RM9200**





## Description

The AT91RM9200 is a complete system-on-chip built around the ARM920T ARM Thumb processor. It incorporates a rich set of system and application peripherals and standard interfaces in order to provide a single-chip solution for a wide range of compute-intensive applications that require maximum functionality at minimum power consumption at lowest cost.

The AT91RM9200 incorporates a high-speed on-chip SRAM workspace, and a low-latency External Bus Interface (EBI) for seamless connection to whatever configuration of off-chip memories and memory-mapped peripherals is required by the application. The EBI incorporates controllers for synchronous DRAM (SDRAM), Burst Flash and Static memories and features specific circuitry facilitating the interface for SmartMedia, CompactFlash and NAND Flash.

The Advanced Interrupt Controller (AIC) enhances the interrupt handling performance of the ARM920T processor by providing multiple vectored, prioritized interrupt sources and reducing the time taken to transfer to an interrupt handler.

The Peripheral Data Controller (PDC) provides DMA channels for all the serial peripherals, enabling them to transfer data to or from on- and off-chip memories without processor intervention. This reduces the processor overhead when dealing with transfers of continuous data streams. The AT91RM9200 benefits from a new generation of PDC which includes dual pointers that simplify significantly buffer chaining.

The set of Parallel I/O (PIO) controllers multiplex the peripheral input/output lines with general-purpose data I/Os for maximum flexibility in device configuration. An input change interrupt, open drain capability and programmable pull-up resistor is included on each line.

The Power Management Controller (PMC) keeps system power consumption to a minimum by selectively enabling/disabling the processor and various peripherals under software control. It uses an enhanced clock generator to provide a selection of clock signals including a slow clock (32 kHz) to optimize power consumption and performance at all times.

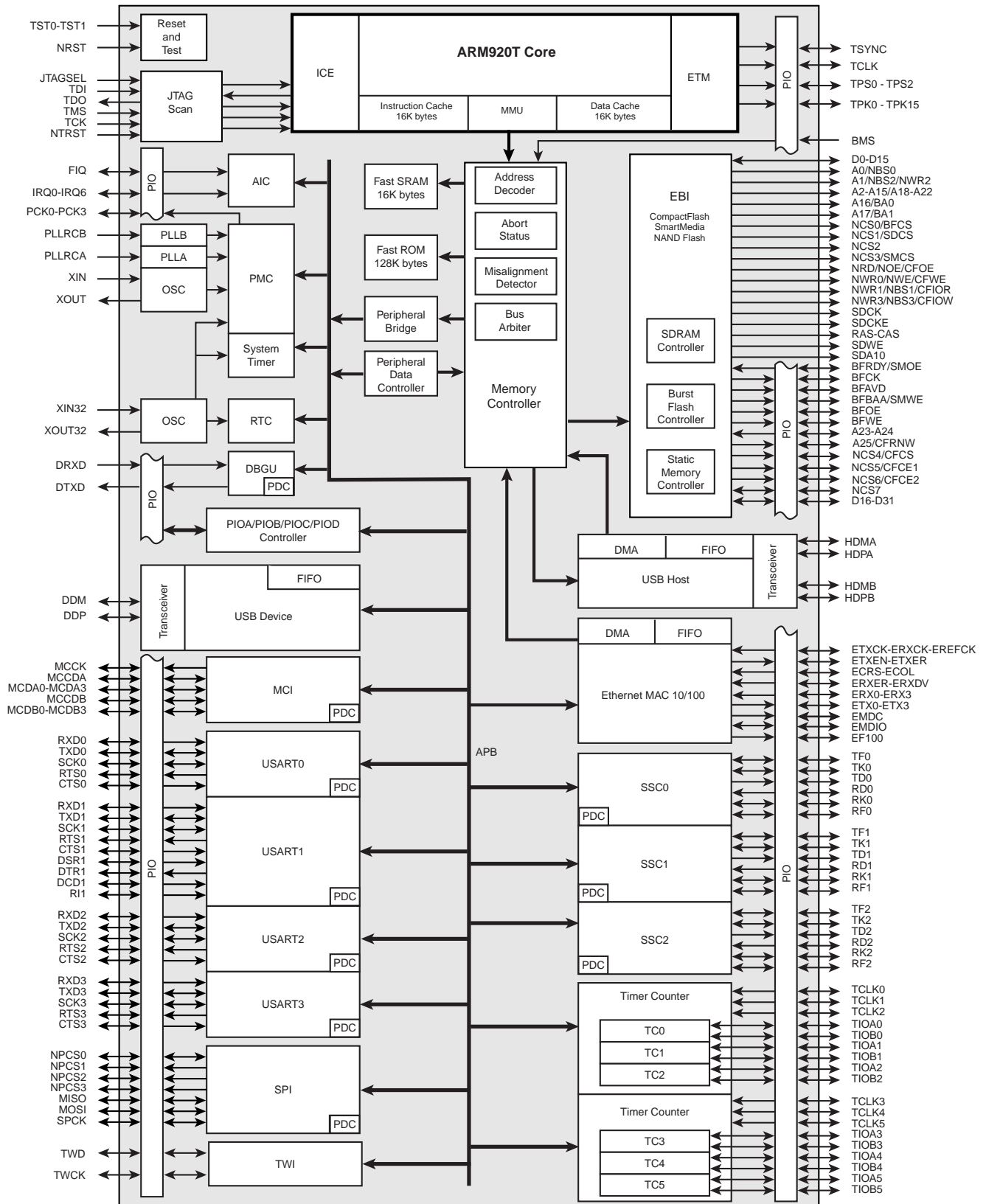
The AT91RM9200 integrates a wide range of standard interfaces including USB 2.0 Full Speed Host and Device and Ethernet 10/100 Base-T Media Access Controller (MAC), which provides connection to a extensive range of external peripheral devices and a widely used networking layer. In addition, it provides an extensive set of peripherals that operate in accordance with several industry standards, such as those used in audio, telecom, Flash Card, infrared and Smart Card applications.

To complete the offer, the AT91RM9200 benefits from the integration of a wide range of debug features including JTAG-ICE, a dedicated UART debug channel (DBGU) and an embedded real time trace. This enables the development and debug of all applications, especially those with real-time constraints.

## Block Diagram

Bold arrows ( **→** ) indicate master-to-slave dependency.

Figure 1. AT91RM9200 Block Diagram



## Key Features

This section presents the key features of each block.

### ARM920T Processor

- ARM9TDMI™ -based on ARM® Architecture v4T
- Two instruction sets
  - ARM® High-performance 32-bit Instruction Set
  - Thumb® High Code Density 16-bit Instruction Set
- 5-Stage Pipeline Architecture:
  - Instruction Fetch (F)
  - Instruction Decode (D)
  - Execute (E)
  - Data Memory (M)
  - Register Write (W)
- 16-Kbyte Data Cache, 16-Kbyte Instruction Cache
  - Virtually-addressed 64-way Associative Cache
  - 8 words per line
  - Write-through and write-back operation
  - Pseudo-random or Round-robin replacement
  - Low-power CAM RAM implementation
- Write Buffer
  - 16-word Data Buffer
  - 4-address Address Buffer
  - Software Control Drain
- Standard ARMv4 Memory Management Unit (MMU)
  - Access permission for sections
  - Access permission for large pages and small pages can be specified separately for each quarter of the pages
  - 16 embedded domains
  - 64 Entry Instruction TLB and 64 Entry Data TLB
- 8-, 16-, 32-bit Data Bus for Instructions and Data

### Debug and Test

- Integrated Embedded In-Circuit-Emulator
- Debug Unit
  - Two-pin UART
  - Debug Communication Channel
  - Chip ID Register
- Embedded Trace Macrocell: ETM9 Rev2a
  - Medium Level Implementation
  - Half-rate Clock Mode
  - Four Pairs of Address Comparators
  - Two Data Comparators
  - Eight Memory Map Decoder Inputs
  - Two Counters
  - One Sequencer
  - One 18-byte FIFO

- IEEE1149.1 JTAG Boundary Scan on all Digital Pins

## Boot Program

- Default Boot Program stored in ROM-based products
- Downloads and runs an application from external storage media into internal SRAM
- Downloaded code size depends on embedded SRAM size
- Automatic detection of valid application
- Bootloader supporting a wide range of non-volatile memories
  - SPI DataFlash<sup>®</sup> connected on SPI NPCSO
  - Two-wire EEPROM
  - 8-bit parallel memories on NCS0 if device integrates EBI
- Boot Uploader in case no valid program is detected in external NVM and supporting several communication media
- Serial communication on a DBGU (XModem protocol)
- USB Device Port (DFU Protocol)

## Embedded Software Services

- Compliant with ATPCS
- Compliant with AINSI/ISO Standard C
- Compiled in ARM/Thumb Interworking
- ROM Entry Service
- Tempo, Xmodem and DataFlash services
- CRC and Sine tables

## Reset Controller

- Two reset input lines (NRST and NTRST) providing, respectively:
- Initialization of the User Interface registers (defined in the user interface of each peripheral) and:
  - Sample the signals needed at bootup
  - Compel the processor to fetch the next instruction at address zero.
- Initialization of the embedded ICE TAP controller.

## Memory Controller

- Programmable Bus Arbiter handling four Masters
  - Internal Bus is shared by ARM920T, PDC, USB Host Port and Ethernet MAC Masters
  - Each Master can be assigned a priority between 0 and 7
- Address Decoder provides selection for
  - Eight external 256-Mbyte memory areas
  - Four internal 1-Mbyte memory areas
  - One 256-Mbyte embedded peripheral area
- Boot Mode Select Option
  - Non-volatile Boot Memory can be internal or external
  - Selection is made by BMS pin sampled at reset
- Abort Status Registers
  - Source, Type and all parameters of the access leading to an abort are saved
- Misalignment Detector
  - Alignment checking of all data accesses

- Abort generation in case of misalignment
- Remap command
  - Provides remapping of an internal SRAM in place of the boot NVM

## External Bus Interface

- Integrates three External Memory Controllers:
  - Static Memory Controller
  - SDRAM Controller
  - Burst Flash Controller
- Additional logic for SmartMedia™ and CompactFlash™ support
- Optimized External Bus:
  - 16- or 32-bit Data Bus
  - Up to 26-bit Address Bus, up to 64-Mbytes addressable
  - Up to 8 Chip Selects, each reserved to one of the eight Memory Areas
  - Optimized pin multiplexing to reduce latencies on External Memories
- Configurable Chip Select Assignment:
  - Burst Flash Controller or Static Memory Controller on NCS0
  - SDRAM Controller or Static Memory Controller on NCS1
  - Static Memory Controller on NCS3, Optional SmartMedia Support
  - Static Memory Controller on NCS4 - NCS6, Optional CompactFlash Support
  - Static Memory Controller on NCS7

## Static Memory Controller

- External memory mapping, 512-Mbyte address space
- Up to 8 Chip Select Lines
- 8- or 16-bit Data Bus
- Remap of Boot Memory
- Multiple Access Modes supported
  - Byte Write or Byte Select Lines
  - Two different Read Protocols for each Memory Bank
- Multiple device adaptability
  - Compliant with LCD Module
  - Programmable Setup Time Read/Write
  - Programmable Hold Time Read/Write
- Multiple Wait State Management
  - Programmable Wait State Generation
  - External Wait Request
  - Programmable Data Float Time

## SDRAM Controller

- Numerous configurations supported
  - 2K, 4K, 8K Row Address Memory Parts
  - SDRAM with two or four Internal Banks
  - SDRAM with 16- or 32-bit Data Path
- Programming facilities
  - Word, half-word, byte access
  - Automatic page break when Memory Boundary has been reached

- Multibank Ping-pong Access
- Timing parameters specified by software
- Automatic refresh operation, refresh rate is programmable
- Energy-saving capabilities
  - Self-refresh and Low-power Modes supported
- Error detection
  - Refresh Error Interrupt
- SDRAM Power-up Initialization by software
- Latency is set to two clocks (CAS Latency of 1, 3 Not Supported)
- Auto Precharge Command not used

## Burst Flash Controller

- Multiple Access Modes supported
  - Asynchronous or Burst Mode Byte, Half-word or Word Read Accesses
  - Asynchronous Mode Half-word Write Accesses
- Adaptability to different device speed grades
  - Programmable Burst Flash Clock Rate
  - Programmable Data Access Time
  - Programmable Latency after Output Enable
- Adaptability to different device access protocols and bus interfaces
  - Two Burst Read Protocols: Clock Control Address Advance or Signal Controlled Address Advance
  - Multiplexed or separate address and data buses
  - Continuous Burst and Page Mode Accesses supported

## Peripheral Data Controller

- Generates transfers to/from peripherals such as DBGU, USART, SSC, SPI and MCI
- Twenty channels
- One Master Clock cycle needed for a transfer from memory to peripheral
- Two Master Clock cycles needed for a transfer from peripheral to memory

## Advanced Interrupt Controller

- Controls the interrupt lines (nIRQ and nFIQ) of an ARM<sup>®</sup> Processor
- Thirty-two individually maskable and vectored interrupt sources
  - Source 0 is reserved for the Fast Interrupt Input (FIQ)
  - Source 1 is reserved for system peripherals (ST, RTC, PMC, DBGU...)
  - Source 2 to Source 31 control thirty embedded peripheral interrupts or external interrupts
  - Programmable Edge-triggered or Level-sensitive Internal Sources
  - Programmable Positive/Negative Edge-triggered or High/Low Level-sensitive External Sources
- 8-level Priority Controller
  - Drives the Normal Interrupt of the processor
  - Handles priority of the interrupt sources 1 to 31
  - Higher priority interrupts can be served during service of lower priority interrupt
- Vectoring
  - Optimizes Interrupt Service Routine Branch and Execution



- One 32-bit Vector Register per interrupt source
- Interrupt Vector Register reads the corresponding current Interrupt Vector
- Protect Mode
  - Easy debugging by preventing automatic operations
- Fast Forcing
  - Permits redirecting any normal interrupt source on the Fast Interrupt of the processor
- General Interrupt Mask
  - Provides processor synchronization on events without triggering an interrupt

## Power Management Controller

- Optimizes the power consumption of the whole system
- Embeds and controls:
  - One Main Oscillator and One Slow Clock Oscillator (32.768Hz)
  - Two Phase Locked Loops (PLLs) and Dividers
  - Clock Prescalers
- Provides:
  - the Processor Clock PCK
  - the Master Clock MCK
  - the USB Clocks, UHPCK and UDPCK, respectively for the USB Host Port and the USB Device Port
  - Programmable automatic PLL switch-off in USB Device suspend conditions
  - up to thirty peripheral clocks
  - four programmable clock outputs PCK0 to PCK3
- Four operating modes:
  - Normal Mode, Idle Mode, Slow Clock Mode, Standby Mode

## System Timer

- One Period Interval Timer, 16-bit programmable counter
- One Watchdog Timer, 16-bit programmable counter
- One Real-time Timer, 20-bit free-running counter
- Interrupt Generation on event

## Real Time Clock

- Low power consumption
- Full asynchronous design
- Two hundred year calendar
- Programmable Periodic Interrupt
- Alarm and update parallel load
- Control of alarm and update Time/Calendar Data In

## Debug Unit

- System peripheral to facilitate debug of Atmel's ARM<sup>®</sup>-based systems
- Composed of four functions
  - Two-pin UART
  - Debug Communication Channel (DCC) support
  - Chip ID Registers
  - ICE Access Prevention



- Two-pin UART
  - Implemented features are 100% compatible with the standard Atmel USART
  - Independent receiver and transmitter with a common programmable Baud Rate Generator
  - Even, Odd, Mark or Space Parity Generation
  - Parity, Framing and Overrun Error Detection
  - Automatic Echo, Local Loopback and Remote Loopback Channel Modes
  - Interrupt generation
  - Support for two PDC channels with connection to receiver and transmitter
- Debug Communication Channel Support
  - Offers visibility of COMMRX and COMMTX signals from the ARM Processor
  - Interrupt generation
- Chip ID Registers
  - Identification of the device revision, sizes of the embedded memories, set of peripherals

## PIO Controller

- Up to 32 programmable I/O Lines
- Fully programmable through Set/Clear Registers
- Multiplexing of two peripheral functions per I/O Line
- For each I/O Line (whether assigned to a peripheral or used as general purpose I/O)
  - Input change interrupt
  - Glitch filter
  - Multi-drive option enables driving in open drain
  - Programmable pull up on each I/O line
  - Pin data status register, supplies visibility of the level on the pin at any time
- Synchronous output, provides Set and Clear of several I/O lines in a single write

## USB Host Port

- Compliance with Open HCI Rev 1.0 specification
- Compliance with USB V2.0 Full-speed and Low-speed Specification
- Supports both Low-speed 1.5 Mbps and Full-speed 12 Mbps USB devices
- Root hub integrated with two downstream USB ports
- Two embedded USB transceivers
- Supports power management
- Operates as a master on the Memory Controller

## USB Device Port

- USB V2.0 full-speed compliant, 12 Mbits per second
- Embedded USB V2.0 full-speed transceiver
- Embedded dual-port RAM for endpoints
- Suspend/Resume logic
- Ping-pong mode (two memory banks) for isochronous and bulk endpoints
- Six general-purpose endpoints
  - Endpoint 0, Endpoint 3: 8 bytes, no ping-pong mode
  - Endpoint 1, Endpoint 2: 64 bytes, ping-pong mode
  - Endpoint 4, Endpoint 5: 256 bytes, ping-pong mode



## Ethernet MAC

- Compatibility with IEEE Standard 802.3
- 10 and 100 Mbits per second data throughput capability
- Full- and half-duplex operation
- MII or RMI interface to the physical layer
- Register interface to address, status and control registers
- DMA interface, operating as a master on the Memory Controller
- Interrupt generation to signal receive and transmit completion
- 28-byte transmit and 28-byte receive FIFOs
- Automatic pad and CRC generation on transmitted frames
- Address checking logic to recognize four 48-bit addresses
- Supports promiscuous mode where all valid frames are copied to memory
- Supports physical layer management through MDIO interface control of alarm and update time/calendar data in

## Serial Peripheral Interface

- Supports communication with serial external devices
  - Four chip selects with external decoder support allow communication with up to 15 peripherals
  - Serial memories, such as DataFlash and 3-wire EEPROMs
  - Serial peripherals, such as ADCs, DACs, LCD Controllers, CAN Controllers and Sensors
  - External co-processors
- Master or slave serial peripheral bus interface
  - 8- to 16-bit programmable data length per chip select
  - Programmable phase and polarity per chip select
  - Programmable transfer delays between consecutive transfers and between clock and data per chip select
  - Programmable delay between consecutive transfers
  - Selectable mode fault detection
- Connection to PDC channel optimizes data transfers
  - One channel for the receiver, one channel for the transmitter
  - Next buffer support

## Two-wire Interface

- Compatibility with standard two-wire serial memory
- One, two or three bytes for slave address
- Sequential read/write operations

## USART

- Programmable Baud Rate Generator
- 5- to 9-bit full-duplex synchronous or asynchronous serial communications
  - 1, 1.5 or 2 stop bits in Asynchronous Mode or 1 or 2 stop bits in Synchronous Mode
  - Parity generation and error detection
  - Framing error detection, overrun error detection
  - MSB- or LSB-first
  - Optional break generation and detection

- By 8 or by-16 over-sampling receiver frequency
- Optional hardware handshaking RTS-CTS
- Optional modem signal management DTR-DSR-DCD-RI
- Receiver time-out and transmitter timeguard
- Optional Multi-drop Mode with address generation and detection
- RS485 with driver control signal
- ISO7816, T = 0 or T = 1 Protocols for interfacing with smart cards
  - NACK handling, error counter with repetition and iteration limit
- IrDA modulation and demodulation
  - Communication at up to 115.2 Kbps
- Test Modes
  - Remote Loopback, Local Loopback, Automatic Echo
- Connection of two Peripheral Data Controller channels (PDC)
  - Offers buffer transfer without processor intervention

## Serial Synchronous Controller

- Provides serial synchronous communication links used in audio and telecom applications
- Contains an independent receiver and transmitter and a common clock divider
- Interfaced with two PDC channels (DMA access) to reduce processor overhead
- Offers a configurable frame sync and data length
- Receiver and transmitter can be programmed to start automatically or on detection of different event on the frame sync signal
- Receiver and transmitter include a data signal, a clock signal and a frame synchronization signal

## Timer Counter

- Three 16-bit Timer Counter Channels
- Wide range of functions including:
  - Frequency Measurement
  - Event Counting
  - Interval Measurement
  - Pulse Generation
  - Delay Timing
  - Pulse Width Modulation
  - Up/down Capabilities
- Each channel is user-configurable and contains:
  - Three external clock inputs
  - Five internal clock inputs
  - Two multi-purpose input/output signals
- Internal interrupt signal
- Two global registers that act on all three TC Channels

## MultiMedia Card Interface

- Compatibility with MultiMedia Card Specification Version 2.2
- Compatibility with SD Memory Card Specification Version 1.0
- Cards clock rate up to Master Clock divided by 2
- Embedded power management to slow down clock rate when not used

- Supports two slots
  - One slot for one MultiMediaCard bus (up to 30 cards) or one SD Memory Card
- Support for stream, block and multi-block data read and write
- Connection to a Peripheral Data Controller channel
  - Minimizes processor intervention for large buffer transfers

## AT91RM9200 Product Properties

### Power Supplies

The AT91RM9200 has five types of power supply pins:

- VDDCORE pins. They power the core, including processor, memories and peripherals; voltage ranges from 1.65V to 1.95V, 1.8V nominal.
- VDDIOM pins. They power the External Bus Interface I/O lines; voltage ranges from 1.65V to 3.6V, 1.8V, 3V or 3.3V nominal.
- VDDIOP pins. They power the Peripheral I/O lines and the USB transceivers; voltage ranges from 1.65V to 3.6V, 1.8V, 3V or 3.3V nominal.<sup>(1)</sup>
- VDDPLL pins. They power the PLL cells; voltage ranges from 1.65V to 1.95V, 1.8V nominal.
- VDDOSC pin. They power both oscillators; voltage ranges from 1.65V to 1.95V, 1.8V nominal.

Note: 1. Powering VDDIOP with a voltage lower than 3V prevents any use of the USB Host and Device Ports. This also affects the operation of the Trace Port.

The double power supplies VDDIOM and VDDIOP are identified in Table 1 on page 14 and Table 2 on page 16. These supplies enable the user to power the device differently for interfacing with memories and for interfacing with peripherals.

Ground pins are common to all power supplies, except VDDPLL and VDDOSC pins. For these pins, GNDPLL and GNDOSC are provided, respectively.

### Pinout

The AT91RM9200 is available in two packages:

- 208-lead PQFP, 31.2 x 31.2 mm, 0.5 mm lead pitch
- 256-ball BGA, 15 x 15 mm, 0.8 mm ball pitch

The product features of the 256-ball BGA package are extended compared to the 208-lead PQFP package. The features that are available only with the 256-ball BGA package are:

- Parallel I/O Controller D
- ETM port with outputs multiplexed on the PIO Controller D
- a second USB Host transceiver, opening the Hub capabilities of the embedded USB Host.

## 208-lead PQFP Package Pinout

**Table 1.** AT91RM9200 Pinout for 208-lead PQFP Package

| Pin Number | Signal Name | Pin Number | Signal Name | Pin Number | Signal Name | Pin Number | Signal Name  |
|------------|-------------|------------|-------------|------------|-------------|------------|--------------|
| 1          | PC24        | 37         | VDDPLL      | 73         | PA27        | 109        | TMS          |
| 2          | PC25        | 38         | PLLRCB      | 74         | PA28        | 110        | NTRST        |
| 3          | PC26        | 39         | GNDPLL      | 75         | VDDIOP      | 111        | VDDIOP       |
| 4          | PC27        | 40         | VDDIOP      | 76         | GND         | 112        | GND          |
| 5          | PC28        | 41         | GND         | 77         | PA29        | 113        | TST0         |
| 6          | PC29        | 42         | PA0         | 78         | PA30        | 114        | TST1         |
| 7          | VDDIOM      | 43         | PA1         | 79         | PA31/BMS    | 115        | NRST         |
| 8          | GND         | 44         | PA2         | 80         | PB0         | 116        | VDDCORE      |
| 9          | PC30        | 45         | PA3         | 81         | PB1         | 117        | GND          |
| 10         | PC31        | 46         | PA4         | 82         | PB2         | 118        | PB23         |
| 11         | PC10        | 47         | PA5         | 83         | PB3         | 119        | PB24         |
| 12         | PC11        | 48         | PA6         | 84         | PB4         | 120        | PB25         |
| 13         | PC12        | 49         | PA7         | 85         | PB5         | 121        | PB26         |
| 14         | PC13        | 50         | PA8         | 86         | PB6         | 122        | PB27         |
| 15         | PC14        | 51         | PA9         | 87         | PB7         | 123        | PB28         |
| 16         | PC15        | 52         | PA10        | 88         | PB8         | 124        | PB29         |
| 17         | PC0         | 53         | PA11        | 89         | PB9         | 125        | HDMA         |
| 18         | PC1         | 54         | PA12        | 90         | PB10        | 126        | HDPA         |
| 19         | VDDCORE     | 55         | PA13        | 91         | PB11        | 127        | DDM          |
| 20         | GND         | 56         | VDDIOP      | 92         | PB12        | 128        | DDP          |
| 21         | PC2         | 57         | GND         | 93         | VDDIOP      | 129        | VDDIOP       |
| 22         | PC3         | 58         | PA14        | 94         | GND         | 130        | GND          |
| 23         | PC4         | 59         | PA15        | 95         | PB13        | 131        | VDDIOM       |
| 24         | PC5         | 60         | PA16        | 96         | PB14        | 132        | GND          |
| 25         | PC6         | 61         | PA17        | 97         | PB15        | 133        | A0/NBS0      |
| 26         | VDDIOM      | 62         | VDDCORE     | 98         | PB16        | 134        | A1/NBS2/NWR2 |
| 27         | GND         | 63         | GND         | 99         | PB17        | 135        | A2           |
| 28         | VDDPLL      | 64         | PA18        | 100        | PB18        | 136        | A3           |
| 29         | PLLRCA      | 65         | PA19        | 101        | PB19        | 137        | A4           |
| 30         | GNDPLL      | 66         | PA20        | 102        | PB20        | 138        | A5           |
| 31         | XOUT        | 67         | PA21        | 103        | PB21        | 139        | A6           |
| 32         | XIN         | 68         | PA22        | 104        | PB22        | 140        | A7           |
| 33         | VDDOSC      | 69         | PA23        | 105        | JTAGSEL     | 141        | A8           |
| 34         | GNDOSC      | 70         | PA24        | 106        | TDI         | 142        | A9           |
| 35         | XOUT32      | 71         | PA25        | 107        | TDO         | 143        | A10          |
| 36         | XIN32       | 72         | PA26        | 108        | TCK         | 144        | SDA10        |

**Table 1.** AT91RM9200 Pinout for 208-lead PQFP Package (Continued)

| Pin Number | Signal Name | Pin Number | Signal Name     | Pin Number | Signal Name | Pin Number | Signal Name |
|------------|-------------|------------|-----------------|------------|-------------|------------|-------------|
| 145        | A11         | 161        | PC7             | 177        | CAS         | 193        | D10         |
| 146        | VDDIOM      | 162        | PC8             | 178        | SDWE        | 194        | D11         |
| 147        | GND         | 163        | PC9             | 179        | D0          | 195        | D12         |
| 148        | A12         | 164        | VDDIOM          | 180        | D1          | 196        | D13         |
| 149        | A13         | 165        | GND             | 181        | D2          | 197        | D14         |
| 150        | A14         | 166        | NCS0/BFCS       | 182        | D3          | 198        | D15         |
| 151        | A15         | 167        | NCS1/SDCS       | 183        | VDDIOM      | 199        | VDDIOM      |
| 152        | VDDCORE     | 168        | NCS2            | 184        | GND         | 200        | GND         |
| 153        | GND         | 169        | NCS3/SMCS       | 185        | D4          | 201        | PC16        |
| 154        | A16/BA0     | 170        | NRD/NOE/CFOE    | 186        | D5          | 202        | PC17        |
| 155        | A17/BA1     | 171        | NWR0/NWE/CFWE   | 187        | D6          | 203        | PC18        |
| 156        | A18         | 172        | NWR1/NBS1/CFIOR | 188        | VDDCORE     | 204        | PC19        |
| 157        | A19         | 173        | NWR3/NBS3/CFIOW | 189        | GND         | 205        | PC20        |
| 158        | A20         | 174        | SDCK            | 190        | D7          | 206        | PC21        |
| 159        | A21         | 175        | SDCKE           | 191        | D8          | 207        | PC22        |
| 160        | A22         | 176        | RAS             | 192        | D9          | 208        | PC23        |

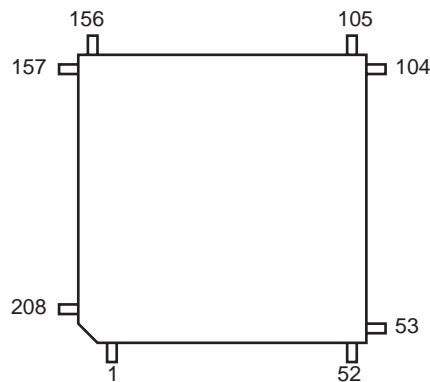
Note: 1. Shaded cells define the pins powered by VDDIOM.

## Mechanical Overview of the 208-lead PQFP Package

Figure 2 shows the orientation of the 208-lead PQFP package.

A detailed mechanical description is given in the section Mechanical Characteristics.

**Figure 2.** 208-lead PQFP Pinout (Top View)



## 256-ball BGA Package Pinout

**Table 2.** AT91RM9200 Pinout for 256-ball BGA Package

| Pin | Signal Name | Pin | Signal Name | Pin | Signal Name | Pin | Signal Name |
|-----|-------------|-----|-------------|-----|-------------|-----|-------------|
| A1  | TDI         | C3  | PD14        | E5  | TCK         | G14 | PA1         |
| A2  | JTAGSEL     | C4  | PB22        | E6  | GND         | G15 | PA2         |
| A3  | PB20        | C5  | PB19        | E7  | PB15        | G16 | PA3         |
| A4  | PB17        | C6  | PD10        | E8  | GND         | G17 | XIN32       |
| A5  | PD11        | C7  | PB13        | E9  | PB7         | H1  | PD23        |
| A6  | PD8         | C8  | PB12        | E10 | PB3         | H2  | PD20        |
| A7  | VDDIOP      | C9  | PB6         | E11 | PA29        | H3  | PD22        |
| A8  | PB9         | C10 | PB1         | E12 | PA26        | H4  | PD21        |
| A9  | PB4         | C11 | GND         | E13 | PA25        | H5  | VDDIOP      |
| A10 | PA31/BMS    | C12 | PA20        | E14 | PA9         | H13 | VDDPLL      |
| A11 | VDDIOP      | C13 | PA18        | E15 | PA6         | H14 | VDDIOP      |
| A12 | PA23        | C14 | VDDCORE     | E16 | PD3         | H15 | GNDPLL      |
| A13 | PA19        | C15 | GND         | E17 | PD0         | H16 | GND         |
| A14 | GND         | C16 | PA8         | F1  | PD16        | H17 | XOUT32      |
| A15 | PA14        | C17 | PD5         | F2  | GND         | J1  | PD25        |
| A16 | VDDIOP      | D1  | TST1        | F3  | PB23        | J2  | PD27        |
| A17 | PA13        | D2  | VDDIOP      | F4  | PB25        | J3  | PD24        |
| B1  | TDO         | D3  | VDDIOP      | F5  | PB24        | J4  | PD26        |
| B2  | PD13        | D4  | GND         | F6  | VDDCORE     | J5  | PB28        |
| B3  | PB18        | D5  | VDDIOP      | F7  | PB16        | J6  | PB29        |
| B4  | PB21        | D6  | PD7         | F9  | PB11        | J12 | GND         |
| B5  | PD12        | D7  | PB14        | F11 | PA30        | J13 | GNDOSC      |
| B6  | PD9         | D8  | VDDIOP      | F12 | PA28        | J14 | VDDOSC      |
| B7  | GND         | D9  | PB8         | F13 | PA4         | J15 | VDDPLLA     |
| B8  | PB10        | D10 | PB2         | F14 | PD2         | J16 | GNDPLLA     |
| B9  | PB5         | D11 | GND         | F15 | PD1         | J17 | XIN         |
| B10 | PB0         | D12 | PA22        | F16 | PA5         | K1  | HDPA        |
| B11 | VDDIOP      | D13 | PA21        | F17 | PLLRCB      | K2  | DDM         |
| B12 | PA24        | D14 | PA16        | G1  | PD19        | K3  | HDMA        |
| B13 | PA17        | D15 | PA10        | G2  | PD17        | K4  | VDDIOP      |
| B14 | PA15        | D16 | PD6         | G3  | GND         | K5  | DDP         |
| B15 | PA11        | D17 | PD4         | G4  | PB26        | K13 | PC5         |
| B16 | PA12        | E1  | NRST        | G5  | PD18        | K14 | PC4         |
| B17 | PA7         | E2  | NTRST       | G6  | PB27        | K15 | PC6         |
| C1  | TMS         | E3  | GND         | G12 | PA27        | K16 | VDDIOM      |
| C2  | PD15        | E4  | TST0        | G13 | PA0         | K17 | XOUT        |



**Table 2.** AT91RM9200 Pinout for 256-ball BGA Package (Continued)

| Pin | Signal Name  | Pin | Signal Name   | Pin | Signal Name         | Pin | Signal Name         |
|-----|--------------|-----|---------------|-----|---------------------|-----|---------------------|
| L1  | GND          | N2  | A5            | P13 | D15                 | T7  | NWR1/NBS1/<br>CFIOR |
| L2  | HDPB         | N3  | A9            | P14 | PC26                | T8  | SDWE                |
| L3  | HDMB         | N4  | A4            | P15 | PC27                | T9  | GND                 |
| L4  | A6           | N5  | A14           | P16 | VDDIOM              | T10 | VDDCORE             |
| L5  | GND          | N6  | SDA10         | P17 | GND                 | T11 | D9                  |
| L6  | VDDIOP       | N7  | A8            | R1  | GND                 | T12 | D12                 |
| L12 | PC10         | N8  | A21           | R2  | GND                 | T13 | GND                 |
| L13 | PC15         | N9  | NRD/NOE/CFOE  | R3  | A18                 | T14 | PC19                |
| L14 | PC2          | N10 | RAS           | R4  | A20                 | T15 | PC21                |
| L15 | PC3          | N11 | D2            | R5  | PC8                 | T16 | PC23                |
| L16 | VDDCORE      | N12 | GND           | R6  | VDDIOM              | T17 | PC25                |
| L17 | PLLRCA       | N13 | PC28          | R7  | NCS3/SMCS           | U1  | VDDCORE             |
| M1  | VDDIOM       | N14 | PC31          | R8  | NWR3/NBS3/<br>CFIOW | U2  | GND                 |
| M2  | GND          | N15 | PC30          | R9  | D0                  | U3  | A16/BA0             |
| M3  | A3           | N16 | PC11          | R10 | VDDIOM              | U4  | A19                 |
| M4  | A1/NBS2/NWR2 | N17 | PC12          | R11 | D8                  | U5  | GND                 |
| M5  | A10          | P1  | A7            | R12 | D13                 | U6  | NCS0/BFCS           |
| M6  | A2           | P2  | A13           | R13 | PC17                | U7  | SDCK                |
| M7  | GND          | P3  | A12           | R14 | VDDIOM              | U8  | CAS                 |
| M9  | NCS1/SDCS    | P4  | VDDIOM        | R15 | PC24                | U9  | D3                  |
| M11 | D4           | P5  | A11           | R16 | PC29                | U10 | D6                  |
| M12 | GND          | P6  | A22           | R17 | VDDIOM              | U11 | D7                  |
| M13 | PC13         | P7  | PC9           | T1  | A15                 | U12 | D11                 |
| M14 | PC1          | P8  | NWR0/NWE/CFWE | T2  | VDDCORE             | U13 | D14                 |
| M15 | PC0          | P9  | SDCKE         | T3  | A17/BA1             | U14 | PC16                |
| M16 | GND          | P10 | D1            | T4  | PC7                 | U15 | PC18                |
| M17 | PC14         | P11 | D5            | T5  | VDDIOM              | U16 | PC20                |
| N1  | A0/NBS0      | P12 | D10           | T6  | NCS2                | U17 | PC22                |

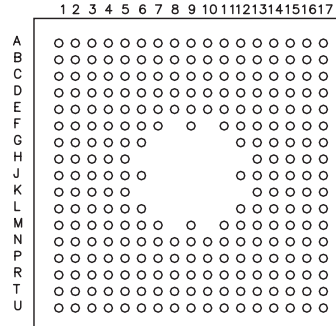
Note: 1. Shaded cells define the pins powered by VDDIOM.

## Mechanical Overview of the 256-ball BGA Package

Figure 3 on page 18 shows the orientation of the 256-ball BGA Package.

A detailed mechanical description is given in the section Mechanical Characteristics.

**Figure 3.** 256-ball BGA Pinout (Top View)



## Peripheral Multiplexing on PIO Lines

The AT91RM9200 features four PIO controllers:

- PIOA and PIOB, multiplexing I/O lines of the peripheral set.
- PIOC, multiplexing the data bus bits 16 to 31 and several External Bus Interface control signals. Using PIOC pins increases the number of general-purpose I/O lines available but prevents 32-bit memory access.
- PIOD, available in the 256-ball BGA package option only, multiplexing outputs of the peripheral set and the ETM port.

Each PIO Controller controls up to 32 lines. Each line can be assigned to one of two peripheral functions, A or B. The tables in the following paragraphs define how the I/O lines of the peripherals A and B are multiplexed on the PIO Controllers A, B, C and D. The two columns “Function” and “Comments” have been inserted for the user’s own comments; they may be used to track how pins are defined in an application.

The column “Reset State” indicates whether the PIO line resets in I/O mode or in peripheral mode. If equal to “I/O”, the PIO line resets in input with the pull-up enabled so that the device is maintained in a static state as soon as the NRST pin is asserted. As a result, the bit corresponding to the PIO line in the register PIO\_PSR (Peripheral Status Register) resets low.

If a signal name is in the “Reset State” column, the PIO line is assigned to this function and the corresponding bit in PIO\_PSR resets high. This is the case for pins controlling memories, either address lines or chip selects, and that require the pin to be driven as soon as NRST raises. Note that the pull-up resistor is also enabled in this case.

See Table 3 on page 19, Table 4 on page 20, Table 5 on page 21 and Table 6 on page 22.

### PIO Controller A Multiplexing

Table 3. Multiplexing on PIO Controller A

| PIO Controller A |              |              |             | Application Usage |          |
|------------------|--------------|--------------|-------------|-------------------|----------|
| I/O Line         | Peripheral A | Peripheral B | Reset State | Function          | Comments |
| PA0              | MISO         | PCK3         | I/O         |                   |          |
| PA1              | MOSI         | PCK0         | I/O         |                   |          |
| PA2              | SPCK         | IRQ4         | I/O         |                   |          |
| PA3              | NPCS0        | IRQ5         | I/O         |                   |          |
| PA4              | NPCS1        | PCK1         | I/O         |                   |          |
| PA5              | NPCS2        | TXD3         | I/O         |                   |          |
| PA6              | NPCS3        | RXD3         | I/O         |                   |          |
| PA7              | ETXCK/EREFCK | PCK2         | I/O         |                   |          |
| PA8              | ETXEN        | MCCDB        | I/O         |                   |          |
| PA9              | ETX0         | MCDB0        | I/O         |                   |          |
| PA10             | ETX1         | MCDB1        | I/O         |                   |          |
| PA11             | ECRS/ECRSVD  | MCDB2        | I/O         |                   |          |
| PA12             | ERX0         | MCDB3        | I/O         |                   |          |
| PA13             | ERX1         | TCLK0        | I/O         |                   |          |
| PA14             | ERXER        | TCLK1        | I/O         |                   |          |
| PA15             | EMDC         | TCLK2        | I/O         |                   |          |
| PA16             | EMDIO        | IRQ6         | I/O         |                   |          |
| PA17             | TXD0         | TIOA0        | I/O         |                   |          |
| PA18             | RXD0         | TIOB0        | I/O         |                   |          |
| PA19             | SCK0         | TIOA1        | I/O         |                   |          |
| PA20             | CTS0         | TIOB1        | I/O         |                   |          |
| PA21             | RTS0         | TIOA2        | I/O         |                   |          |
| PA22             | RXD2         | TIOB2        | I/O         |                   |          |
| PA23             | TXD2         | IRQ3         | I/O         |                   |          |
| PA24             | SCK2         | PCK1         | I/O         |                   |          |
| PA25             | TWD          | IRQ2         | I/O         |                   |          |
| PA26             | TWCK         | IRQ1         | I/O         |                   |          |
| PA27             | MCCK         | TCLK3        | I/O         |                   |          |
| PA28             | MCCDA        | TCLK4        | I/O         |                   |          |
| PA29             | MCDA0        | TCLK5        | I/O         |                   |          |
| PA30             | DRXD         | CTS2         | I/O         |                   |          |
| PA31             | DTXD         | RTS2         | I/O         |                   |          |

## PIO Controller B Multiplexing

**Table 4.** Multiplexing on PIO Controller B

| PIO Controller B |              |              |             | Application Usage |          |
|------------------|--------------|--------------|-------------|-------------------|----------|
| I/O Line         | Peripheral A | Peripheral B | Reset State | Function          | Comments |
| PB0              | TF0          | RTS3         | I/O         |                   |          |
| PB1              | TK0          | CTS3         | I/O         |                   |          |
| PB2              | TD0          | SCK3         | I/O         |                   |          |
| PB3              | RD0          | MCDA1        | I/O         |                   |          |
| PB4              | RK0          | MCDA2        | I/O         |                   |          |
| PB5              | RF0          | MCDA3        | I/O         |                   |          |
| PB6              | TF1          | TIOA3        | I/O         |                   |          |
| PB7              | TK1          | TIOB3        | I/O         |                   |          |
| PB8              | TD1          | TIOA4        | I/O         |                   |          |
| PB9              | RD1          | TIOB4        | I/O         |                   |          |
| PB10             | RK1          | TIOA5        | I/O         |                   |          |
| PB11             | RF1          | TIOB5        | I/O         |                   |          |
| PB12             | TF2          | ETX2         | I/O         |                   |          |
| PB13             | TK2          | ETX3         | I/O         |                   |          |
| PB14             | TD2          | ETXER        | I/O         |                   |          |
| PB15             | RD2          | ERX2         | I/O         |                   |          |
| PB16             | RK2          | ERX3         | I/O         |                   |          |
| PB17             | RF2          | ERXDV        | I/O         |                   |          |
| PB18             | RI1          | ECOL         | I/O         |                   |          |
| PB19             | DTR1         | ERXCK        | I/O         |                   |          |
| PB20             | TXD1         |              | I/O         |                   |          |
| PB21             | RXD1         |              | I/O         |                   |          |
| PB22             | SCK1         |              | I/O         |                   |          |
| PB23             | DCD1         |              | I/O         |                   |          |
| PB24             | CTS1         |              | I/O         |                   |          |
| PB25             | DSR1         | EF100        | I/O         |                   |          |
| PB26             | RTS1         |              | I/O         |                   |          |
| PB27             | PCK0         |              | I/O         |                   |          |
| PB28             | FIQ          |              | I/O         |                   |          |
| PB29             | IRQ0         |              | I/O         |                   |          |

## PIO Controller C Multiplexing

The PIO Controller C has no multiplexing and only peripheral A lines are used. Selecting Peripheral B on the PIO Controller C has no effect.

**Table 5.** Multiplexing on PIO Controller C

| PIO Controller C |              |              |             | Application Usage |          |
|------------------|--------------|--------------|-------------|-------------------|----------|
| I/O Line         | Peripheral A | Peripheral B | Reset State | Function          | Comments |
| PC0              | BFCK         |              | I/O         |                   |          |
| PC1              | BFRDY/SMOE   |              | I/O         |                   |          |
| PC2              | BFAVD        |              | I/O         |                   |          |
| PC3              | BFBA/SMWE    |              | I/O         |                   |          |
| PC4              | BFOE         |              | I/O         |                   |          |
| PC5              | BFWE         |              | I/O         |                   |          |
| PC6              | NWAIT        |              | I/O         |                   |          |
| PC7              | A23          |              | A23         |                   |          |
| PC8              | A24          |              | A24         |                   |          |
| PC9              | A25/CFRNW    |              | A25         |                   |          |
| PC10             | NCS4/CFCS    |              | NCS4        |                   |          |
| PC11             | NCS5/CFCE1   |              | NCS5        |                   |          |
| PC12             | NCS6/CFCE2   |              | NCS6        |                   |          |
| PC13             | NCS7         |              | NCS7        |                   |          |
| PC14             |              |              | I/O         |                   |          |
| PC15             |              |              | I/O         |                   |          |
| PC16             | D16          |              | I/O         |                   |          |
| PC17             | D17          |              | I/O         |                   |          |
| PC18             | D18          |              | I/O         |                   |          |
| PC19             | D19          |              | I/O         |                   |          |
| PC20             | D20          |              | I/O         |                   |          |
| PC21             | D21          |              | I/O         |                   |          |
| PC22             | D22          |              | I/O         |                   |          |
| PC23             | D23          |              | I/O         |                   |          |
| PC24             | D24          |              | I/O         |                   |          |
| PC25             | D25          |              | I/O         |                   |          |
| PC26             | D26          |              | I/O         |                   |          |
| PC27             | D27          |              | I/O         |                   |          |
| PC28             | D28          |              | I/O         |                   |          |
| PC29             | D29          |              | I/O         |                   |          |
| PC30             | D30          |              | I/O         |                   |          |
| PC31             | D31          |              | I/O         |                   |          |



## PIO Controller D Multiplexing

The PIO Controller D multiplexes pure output signals on peripheral A connections, in particular from the EMAC RMII interface and the ETM Port on the peripheral B connections.

The PIO Controller D is available only in the 256-ball BGA package option of the AT91RM9200.

**Table 6.** Multiplexing on PIO Controller D

| PIO Controller D |              |              |             | Application Usage |          |
|------------------|--------------|--------------|-------------|-------------------|----------|
| I/O Line         | Peripheral A | Peripheral B | Reset State | Function          | Comments |
| PD0              | ETX0         |              | I/O         |                   |          |
| PD1              | ETX1         |              | I/O         |                   |          |
| PD2              | ETX2         |              | I/O         |                   |          |
| PD3              | ETX3         |              | I/O         |                   |          |
| PD4              | ETXEN        |              | I/O         |                   |          |
| PD5              | ETXER        |              | I/O         |                   |          |
| PD6              | DTXD         |              | I/O         |                   |          |
| PD7              | PCK0         | TSYNC        | I/O         |                   |          |
| PD8              | PCK1         | TCLK         | I/O         |                   |          |
| PD9              | PCK2         | TPS0         | I/O         |                   |          |
| PD10             | PCK3         | TPS1         | I/O         |                   |          |
| PD11             |              | TPS2         | I/O         |                   |          |
| PD12             |              | TPK0         | I/O         |                   |          |
| PD13             |              | TPK1         | I/O         |                   |          |
| PD14             |              | TPK2         | I/O         |                   |          |
| PD15             | TD0          | TPK3         | I/O         |                   |          |
| PD16             | TD1          | TPK4         | I/O         |                   |          |
| PD17             | TD2          | TPK5         | I/O         |                   |          |
| PD18             | NPCS1        | TPK6         | I/O         |                   |          |
| PD19             | NPCS2        | TPK7         | I/O         |                   |          |
| PD20             | NPCS3        | TPK8         | I/O         |                   |          |
| PD21             | RTS0         | TPK9         | I/O         |                   |          |
| PD22             | RTS1         | TPK10        | I/O         |                   |          |
| PD23             | RTS2         | TPK11        | I/O         |                   |          |
| PD24             | RTS3         | TPK12        | I/O         |                   |          |
| PD25             | DTR1         | TPK13        | I/O         |                   |          |
| PD26             |              | TPK14        | I/O         |                   |          |
| PD27             |              | TPK15        | I/O         |                   |          |

## Pin Name Description

Table 7 gives details on the pin name classified by peripheral.

**Table 7.** Pin Description List

| Pin Name                            | Function                          | Type   | Active Level | Comments                              |
|-------------------------------------|-----------------------------------|--------|--------------|---------------------------------------|
| <b>Power</b>                        |                                   |        |              |                                       |
| VDDIOM                              | Memory I/O Lines Power Supply     | Power  |              | 1.65V to 3.6V                         |
| VDDIOP                              | Peripheral I/O Lines Power Supply | Power  |              | 1.65V to 3.6V                         |
| VDDPLL                              | Oscillator and PLL Power Supply   | Power  |              | 1.65V to 1.95V                        |
| VDDCORE                             | Core Chip Power Supply            | Power  |              | 1.65V to 1.95V                        |
| VDDOSC                              | Oscillator Power Supply           | Power  |              | 1.65V to 1.95V                        |
| GND                                 | Ground                            | Ground |              |                                       |
| GNDPLL                              | PLL Ground                        | Ground |              |                                       |
| GNDOSC                              | Oscillator Ground                 | Ground |              |                                       |
| <b>Clocks, Oscillators and PLLs</b> |                                   |        |              |                                       |
| XIN                                 | Main Crystal Input                | Input  |              |                                       |
| XOUT                                | Main Crystal Output               | Output |              |                                       |
| XIN32                               | 32KHz Crystal Input               | Input  |              |                                       |
| XOUT32                              | 32KHz Crystal Output              | Output |              |                                       |
| PLLRCA                              | PLL A Filter                      | Input  |              |                                       |
| PLLRCB                              | PLL B Filter                      | Input  |              |                                       |
| PCK0 - PCK3                         | Programmable Clock Output         | Output |              |                                       |
| <b>ICE and JTAG</b>                 |                                   |        |              |                                       |
| TCK                                 | Test Clock                        | Input  |              |                                       |
| TDI                                 | Test Data In                      | Input  |              |                                       |
| TDO                                 | Test Data Out                     | Output |              |                                       |
| TMS                                 | Test Mode Select                  | Input  |              |                                       |
| NTRST                               | Test Reset Signal                 | Input  | Low          |                                       |
| JTAGSEL                             | JTAG Selection                    | Input  |              |                                       |
| <b>ETM</b>                          |                                   |        |              |                                       |
| TSYNC                               | Trace Synchronization Signal      | Output |              |                                       |
| TCLK                                | Trace Clock                       | Output |              |                                       |
| TPS0 - TPS2                         | Trace ARM Pipeline Status         | Output |              |                                       |
| TPK0 - TPK15                        | Trace Packet Port                 | Output |              |                                       |
| <b>Reset/Test</b>                   |                                   |        |              |                                       |
| NRST                                | Microcontroller Reset             | Input  | Low          | No on-chip pull-up                    |
| TST0 - TST1                         | Test Mode Select                  | Input  |              | Must be tied low for normal operation |

**Table 7. Pin Description List (Continued)**

| Pin Name                            | Function                   | Type   | Active Level | Comments                 |
|-------------------------------------|----------------------------|--------|--------------|--------------------------|
| <b>Memory Controller</b>            |                            |        |              |                          |
| BMS                                 | Boot Mode Select           | Input  |              |                          |
| <b>Debug Unit</b>                   |                            |        |              |                          |
| DRXD                                | Debug Receive Data         | Input  |              | Debug Receive Data       |
| DTXD                                | Debug Transmit Data        | Output |              | Debug Transmit Data      |
| <b>AIC</b>                          |                            |        |              |                          |
| IRQ0 - IRQ6                         | External Interrupt Inputs  | Input  |              |                          |
| FIQ                                 | Fast Interrupt Input       | Input  |              |                          |
| <b>PIO</b>                          |                            |        |              |                          |
| PA0 - PA31                          | Parallel IO Controller A   | I/O    |              | Pulled-up input at reset |
| PB0 - PB29                          | Parallel IO Controller B   | I/O    |              | Pulled-up input at reset |
| PC0 - PC31                          | Parallel IO Controller C   | I/O    |              | Pulled-up input at reset |
| PD0 - PD27                          | Parallel IO Controller D   | I/O    |              | Pulled-up input at reset |
| <b>EBI</b>                          |                            |        |              |                          |
| D0 - D15                            | Data Bus                   | I/O    |              | Pulled-up input at reset |
| D16 - D31                           | Data Bus                   | I/O    |              | Pulled-up input at reset |
| A0 - A25                            | Address Bus                | Output |              | 0 at reset               |
| <b>SMC</b>                          |                            |        |              |                          |
| NCS0 - NCS7                         | Chip Select Lines          | Output | Low          | 1 at reset               |
| NWR0 - NWR3                         | Write Signal               | Output | Low          | 1 at reset               |
| NOE                                 | Output Enable              | Output | Low          | 1 at reset               |
| NRD                                 | Read Signal                | Output | Low          | 1 at reset               |
| NUB                                 | Upper Byte Select          | Output | Low          | 1 at reset               |
| NLB                                 | Lower Byte Select          | Output | Low          | 1 at reset               |
| NWE                                 | Write Enable               | Output | Low          | 1 at reset               |
| NBS0 - NBS3                         | Byte Mask Signal           | Output | Low          | 1 at reset               |
| <b>EBI for CompactFlash Support</b> |                            |        |              |                          |
| CFCE1 - CFCE2                       | CompactFlash Chip Enable   | Output | Low          |                          |
| CFOE                                | CompactFlash Output Enable | Output | Low          |                          |
| CFWE                                | CompactFlash Write Enable  | Output | Low          |                          |
| CFIOR                               | CompactFlash IO Read       | Output | Low          |                          |
| CFIOW                               | CompactFlash IO Write      | Output | Low          |                          |



**Table 7.** Pin Description List (Continued)

| Pin Name                          | Function                     | Type   | Active Level | Comments |
|-----------------------------------|------------------------------|--------|--------------|----------|
| CFRNW                             | CompactFlash Read Not Write  | Output |              |          |
| CFCS                              | CompactFlash Chip Select     | Output | Low          |          |
| <b>EBI for SmartMedia Support</b> |                              |        |              |          |
| SMCS                              | SmartMedia Chip Select       | Output | Low          |          |
| SMOE                              | SmartMedia Output Enable     | Output | Low          |          |
| SMWE                              | SmartMedia Write Enable      | Output | Low          |          |
| <b>SDRAM Controller</b>           |                              |        |              |          |
| SDCK                              | SDRAM Clock                  | Output |              |          |
| SDCKE                             | SDRAM Clock Enable           | Output | High         |          |
| SDCS                              | SDRAM Controller Chip Select | Output | Low          |          |
| BA0 - BA1                         | Bank Select                  | Output |              |          |
| SDWE                              | SDRAM Write Enable           | Output | Low          |          |
| RAS - CAS                         | Row and Column Signal        | Output | Low          |          |
| SDA10                             | SDRAM Address 10 Line        | Output |              |          |
| <b>Burst Flash Controller</b>     |                              |        |              |          |
| BFCK                              | Burst Flash Clock            | Output |              |          |
| BFCS                              | Burst Flash Chip Select      | Output | Low          |          |
| BFAVD                             | Burst Flash Address Valid    | Output | Low          |          |
| BFBA                              | Burst Flash Address Advance  | Output | Low          |          |
| BFOE                              | Burst Flash Output Enable    | Output | Low          |          |
| BFRDY                             | Burst Flash Ready            | Input  | High         |          |
| BFWE                              | Burst Flash Write Enable     | Output | Low          |          |
| <b>Multimedia Card Interface</b>  |                              |        |              |          |
| MCK                               | Multimedia Card Clock        | Output |              |          |
| MCCDA                             | Multimedia Card A Command    | I/O    |              |          |
| MCDA0 - MCDA3                     | Multimedia Card A Data       | I/O    |              |          |
| MCCDB                             | Multimedia Card B Command    | I/O    |              |          |
| MCDB0 - MCDB3                     | Multimedia Card B Data       | I/O    |              |          |
| <b>USART</b>                      |                              |        |              |          |
| SCK0 - SCK3                       | Serial Clock                 | I/O    |              |          |
| TXD0 - TXD3                       | Transmit Data                | Output |              |          |
| RXD0 - RXD3                       | Receive Data                 | Input  |              |          |
| RTS0 - RTS3                       | Ready To Send                | Output |              |          |
| CTS0 - CTS3                       | Clear To Send                | Input  |              |          |
| DSR1                              | Data Set Ready               | Input  |              |          |



**Table 7. Pin Description List (Continued)**

| Pin Name                             | Function                     | Type   | Active Level | Comments                 |
|--------------------------------------|------------------------------|--------|--------------|--------------------------|
| DTR1                                 | Data Terminal Ready          | Output |              |                          |
| DCD1                                 | Data Carrier Detect          | Input  |              |                          |
| RI1                                  | Ring Indicator               | Input  |              |                          |
| <b>USB Device Port</b>               |                              |        |              |                          |
| DDM                                  | USB Device Port Data -       | Analog |              |                          |
| DDP                                  | USB Device Port Data +       | Analog |              |                          |
| <b>USB Host Port</b>                 |                              |        |              |                          |
| HDMA                                 | USB Host Port A Data -       | Analog |              |                          |
| HDP A                                | USB Host Port A Data +       | Analog |              |                          |
| HDMB                                 | USB Host Port B Data -       | Analog |              |                          |
| HDPB                                 | USB Host Port B Data +       | Analog |              |                          |
| <b>Ethernet MAC</b>                  |                              |        |              |                          |
| EREFCK                               | Reference Clock              | Input  |              | RMII only                |
| ETXCK                                | Transmit Clock               | Input  |              | MII only                 |
| ERXCK                                | Receive Clock                | Input  |              | MII only                 |
| ETXEN                                | Transmit Enable              | Output |              |                          |
| ETX0 - ETX3                          | Transmit Data                | Output |              | ETX0 - ETX1 only in RMII |
| ETXER                                | Transmit Coding Error        | Output |              | MII only                 |
| ERXDV                                | Receive Data Valid           | Input  |              | MII only                 |
| ECRSDV                               | Carrier Sense and Data Valid | Input  |              | RMII only                |
| ERX0 - ERX3                          | Receive Data                 | Input  |              | ERX0 - ERX1 only in RMII |
| ERXER                                | Receive Error                | Input  |              |                          |
| ECRS                                 | Carrier Sense                | Input  |              | MII only                 |
| ECOL                                 | Collision Detected           | Input  |              | MII only                 |
| EMDC                                 | Management Data Clock        | Output |              |                          |
| EMDIO                                | Management Data Input/Output | I/O    |              |                          |
| EF100                                | Force 100 Mbits/sec.         | Output | High         | RMII only                |
| <b>Synchronous Serial Controller</b> |                              |        |              |                          |
| TD0 - TD2                            | Transmit Data                | Output |              |                          |
| RD0 - RD2                            | Receive Data                 | Input  |              |                          |
| TK0 - TK2                            | Transmit Clock               | I/O    |              |                          |
| RK0 - RK2                            | Receive Clock                | I/O    |              |                          |
| TF0 - TF2                            | Transmit Frame Sync          | I/O    |              |                          |
| RF0 - RF2                            | Receive Frame Sync           | I/O    |              |                          |

**Table 7.** Pin Description List (Continued)

| Pin Name                  | Function                     | Type   | Active Level | Comments |
|---------------------------|------------------------------|--------|--------------|----------|
| <b>Timer/Counter</b>      |                              |        |              |          |
| TCLK0 - TCLK5             | External Clock Input         | Input  |              |          |
| TIOA0 - TIOA5             | I/O Line A                   | I/O    |              |          |
| TIOB0 - TIOB5             | I/O Line B                   | I/O    |              |          |
| <b>SPI</b>                |                              |        |              |          |
| MISO                      | Master In Slave Out          | I/O    |              |          |
| MOSI                      | Master Out Slave In          | I/O    |              |          |
| SPCK                      | SPI Serial Clock             | I/O    |              |          |
| NPCS0                     | SPI Peripheral Chip Select 0 | I/O    | Low          |          |
| NPCS1 - NPCS3             | SPI Peripheral Chip Select   | Output | Low          |          |
| <b>Two-Wire Interface</b> |                              |        |              |          |
| TWD                       | Two-wire Serial Data         | I/O    |              |          |
| TWCK                      | Two-wire Serial Clock        | I/O    |              |          |

## Peripheral Identifiers

The AT91RM9200 embeds a wide range of peripherals. Table 8 defines the peripheral identifiers of the AT91RM9200. A peripheral identifier is required for the control of the peripheral interrupt with the Advanced Interrupt Controller and for the control of the peripheral clock with the Power Management Controller.

**Table 8.** Peripheral Identifiers

| Peripheral ID | Peripheral Mnemonic | Peripheral Name                 | External Interrupt |
|---------------|---------------------|---------------------------------|--------------------|
| 0             | AIC                 | Advanced Interrupt Controller   | FIQ                |
| 1             | SYSIRQ              |                                 |                    |
| 2             | PIOA                | Parallel I/O Controller A       |                    |
| 3             | PIOB                | Parallel I/O Controller B       |                    |
| 4             | PIOC                | Parallel I/O Controller C       |                    |
| 5             | PIOD                | Parallel I/O Controller D       |                    |
| 6             | US0                 | USART 0                         |                    |
| 7             | US1                 | USART 1                         |                    |
| 8             | US2                 | USART 2                         |                    |
| 9             | US3                 | USART 3                         |                    |
| 10            | MCI                 | Multimedia Card Interface       |                    |
| 11            | UDP                 | USB Device Port                 |                    |
| 12            | TWI                 | Two-wire Interface              |                    |
| 13            | SPI                 | Serial Peripheral Interface     |                    |
| 14            | SSC0                | Synchronous Serial Controller 0 |                    |
| 15            | SSC1                | Synchronous Serial Controller 1 |                    |
| 16            | SSC2                | Synchronous Serial Controller 2 |                    |
| 17            | TC0                 | Timer/Counter 0                 |                    |
| 18            | TC1                 | Timer/Counter 1                 |                    |
| 19            | TC2                 | Timer/Counter 2                 |                    |
| 20            | TC3                 | Timer/Counter 3                 |                    |
| 21            | TC4                 | Timer/Counter 4                 |                    |
| 22            | TC5                 | Timer/Counter 5                 |                    |
| 23            | UHP                 | USB Host Port                   |                    |
| 24            | EMAC                | Ethernet MAC                    |                    |
| 25            | AIC                 | Advanced Interrupt Controller   | IRQ0               |
| 26            | AIC                 | Advanced Interrupt Controller   | IRQ1               |
| 27            | AIC                 | Advanced Interrupt Controller   | IRQ2               |
| 28            | AIC                 | Advanced Interrupt Controller   | IRQ3               |
| 29            | AIC                 | Advanced Interrupt Controller   | IRQ4               |
| 30            | AIC                 | Advanced Interrupt Controller   | IRQ5               |
| 31            | AIC                 | Advanced Interrupt Controller   | IRQ6               |

**System Interrupt**

The System Interrupt is the wired-OR of the interrupt signals coming from:

- the Memory Controller
- the Debug Unit
- the System Timer
- the Real-Time Clock
- the Power Management Controller

The clock of these peripherals cannot be controlled and the Peripheral ID 1 can only be used within the Advanced Interrupt Controller.

**External Interrupts**

All external interrupt signals, i.e., the Fast Interrupt signal FIQ or the Interrupt signals IRQ0 to IRQ6, use a dedicated Peripheral ID. However, there is no clock control associated with these peripheral IDs.

## Product Memory Mapping

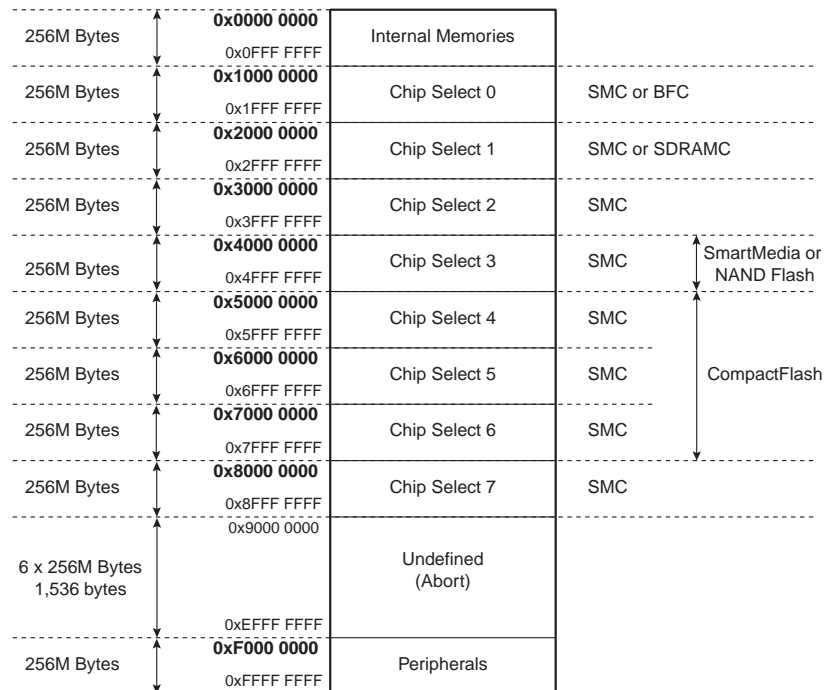
A first level of address decoding is performed by the Memory Controller, i.e., by the implementation of the Advanced System Bus (ASB) with additional features.

Decoding splits the 4G bytes of address space into 16 areas of 256M bytes. The areas 1 to 8 are directed to the EBI that associates these areas to the external chip selects NC0 to NCS7. The area 0 is reserved for the addressing of the internal memories, and a second level of decoding provides 1M bytes of internal memory area. The area 15 is reserved for the peripherals and provides access to the Advanced Peripheral Bus (APB).

Other areas are unused and performing an access within them provides an abort to the master requesting such an access.

## External Memory Mapping

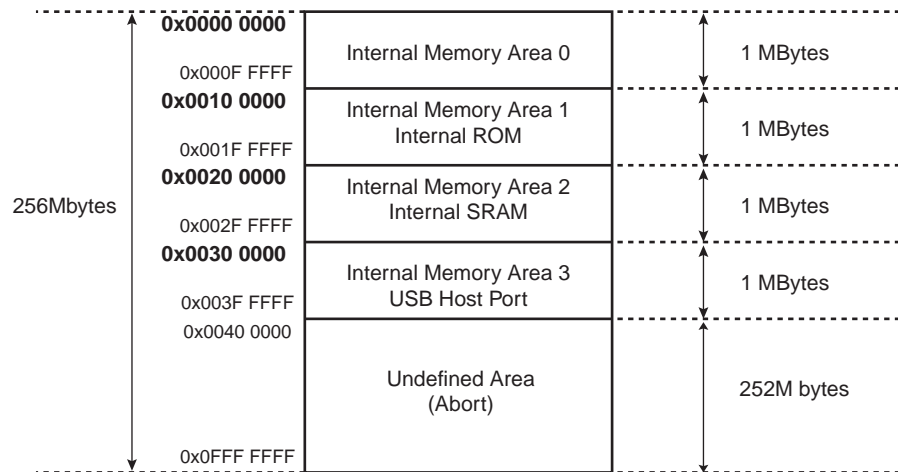
Figure 4. External Memory Mapping



## Internal Memory Mapping

- Internal RAM** The AT91RM9200 integrates a high-speed, 16-Kbyte internal SRAM. After reset and until the Remap Command is performed, the SRAM is only accessible at address 0x20 0000. After Remap, the SRAM is also available at address 0x0.
- Internal ROM** The AT91RM9200 integrates a 128-Kbyte Internal ROM. At any time, the ROM is mapped at address 0x10 0000. It is also accessible at address 0x0 after reset and before the Remap Command if the BMS is tied high during reset.
- USB Host Port** The AT91RM9200 integrates a USB Host Port Open Host Controller Interface (OHCI). The registers of this interface are directly accessible on the ASB Bus and are mapped like a standard internal memory at address 0x30 0000.

**Figure 5.** Internal Memory Mapping



## Peripheral Mapping

### System Peripherals Mapping

The System Peripherals are mapped to the top 4K bytes of the address space, between the addresses 0xFFFF F000 and 0xFFFF FFFF. Each peripheral has 256 or 512 bytes.

**Figure 6.** System Peripherals Mapping

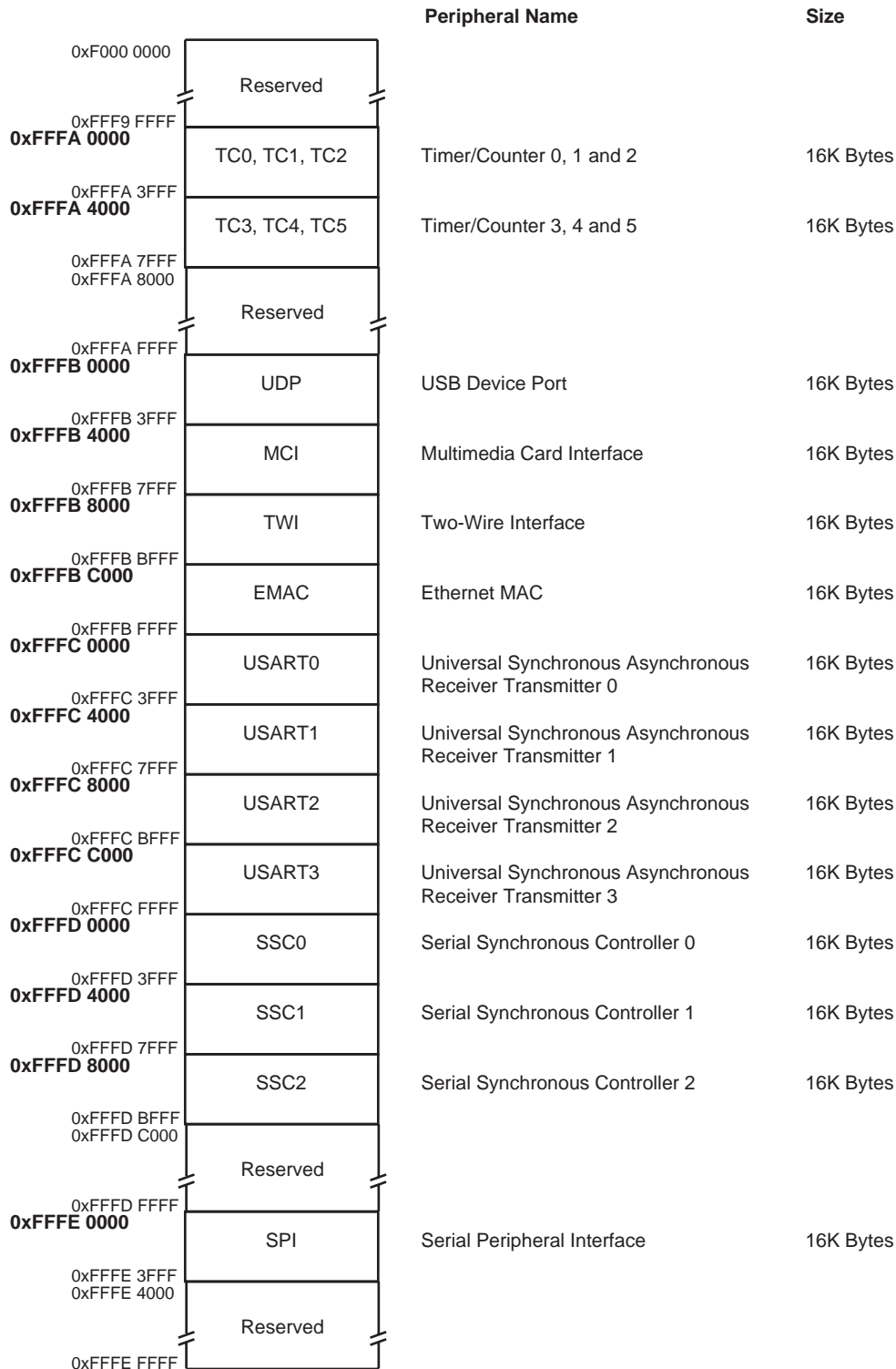
|                            |      | Peripheral Name               | Size                    |
|----------------------------|------|-------------------------------|-------------------------|
| 0xFFFF F000                | AIC  | Advanced Interrupt Controller | 512 bytes/128 registers |
| 0xFFFF F1FF<br>0xFFFF F200 | DBGU | Debug Unit                    | 512 bytes/128 registers |
| 0xFFFF F3FF<br>0xFFFF F400 | PIOA | PIO Controller A              | 512 bytes/128 registers |
| 0xFFFF F5FF<br>0xFFFF F600 | PIOB | PIO Controller B              | 512 bytes/128 registers |
| 0xFFFF F7FF<br>0xFFFF F800 | PIOC | PIO Controller C              | 512 bytes/128 registers |
| 0xFFFF F9FF<br>0xFFFF FA00 | PIOD | PIO Controller D              | 512 bytes/128 registers |
| 0xFFFF FBFF<br>0xFFFF FC00 | PMC  | Power Management Controller   | 256 bytes/64 registers  |
| 0xFFFF FCFF<br>0xFFFF FD00 | ST   | System Timer                  | 256 bytes/64 registers  |
| 0xFFFF FDFE<br>0xFFFF FE00 | RTC  | Real-time Clock               | 256 bytes/64 registers  |
| 0xFFFF FEFF<br>0xFFFF FF00 | MC   | Memory Controller             | 256 bytes/64 registers  |
| 0xFFFF FFFF                |      |                               |                         |



## User Peripherals Mapping

The User Peripherals are mapped in the upper 256M bytes of the address space, between the addresses 0xFFFA 0000 and 0xFFFE 3FFF. Each peripheral has a 16-Kbyte address space.

Figure 7. User Peripherals Mapping



## Peripheral Implementation

### USART

The USART describes features allowing management of the Modem Signals DTR, DSR, DCD and RI. For details, see “Modem Mode” on page 422.

In the AT91RM9200, only the USART1 implements these signals, named DTR1, DSR1, DCD1 and RI1.

The USART0, USART2 and USART3 do not implement all the modem signals. Only RTS and CTS (RTS0 and CTS0, RTS2 and CTS2, RTS3 and CTS3, respectively) are implemented in these USARTs for other features.

Thus, programming the USART0, USART2 or the USART3 in Modem Mode may lead to unpredictable results. In these USARTs, the commands relating to the Modem Mode have no effect and the status bits relating the status of the modem signals are never activated.

### Timer Counter

The Timer Counter 0 to 5 are described with five generic clock inputs, TIMER\_CLOCK1 to TIMER\_CLOCK5. In the AT91RM9200, these clock inputs are connected to the Master Clock (MCK), to the Slow Clock (SLCK) and to divisions of the Master Clock. For details, see “Clock Control” on page 476.

Table 2 gives the correspondence between the Timer Counter clock inputs and clocks in the AT91RM9200. Each Timer Counter 0 to 5 displays the same configuration.

**Table 2.** Timer Counter Clocks Assignment

| TC Clock Input | Clock   |
|----------------|---------|
| TIMER_CLOCK1   | MCK/2   |
| TIMER_CLOCK2   | MCK/8   |
| TIMER_CLOCK3   | MCK/32  |
| TIMER_CLOCK4   | MCK/128 |
| TIMER_CLOCK5   | SLCK    |

## ARM920T Processor Overview

### Overview

The ARM920T cached processor is a member of the ARM9™ Thumb® family of high-performance 32-bit system-on-a-chip processors. It provides a complete high performance CPU subsystem including:

- ARM9TDMI RISC integer CPU
- 16-Kbyte instruction and 16-Kbyte data caches
- Instruction and data memory management units (MMUs)
- Write buffer
- AMBA™ (Advanced Microprocessor Bus Architecture) bus interface
- Embedded Trace Macrocell (ETM) interface

The ARM9TDMI core within the ARM920T executes both the 32-bit ARM and 16-bit Thumb instruction sets. The ARM9TDMI processor is a Harvard architecture device, implementing a five-stage pipeline consisting of Fetch, Decode, Execute, Memory and Write stages.

The ARM920T processor incorporates two coprocessors:

- CP14 - Controls software access to the debug communication channel
- CP15 - System Control Processor, providing 16 additional registers that are used to configure and control the caches, the MMU, protection system, clocking mode and other system options

The main features of the ARM920T processor are:

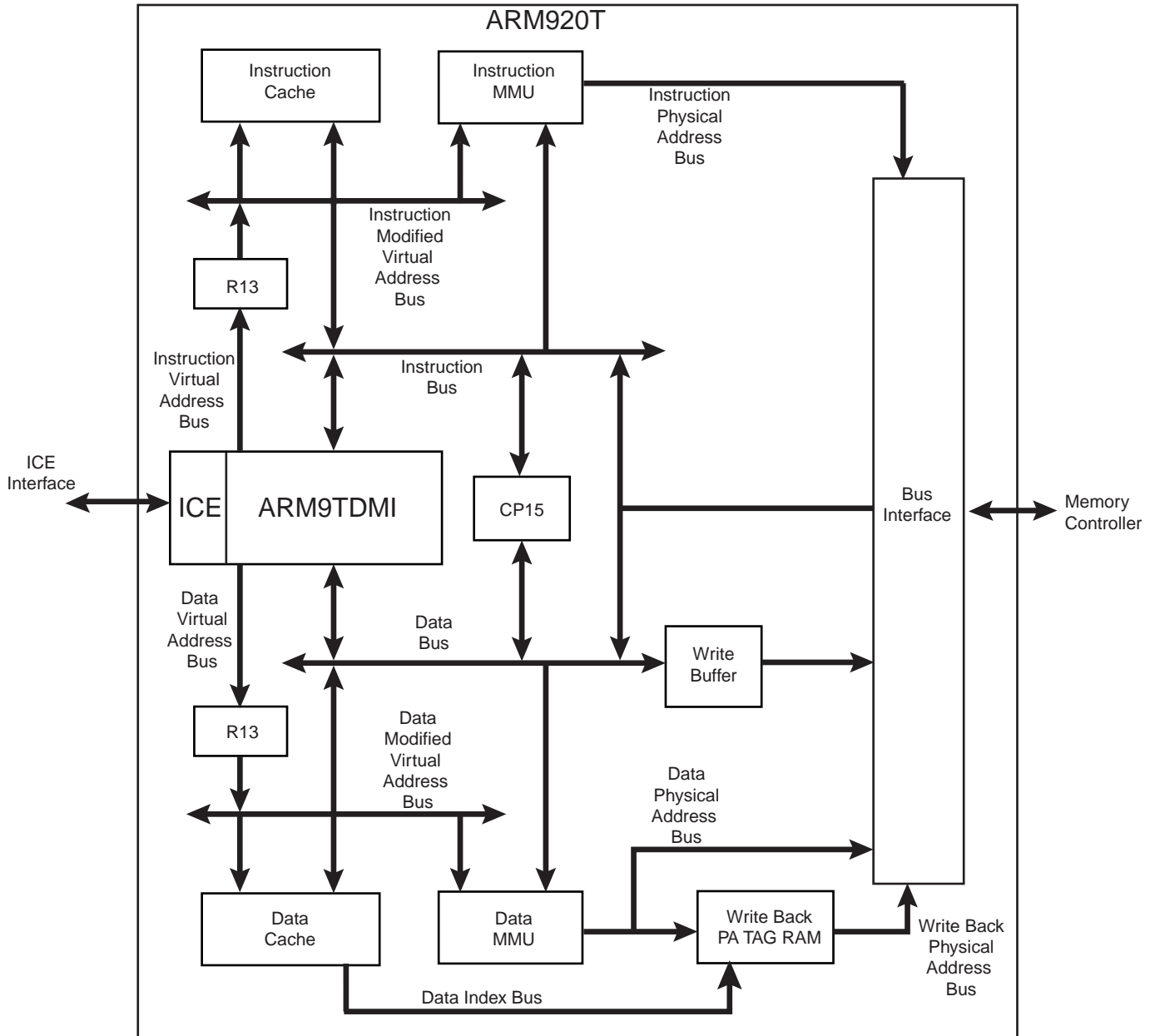
- ARM9TDMI®-based, ARM® Architecture v4T
- Two Instruction Sets
  - ARM High-performance 32-bit Instruction Set
  - Thumb High Code Density 16-bit Instruction Set
- 5-Stage Pipeline Architecture
  - Instruction Fetch (F)
  - Instruction Decode (D)
  - Execute (E)
  - Data Memory Access (M)
  - Register Write (W)
- 16-Kbyte Data Cache, 16-Kbyte Instruction Cache
  - Virtually-addressed 64-way Associative Cache
  - 8 Words per Line
  - Write-through and Write-back Operation
  - Pseudo-random or Round-robin Replacement
  - Low-power CAM RAM Implementation
- Write Buffer
  - 16-word Data Buffer
  - 4-address Address Buffer
  - Software Control Drain
- Standard ARMv4 Memory Management Unit (MMU)
  - Access Permission for Sections



- Access Permission for Large Pages and Small Pages Can be Specified Separately for Each Quarter of the Pages
- 16 Embedded Domains
- 64-entry Instruction TLB and 64-entry Data TLB
- 8-, 16-, 32-bit Data Bus for Instructions and Data

## Block Diagram

Figure 8. ARM920T Internal Functional Block Diagram



## ARM9TDMI Processor

### Instruction Type

Instructions are either 32 bits (in ARM state) or 16 bits (in Thumb state).

### Data Types

ARM9TDMI supports byte (8-bit), half-word (16-bit) and word (32-bit) data types. Words must be aligned to four-byte boundaries and half-words to two-byte boundaries.

Unaligned data access behavior depends on which instruction is used in a particular location.

### ARM9TDMI Operating Modes

The ARM9TDMI, based on ARM architecture v4T, supports seven processor modes:

- User: Standard ARM program execution state
- FIQ: Designed to support high-speed data transfer or channel processes
- IRQ: Used for general-purpose interrupt handling
- Supervisor: Protected mode for the operating system
- Abort mode: Implements virtual memory and/or memory protection
- System: A privileged user mode for the operating system
- Undefined: Supports software emulation of hardware coprocessors

Mode changes may be made under software control, or may be brought about by external interrupts or exception processing. Most application programs will execute in User Mode. The non-user modes, known as privileged modes, are entered in order to service interrupts or exceptions or to access protected resources.

## ARM9TDMI Registers

The ARM9TDMI processor core consists of a 32-bit datapath and associated control logic. That datapath contains 31 general-purpose registers, coupled to a full shifter, Arithmetic Logic Unit and multiplier.

At any one time, 16 registers are visible to the user. The remainder are synonyms used to speed up exception processing.

Register 15 is the Program Counter (PC) and can be used in all instructions to reference data relative to the current instruction.

R14 holds the return address after a subroutine call.

R13 is used (by software convention) as a stack pointer.

**Table 9.** ARM9TDMI Modes and Register Layout

| User and System Mode | Supervisor Mode | Abort Mode | Undefined Mode | Interrupt Mode | Fast Interrupt Mode |
|----------------------|-----------------|------------|----------------|----------------|---------------------|
| R0                   | R0              | R0         | R0             | R0             | R0                  |
| R1                   | R1              | R1         | R1             | R1             | R1                  |
| R2                   | R2              | R2         | R2             | R2             | R2                  |
| R3                   | R3              | R3         | R3             | R3             | R3                  |
| R4                   | R4              | R4         | R4             | R4             | R4                  |
| R5                   | R5              | R5         | R5             | R5             | R5                  |
| R6                   | R6              | R6         | R6             | R6             | R6                  |
| R7                   | R7              | R7         | R7             | R7             | R7                  |
| R8                   | R8              | R8         | R8             | R8             | R8_FIQ              |
| R9                   | R9              | R9         | R9             | R9             | R9_FIQ              |
| R10                  | R10             | R10        | R10            | R10            | R10_FIQ             |
| R11                  | R11             | R11        | R11            | R11            | R11_FIQ             |
| R12                  | R12             | R12        | R12            | R12            | R12_FIQ             |
| R13                  | R13_SVC         | R13_ABORT  | R13_UNDEF      | R13_IRQ        | R13_FIQ             |
| R14                  | R14_SVC         | R14_ABORT  | R14_UNDEF      | R14_IRQ        | R14_FIQ             |
| PC                   | PC              | PC         | PC             | PC             | PC                  |

|      |          |            |            |          |          |
|------|----------|------------|------------|----------|----------|
| CPSR | CPSR     | CPSR       | CPSR       | CPSR     | CPSR     |
|      | SPSR_SVC | SPSR_ABORT | SPSR_UNDEF | SPSR_IRQ | SPSR_FIQ |



Mode-specific banked registers

Registers R0 to R7 are unbanked registers, thus each of them refers to the same 32-bit physical register in all processor modes. They are general-purpose registers, with no special uses managed by the architecture, and can be used wherever an instruction allows a general-purpose register to be specified.

Registers R8 to R14 are banked registers. This means that each of them depends of the current processor mode.

For further details, see the ARM Architecture Reference Manual, Rev. DDI0100E.

## Modes and Exception Handling

All exceptions have banked registers for R14 and R13.

After an exception, R14 holds the return address for exception processing. This address is used both to return after the exception is processed and to address the instruction that caused the exception.

R13 is banked across exception modes to provide each exception handler with a private stack pointer.

The fast interrupt mode also banks registers 8 to 12 so that interrupt processing can begin without the need to save these registers.

A seventh processing mode, System Mode, does not have any banked registers. It uses the User Mode registers. System Mode runs tasks that require a privileged processor mode and allows them to invoke all classes of exceptions.

## Status Registers

All other processor states are held in status registers. The current operating processor status is in the Current Program Status Register (CPSR). The CPSR holds:

- four ALU flags (Negative, Zero, Carry, and Overflow),
- two interrupt disable bits (one for each type of interrupt),
- one bit to indicate ARM or Thumb execution
- five bits to encode the current processor mode

All five exception modes also have a Saved Program Status Register (SPSR) which holds the CPSR of the task immediately before the exception occurred.

## Exception Types

The ARM9TDMI supports five types of exceptions and a privileged processing mode for each type. The types of exceptions are:

- fast interrupt (FIQ)
- normal interrupt (IRQ)
- memory aborts (used to implement memory protection or virtual memory)
- attempted execution of an undefined instruction
- software interrupt (SWIs)

Exceptions are generated by internal and external sources.

More than one exception can occur at the same time.

When an exception occurs, the banked version of R14 and the SPSR for the exception mode are used to save the state.

To return after handling the exception, the SPSR is moved to the CPSR and R14 is moved to the PC. This can be done in two ways:

- use of a data-processing instruction with the S-bit set, and the PC as the destination
- use of the Load Multiple with Restore CPSR instruction (LDM)

## ARM Instruction Set Overview

The ARM instruction set is divided into:

- Branch instructions
- Data processing instructions
- Status register transfer instructions
- Load and Store instructions
- Coprocessor instructions
- Exception-generating instructions

ARM instructions can be executed conditionally. Every instruction contains a 4-bit condition code field (bits[31:28]).

For further details, see the ARM920T Technical Reference Manual, Rev. DDI0151C.

Table 10 gives the ARM instruction mnemonic list.

**Table 10.** ARM Instruction Mnemonic List

| Mnemonic | Operation                           |
|----------|-------------------------------------|
| MOV      | Move                                |
| ADD      | Add                                 |
| SUB      | Subtract                            |
| RSB      | Reverse Subtract                    |
| CMP      | Compare                             |
| TST      | Test                                |
| AND      | Logical AND                         |
| EOR      | Logical Exclusive OR                |
| MUL      | Multiply                            |
| SMULL    | Sign Long Multiply                  |
| SMLAL    | Signed Long Multiply Accumulate     |
| MSR      | Move to Status Register             |
| B        | Branch                              |
| BX       | Branch and Exchange                 |
| LDR      | Load Word                           |
| LDRSH    | Load Signed Halfword                |
| LDRSB    | Load Signed Byte                    |
| LDRH     | Load Half Word                      |
| LDRB     | Load Byte                           |
| LDRBT    | Load Register Byte with Translation |
| LDRT     | Load Register with Translation      |
| LDM      | Load Multiple                       |
| SWP      | Swap Word                           |
| MCR      | Move To Coprocessor                 |
| LDC      | Load To Coprocessor                 |

| Mnemonic | Operation                            |
|----------|--------------------------------------|
| CDP      | Coprocessor Data Processing          |
| MVN      | Move Not                             |
| ADC      | Add with Carry                       |
| SBC      | Subtract with Carry                  |
| RSC      | Reverse Subtract with Carry          |
| CMN      | Compare Negated                      |
| TEQ      | Test Equivalence                     |
| BIC      | Bit Clear                            |
| ORR      | Logical (inclusive) OR               |
| MLA      | Multiply Accumulate                  |
| UMULL    | Unsigned Long Multiply               |
| UMLAL    | Unsigned Long Multiply Accumulate    |
| MRS      | Move From Status Register            |
| BL       | Branch and Link                      |
| SWI      | Software Interrupt                   |
| STR      | Store Word                           |
| STRH     | Store Half Word                      |
| STRB     | Store Byte                           |
| STRBT    | Store Register Byte with Translation |
| STRT     | Store Register with Translation      |
| STM      | Store Multiple                       |
| SWPB     | Swap Byte                            |
| MRC      | Move From Coprocessor                |
| STC      | Store From Coprocessor               |



## Thumb Instruction Set Overview

The Thumb instruction set is a re-encoded subset of the ARM instruction set.

The Thumb instruction set is divided into:

- Branch instructions
- Data processing instructions
- Load and Store instructions
- Load and Store multiple instructions
- Exception-generating instruction

In Thumb mode, eight general-purpose registers are available, R0 to R7, that are the same physical registers as R0 to R7 when executing ARM instructions. Some Thumb instructions also access the Program Counter (ARM Register 15), the Link Register (ARM Register 14) and the Stack Pointer (ARM Register 13). Further instructions allow limited access to the ARM register 8 to 15.

For further details, see the ARM920T Technical Reference Manual, Rev. DDI0151C.

Table 11 gives the Thumb instruction mnemonic list.

**Table 11.** Thumb Instruction Mnemonic List

| Mnemonic | Operation              |
|----------|------------------------|
| MOV      | Move                   |
| ADD      | Add                    |
| SUB      | Subtract               |
| CMP      | Compare                |
| TST      | Test                   |
| AND      | Logical AND            |
| EOR      | Logical Exclusive OR   |
| LSL      | Logical Shift Left     |
| ASR      | Arithmetic Shift Right |
| MUL      | Multiply               |
| B        | Branch                 |
| BX       | Branch and Exchange    |
| LDR      | Load Word              |
| LDRH     | Load Half Word         |
| LDRB     | Load Byte              |
| LDRSH    | Load Signed Halfword   |
| LDMIA    | Load Multiple          |
| PUSH     | Push Register to stack |

| Mnemonic | Operation               |
|----------|-------------------------|
| MVN      | Move Not                |
| ADC      | Add with Carry          |
| SBC      | Subtract with Carry     |
| CMN      | Compare Negated         |
| NEG      | Negate                  |
| BIC      | Bit Clear               |
| ORR      | Logical (inclusive) OR  |
| LSR      | Logical Shift Right     |
| ROR      | Rotate Right            |
|          |                         |
| BL       | Branch and Link         |
| SWI      | Software Interrupt      |
| STR      | Store Word              |
| STRH     | Store Half Word         |
| STRB     | Store Byte              |
| LDRSB    | Load Signed Byte        |
| STMIA    | Store Multiple          |
| POP      | Pop Register from stack |

## CP15 Coprocessor

Coprocessor 15, or System Control Coprocessor CP15, is used when special features are used with the ARM9TDMI such as:

- On-chip Memory Management Unit (MMU)
- Instruction and/or Data Cache
- Write buffer

To control these features, CP15 provides 16 additional registers. See Table 12.

**Table 12.** CP15 Registers

| Register | Name                          | Access     |
|----------|-------------------------------|------------|
| 0        | ID Register                   | Read-only  |
| 1        | Control                       | Read/write |
| 2        | Translation Table Base        | Read/write |
| 3        | Domain Access Control         | Read/write |
| 4        | Reserved                      | None       |
| 5        | Fault Status                  | Read/write |
| 6        | Fault Address                 | Read/write |
| 7        | Cache Operations              | Write-only |
| 8        | TLB <sup>(1)</sup> Operations | Write-only |
| 9        | cache lockdown                | Read/write |
| 10       | TLB lockdown                  | Read/write |
| 11       | Reserved                      | None       |
| 12       | Reserved                      | None       |
| 13       | FCSE PID <sup>(2)</sup>       | Read/write |
| 14       | Reserved                      | None       |
| 15       | Test configuration            | None       |

- Notes: 1. TLB: Translation Lookaside Buffer  
 2. FCSE PID: Fast Context Switch Extension Process Identifier

## CP15 Register Access

CP15 registers can only be accessed in privileged mode by:

- MCR (Move to Coprocessor from ARM Register) instruction
- MRC (Move to ARM Register from Coprocessor) instruction

Other instructions (CDP, LDC, STC) cause an undefined instruction exception.

The MCR instruction is used to write an ARM register to CP15.

The MRC instruction is used to read the value of CP15 to an ARM register.

The assembler code for these instructions is:

```
MCR/MRC{cond} p15, opcode_1, Rd, CRn, CRm, opcode_2.
```

The MCR, MRC instructions bit pattern is shown below:

|          |    |    |    |     |    |    |    |
|----------|----|----|----|-----|----|----|----|
| 31       | 30 | 29 | 28 | 27  | 26 | 25 | 24 |
| Cond     |    |    |    | 1   | 1  | 1  | 0  |
| 23       | 22 | 21 | 20 | 19  | 18 | 17 | 16 |
| opcode_1 |    |    | L  | CRn |    |    |    |
| 15       | 14 | 13 | 12 | 11  | 10 | 9  | 8  |
| Rd       |    |    |    | 1   | 1  | 1  | 1  |
| 7        | 6  | 5  | 4  | 3   | 2  | 1  | 0  |
| opcode_2 |    |    | 1  | CRm |    |    |    |

- **CRm[3:0]: Specified Coprocessor Action**

Determines specific coprocessor action. Its value is dependent on the CP15 register used. For details, refer to CP15 specific register behavior.

- **opcode\_2[7:5]**

Determines specific coprocessor operation code. By default, set to 0.

- **Rd[15:12]: ARM Register**

Defines the ARM register whose value is transferred to the coprocessor. If R15 is chosen, the result is unpredictable.

- **CRn[19:16]: Coprocessor Register**

Determines the destination coprocessor register.

- **opcode\_1[23:20]: Coprocessor Code**

Defines the coprocessor specific code. Value is c15 for CP15.

- **L: Instruction Bit**

0 = MCR instruction

1 = MRC instruction

- **Cond [31:28]: Condition**

## Memory Management Unit (MMU)

The ARM920T processor implements an enhanced ARM architecture v4 MMU to provide translation and access permission checks for the instruction and data address ports of the ARM9TDMI core. The MMU is controlled from a single set of two-level page tables stored in the main memory, providing a single address and translation protection scheme. Independently, instruction and data TLBs in the MMU can be locked and flushed.

**Table 13.** Mapping Details

| Mapping Name | Mapping Size | Access Permission By | Subpage Size |
|--------------|--------------|----------------------|--------------|
| Section      | 1M byte      | Section              | -            |
| Large Page   | 64K bytes    | 4 separated subpages | 16K bytes    |
| Small Page   | 4K bytes     | 4 separated subpages | 1K byte      |
| Tiny Page    | 1K byte      | Tiny Page            | -            |

### Domain

A domain is a collection of sections and pages. The ARM920T supports 16 domains. Access to the domains is controlled by the Domain Access Control register. For details, refer to “CP15 Register 3, Domain Access Control Register” on page 52.

### MMU Faults

The MMU generates alignment faults, translation faults, domain faults and permission faults. Alignment fault checking is not affected by whether the MMU is enabled or not.

The access controls of the MMU detect the conditions that produce these faults. If the fault is a result of memory access, the MMU aborts the access and signals the fault to the CPU core. The MMU stores the status and address fault in the FSR and FAR registers (only for faults generated by data access).

The MMU does not store fault information about faults generated by an instruction fetch.

The memory system can abort during line fetches, memory accesses and translation table access.

## Caches, Write Buffers and Physical Address

The ARM920T includes an Instruction Cache (ICache), a Data Cache (DCache), a write buffer and a Physical Address (PA) TAG RAM to reduce the effect on main memory bandwidth and latency performance.

The ARM920T implements separate 16-Kbyte Instruction and 16-Kbyte Data Caches.

The caches and the write buffer are controlled by the CP15 Register 1 (Control), CP15 Register 7 (Cache Operations) and CP15 Register 9 (Cache Lockdown).

### Instruction Cache (ICache)

The ARM920T includes a 16-Kbyte Instruction Cache (ICache). The ICache has 512 lines of 32 bytes, arranged as a 64-way set associative cache.

Instruction access is subject to MMU permission and translation checks.

If the ICache is enabled with the MMU disabled, all instructions fetched as threats are cachable. No protection checks are made and the physical address is flat-mapped to the modified virtual address.

When the ICache is disabled, the cache contents are ignored and all instruction fetches appear on the AMBA bus.

On reset, the ICache entries are invalidated and the ICache is disabled. For best performance, ICache should be enabled as soon as possible after reset.

The ICache is enabled by writing 1 to I bit of the CP15 Register 1 and disabled by writing 0 to this bit. For more details, see “CP15 Register 1, Control” on page 49.

The ICache is organized as eight segments, each containing 64 lines with each line made up of 8 words. The position of the line within the segment is called the index and is numbered from 0 to 63.

A line in the cache is identified by the index and segment. The index is independent of the MVA (Modified Virtual Address), and the segment is the bit[7:5] of the MVA.

### Data Cache (DCache) and Write Buffer

The ARM920T includes a 16-Kbyte data cache (DCache). The DCache has 512 lines of 32 bytes, arranged as a 64-way set associative cache, and uses MVAs translated by CP15 Register 13 from the ARM9DTMI core.

#### DCache

The DCache is organized as eight segments, each containing 64 lines with each line made up of eight words. The position of the line within the segment is called the index and is a number from 0 to 63.

The Write Buffer can hold up to 16 words of data and four separate addresses.

DCache and Write Buffer operations are closely connected as their configuration is set in each section by the page descriptor in the MMU translation table.

All data accesses are subject to MMU permission and translation checks. Data accesses aborted by the MMU cannot cause linefill or data access via the AMBA ASB interface.

#### Write-through Operation

When a cache hit occurs for a data access, the cache line that contains the data is updated to contains its value. The new data is also immediately written to the main memory.

#### Write-back Operation

When a cache hit occurs for a data access, the cache line is marked as dirty, meaning that its contents are not up-to-date with those in the main memory.

### **Write Buffer**

The ARM920T incorporates a 16-entry write buffer to avoid stalling the processor when writes to external memory are performed. When a store occurs, its data, address and other details are written to the write buffer at high speed. The write buffer then completes the store at the main memory speed (typically slower than the ARM speed). In parallel, the ARM9TDMI processor can execute further instructions at full speed.

### **Physical Address Tag RAM (PA TAG RAM)**

The ARM920T implements Physical Address Tag RAM (PA TAG RAM) to perform write-backs from the data cache. The physical address of all the lines held in the data cache is stored in the PA TAG memory, removing the need for address translation when evicting a line from the cache.

When a line is written into the data cache, the physical address TAG is written into the PA TAG RAM. If this line has to be written back to the main memory, the PA TAG RAM is read and the physical address is used by the AMBA ASB interface to perform the write-back.

For a 16-Kbyte DCache, the PA TAG RAM is organized by eight segments with:

- 64 rows per segments
- 26 bits per rows

## ARM920T User Interface

### CP15 Register 0, ID Code and Cache Type

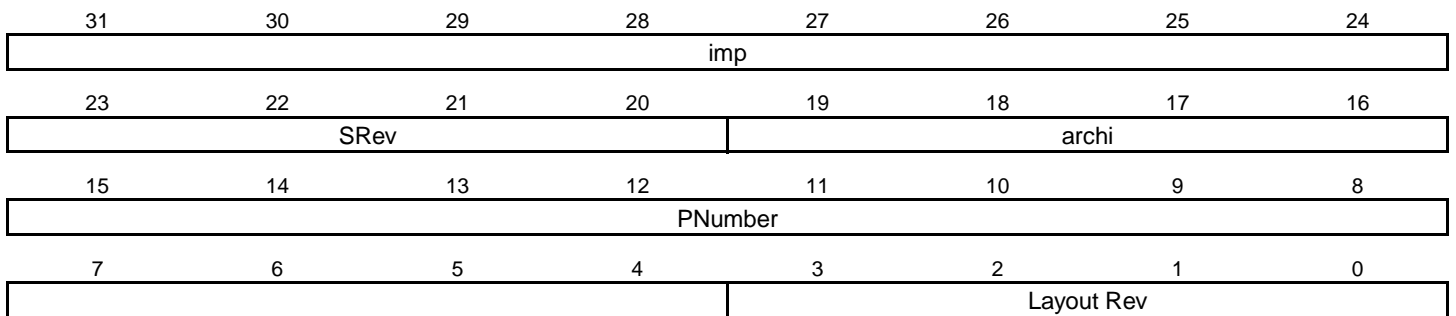
**Access:** Read-only

The CP Register 0 contains specific hardware information. The contents of the read accesses are determined by the opcode\_2 field value. Writing to Register 0 is unpredictable.

#### ID Code

The ID code register is accessed by reading the register 0 with the opcode\_2 field set to 0.

The contents of the ID code is shown below:



- **LayoutRev[3:0]: Revision**

Contains the processor revision number

- **PNumber[15:4]: Processor Part Number**

0x920 value for ARM920T processor.

- **archi[19:16]: Architecture**

Details the implementor architecture code.

0x2 value means ARMv4T architecture.

- **SRev[23:20]: Specification Revision Number**

0x1 value; specification revision number used to distinguished two variants of the same primary part.

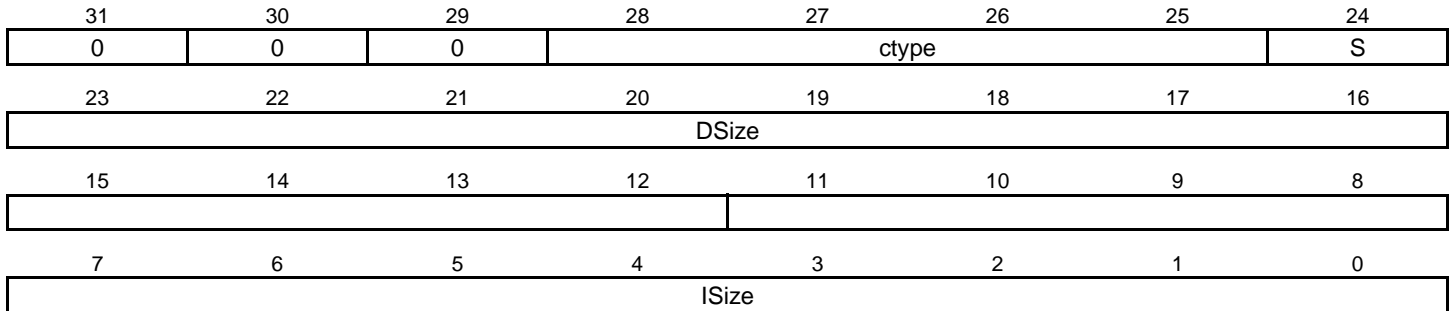
- **imp[31:24]: Implementor Code**

0x41 (= A); means ARM Ltd.

## Cache Type

The Cache Type register is accessed by reading the register 0 with the opcode\_2 field set to 1.

The Cache Type register contains information about the size and architecture of the caches.



- **ISize[11:0]: Instruction Cache Size**

Indicates the size, line length and associativity of the instruction cache.

- **DSize[23:12]: Data Cache Size**

Indicates the size, line length and associativity of the data cache.

- **S[24]: Cache**

Indicates if the cache is unified or has separate instruction and data caches.

Set to 1, this field indicates separate Instruction and Data caches.

- **ctype[28:25]: Cache Type**

Defines the cache type.

For details on bits DSize and ISize, refer to the ARM920T Technical Reference Manual, Rev. DDI0151C.



## CP15 Register 1, Control

**Access:** Read/write

The CP15 Register 1, or Control Register, contains the control bits of the ARM920T.

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| iA | nF | -  | -  | -  | -  | -  | -  |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| -  | -  | -  | -  | -  | -  | -  | -  |
| 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| -  | RR | V  | I  | 0  | 0  | R  | S  |
| 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| B  | 1  | 1  | 1  | 1  | C  | A  | M  |

- **M[0]: MMU Enable**

0 = MMU disabled.

1 = MMU enabled.

- **A[1]: Alignment Fault Enable**

0 = Fault checking disabled.

1 = Fault checking enabled.

- **C[2]: DCache Enable**

0 = DCache disabled.

1 = DCache enabled.

- **B[7]: Endianness**

0 = Little endian mode.

1 = Big endian mode.

- **S[8]: System Protection**

Modifies the MMU protection system.

For further details, see the ARM920T Technical Reference Manual, Rev. DDI0151C.

- **R[9]: ROM Protection**

Modifies the MMU protection system.

For further details, see the ARM920T Technical Reference Manual, Rev. DDI0151C.

- **I[12]: ICache Control**

0 = ICache disabled.

1 = ICache enabled.

- **V[13]: Base Location of Exception Register**

0 = Low address means 0x00000000.

1 = High address means 0xFFFF0000.

- **RR[14]: Round Robin Replacement**

0 = Random replacement.

1 = Round robin replacement.

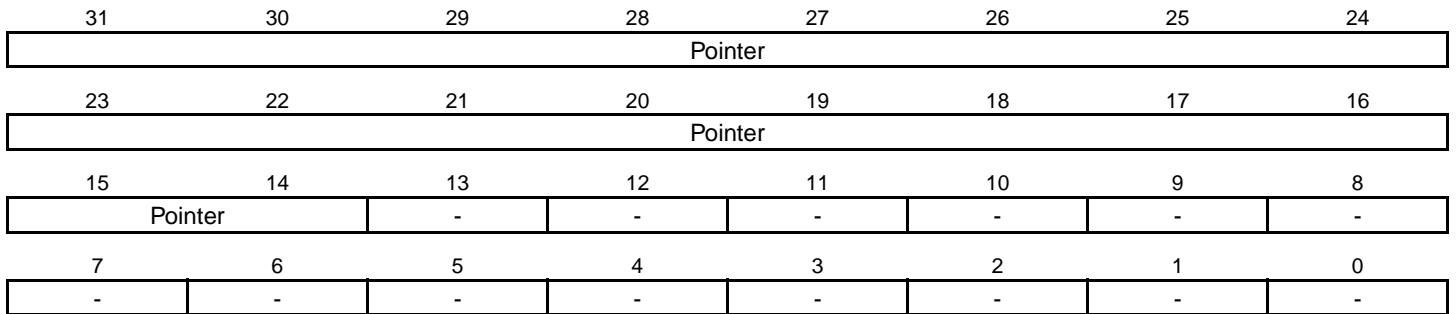
- **Clocking Mode[31:30] (iA and nF bits)**

| iA | nF | Clocking mode |
|----|----|---------------|
| 0  | 0  | Fast Bus      |
| 0  | 1  | Synchronous   |
| 1  | 0  | Reserved      |
| 1  | 1  | Asynchronous  |

**CP15 Register 2, TTB**

**Access:** Read/write

The CP15 Register 2, or Translation Table Base (TTB) Register, defines the first-level translation table.



- **Pointer[31:14]**

Points to the first-level translation table base. Read returns the currently active first-level translation table. Write sets the pointer to the first-level table to the written value.

The non-defined bits should be zero when written and are unpredictable when read.

## CP15 Register 3, Domain Access Control Register

**Access:** Read/write

The CP 15 Register 3, or Domain Access Control Register, defines the domain's access permission.

MMU accesses are priory controlled through the use of 16 domains.

Each field of Register 3 is associated with one domain.

|     |    |     |    |     |    |     |    |
|-----|----|-----|----|-----|----|-----|----|
| 31  | 30 | 29  | 28 | 27  | 26 | 25  | 24 |
| D15 |    | D14 |    | D13 |    | D12 |    |
| 23  | 22 | 21  | 20 | 19  | 18 | 17  | 16 |
| D11 |    | D10 |    | D9  |    | D8  |    |
| 15  | 14 | 13  | 12 | 11  | 10 | 9   | 8  |
| D7  |    | D6  |    | D5  |    | D4  |    |
| 7   | 6  | 5   | 4  | 3   | 2  | 1   | 0  |
| D3  |    | D2  |    | D1  |    | D0  |    |

- **D15 to D0: Named Domain Access**

The 2-bit field value allows domain access as described in the table below.

| Value |   | Access    | Description   |
|-------|---|-----------|---|
| 0     | 0 | No access | Any access generates a domain fault   |
| 0     | 1 | Client    | The Users of domain (execute programs, access data), the domain access permission controlled the domain access. |
| 1     | 0 | Reserved  | Reserved  |
| 1     | 1 | Manager   | Controls the behavior of the domain, no checking of the domain access permission is done                        |

**CP15 Register 4, Reserved**

Any access (Read or Write) to this register causes unpredictable behavior.

**CP15 Register 5, Fault Status Register**

**Access:** Read/write

Reading the CP 15 Register 5, or Fault Status Register (FSR), returns the source of the last data fault, indicating the domain and type of access being attempted when the data abort occurred.

In addition, the virtual address which caused the data abort is written into the Fault Address Register (CP15 Register 6).

Writing the CP 15 Register 5, or Fault Status Register (FSR), sets the FSR to the value of the data written. This is useful for a debugger to restore the value of the FSR.

|        |    |    |    |        |    |    |    |
|--------|----|----|----|--------|----|----|----|
| 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  |
| Domain |    |    |    | Status |    |    |    |

- **Status[3:0]: Fault Type**

Indicates the fault type. The status field is encoded by the MMU when a data abort occurs. The interpretation of the Status field is dependant on the domain field and the MVA associated with the data abort (stored in the FAR).

- **Domain[7:4]: Domain**

Indicates the domain (D15 - D0) being accessed when the fault occurred.

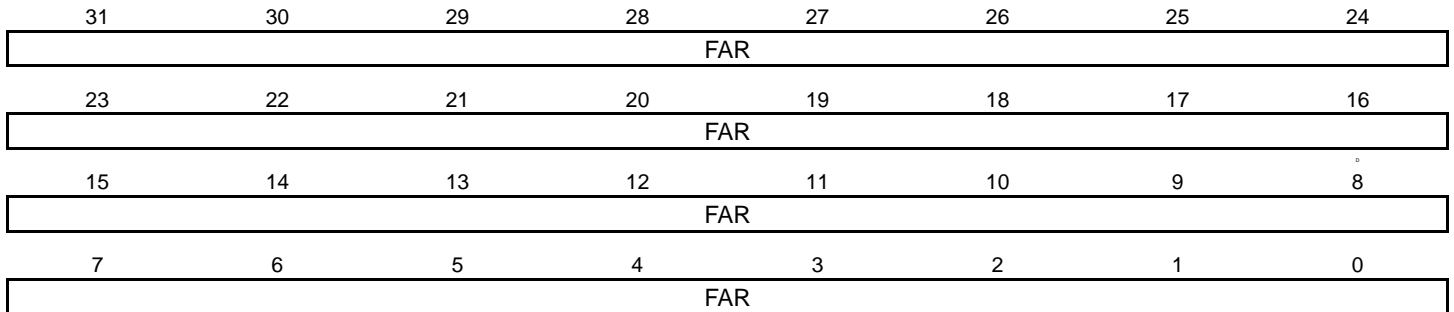
The non-defined bits should be zero when written and are unpredictable when read.

## CP15 Register 6, Fault Address Register

**Access:** Read/write

The CP 15 Register 6, or Fault Address Register (FAR), contains the MVA (Modified Virtual Address) of the access being attempted when the last fault occurred. The FAR is only updated for data faults, not for prefetch faults.

The ability to write to the FAR is provided to allow a debugger to restore a previous state.



- **FAR[31:0]: Fault Address**

On reading: returns the value of the FAR. The FAR holds the virtual address of the access which was attempted when fault occurred.

On writing: sets the FAR to the value of the written data. This is useful for a debugger to restore the value of the FAR.

## CP15 Register 7, Cache Operation Register

**Access:** Write-only

The CP15 Register 7, or Cache Operation Register, is used to manage the Instruction Cache (ICache) and the Data Cache (DCache).

The function of each cache operation is selected by the opcode\_2 and CRm fields in the MCR instruction used to write CP15 Register 7.

**Table 14.** Cache Functions

| Function  | Data         | CRm | opcode_2 |
|---|--------------|-----|----------|
| Wait for Interrupt                              | SBZ          | c0  | 4        |
| Invalidate ICache                               | SBZ          | c5  | 0        |
| Invalidate ICache single entry (using MVA)      | MVA format   | c5  | 1        |
| Invalidate DCache                               | SBZ          | c6  | 0        |
| Invalidate DCache single entry (using MVA)      | MVA format   | c6  | 1        |
| Invalidate ICache and DCache                    | SBZ          | c7  | 0        |
| Clean DCache single entry (using MVA)           | MVA format   | c10 | 1        |
| Clean DCache single entry (using index)         | Index format | c10 | 2        |
| Drain write buffer                              | SBZ          | c10 | 4        |
| Prefetch ICache line (using MVA)                | MVA format   | c13 | 1        |
| Clean and Invalidate DCache entry (using MVA)   | MVA format   | c14 | 1        |
| Clean and Invalidate DCache entry (using index) | Index format | c14 | 2        |

### Functions Details

- Wait for interrupt

Stops execution in low-power state until an interrupt occurs.

- Invalidate

The cache line (or lines) is marked as invalid, so no cache hits occur in that line until it is re-allocated to an address.

- Clean

Applies to write-back data caches. If the cache line contains stored data that has not yet been written out to the main memory, it is written to main memory immediately.

- Drain write buffer

Stops the execution until all data in the write buffer has been stored in the main memory.

- Prefetch

The memory cache line at the specified virtual address is loaded into the cache.

The operation carried out on a single cache line identifies the line using the data transferred in the MCR instruction.

The data is interpreted as using one of the two formats:

- MVA format
- index format

Below are the details of CP15 Register 7, or Cache Function Register, in MVA format.

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

• **mva[31:5]: Modified Virtual Address**

The non-defined bits should be zero when written and are unpredictable when read.

Below the details of CP15 Register 7, or Cache Function Register, in Index format:

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

• **index[31:26]: Line**

Determines the cache line.

• **set[7:5]: Segment**

Determines the cache segment.

The non-defined bits should be zero when written and are unpredictable when read.

Writing other opcode\_2 values or CRm values is unpredictable.

Reading from CP15 Register 7 is unpredictable.



### CP15 Register 8, TLB Operations Register

**Access:** Write-only

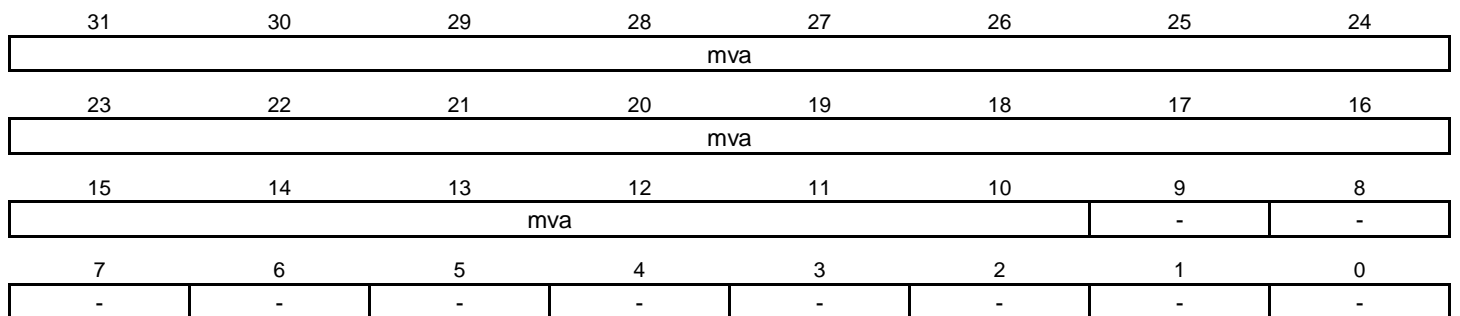
The CP15 Register 8, or Translation Lookaside Buffer (TLB) Operations Register, is used to manage instruction TLBs and data TLBs.

The TLB operation is selected by opcode\_2 and CRm fields in the MCR instruction used to write CP15 Register 8.

**Table 15.** TLB Operations

| Function                                  | Data       | CRm | opcode_2 |
|---|------------|-----|----------|
| Invalidate I TLB                          | SBZ        | 5   | 0        |
| Invalidate I TLB single entry (using MVA) | MVA format | 5   | 1        |
| Invalidate D TLB                          | SBZ        | 6   | 0        |
| Invalidate D TLB single entry (using MVA) | MVA format | 6   | 1        |
| Invalidate both Instruction and Data TLB  | SBZ        | 7   |          |

Below are details of the CP15 Register 8 for TLB operation on MVA format and one single entry.



- **mva[31:10]: Modified Virtual Address**

The non-defined bits should be zero when written and are unpredictable when read.

Writing other opcode\_2 values or CRm values is unpredictable.

Reading from CP15 Register 8 is unpredictable.

## CP15 Register 9, Cache Lockdown Register

**Access:** Read/write

The CP15 Register 9, or Cache Lockdown Register, is 0x0 on reset. The Cache Lockdown Register allows software to control which cache line in the ICache or DCache is loaded for a linefill. It prevents lines in the ICache or DCache from being evicted during a linefill, locking them into the cache.

Reading from the CP15 Register 9 returns the value of the Cache Lockdown Register that is the base pointer for all cache segments.

Only the bits[31:26] are returned; others are unpredictable.

Writing to the CP15 Register 9 updates the Cache Lockdown Register with both the base and the current victim pointers for all cache segments.

**Table 16.** Cache Lockdown Functions

| Function                              | Data          | CRm | opcode_2 |
|---------------------------------------|---------------|-----|----------|
| Read DCache lockdown base             | Base          | 0   | 0        |
| Write DCache victim and lockdown base | Victim = Base | 0   | 0        |
| Read ICache lockdown base             | Base          | 0   | 1        |
| Write ICache victim and lockdown base | Victim = Base | 0   | 1        |

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

- **index[31:26]: Victim Pointer**

Current victim pointer that specifies the cache line to be used as victim for the next linefill.

The non-defined bits should be zero when written and are unpredictable when read.

### CP15 Register 10, TLB Lockdown Register

**Access:** Read/write

The CP15 Register 10, or TLB Lockdown Register, is 0x0 on reset. There is a TLB Lockdown Register for each of the TLBs; the value of opcode\_2 determines which TLB register to access:

- opcode\_2 = 0x0 for D TLB register
- opcode\_2 = 0x1 for I TLB register

**Table 17.** TLB Lockdown Functions

| Function             | Data         | CRm | Opcode_2 |
|----------------------|--------------|-----|----------|
| Read D TLB lockdown  | TLB lockdown | 0   | 0        |
| Write D TLB lockdown | TLB lockdown | 0   | 0        |
| Read I TLB lockdown  | TLB lockdown | 0   | 1        |
| Write I TLB lockdown | TLB lockdown | 0   | 1        |

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

• **Base[31:26]: Base**

The TLB replacement strategy only uses the TLB entries numbered from base to 63. The Victim field provided is in that range.

• **Victim[25:20]: Victim Counter**

Specifies the TLB entry (line) being overwritten.

• **P[0]: Preserved**

If 0, the TLB entry can be invalidated.

If 1, the TLB entry is protected. It cannot be invalidated during the Invalidate All instruction. Refer to “CP15 Register 8, TLB Operations Register” on page 57.

The non-defined bits should be zero when written and are unpredictable when read.

### CP15 Registers 11, 12, Reserved

Any access (Read or Write) to these registers causes unpredictable behavior.

## CP15 Register 13, FCSE PID Register

**Access:** Read/write

The CP15 Register 13, or Fast Context Switch Extension (FCSE) Process Identifier (PID) Register, is set to 0x0 on reset. Reading from CP15 Register 13 returns the FCSE PID value.

Writing to CP15 Register 13 sets the FCSE PID.

The FCSE PID sets the mapping between the ARM9TDMI and the MMU of the cache memories.

The addresses issued by the ARM9TDMI are in the range of 0 to 32 Mbytes and are translated via the FCSE PID.

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

- **FCSEPID[31:25]: FCSE PID**

The FCSE PID modifies the behavior of the of the ARM920T memory system. This modification allows multiple programs to run on the ARM.

The 4-GB virtual address is divided into 128 process blocks of 32 Mbytes each. Each process block can contain a program that has been compiled to use the address range 0x00000000 to 0x01FFFFFF. For each  $i = 0$  to 127 process blocks,  $i$  runs from address  $i * 0x20000000$  to address  $i * 0x20000000 + 0x01FFFFFF$ .

For further details, see the ARM920T Technical Reference Manual, Rev. DDI0151C.

The non-defined bits should be zero when written and are unpredictable when read.

## CP15 Register 14, Reserved

Any access (Read or Write) of these registers causes unpredictable behavior.

## CP15 Register 15, Test Configuration Register

CP15 Register 15, or Test Configuration Register, is used for test purposed. Any access (write or read) to this register causes unpredictable behavior.

## Debug and Test Features (DBG Test)

### Overview

The AT91RM9200 features a number of complementary debug and test capabilities. A common JTAG/ICE (In-Circuit Emulator) port is used for standard debugging functions such as downloading code and single-stepping through programs. An ETM (Embedded Trace Macrocell) provides more sophisticated debug features such as address and data comparators, half-rate clock mode, counters, sequencer and FIFO. The Debug Unit provides a two-pin UART that can be used to upload an application into internal SRAM. It manages the interrupt handling of the internal COMMTX and COMMRX signals that trace the activity of the Debug Communication Channel.

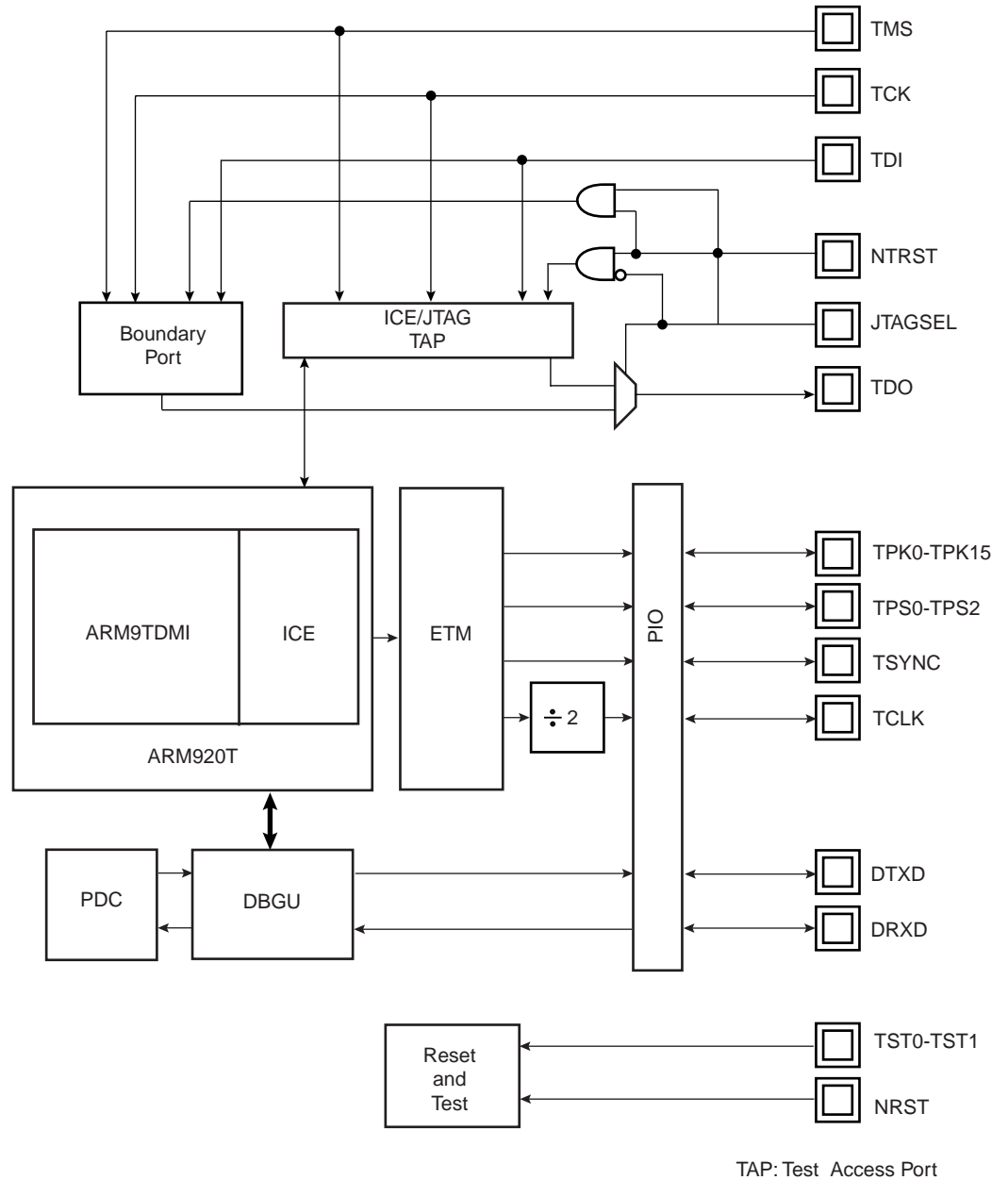
A set of dedicated debug and test input/output pins give direct access to these capabilities from a PC-based test environment.

Features of Debug and Test Features are:

- Integrated Embedded In-Circuit-Emulator
- Debug Unit
  - Two-pin UART
  - Debug Communication Channel
  - Chip ID Register
- Embedded Trace Macrocell: ETM9 Rev2a
  - Medium Level Implementation
  - Half-rate Clock Mode
  - Four Pairs of Address Comparators
  - Two Data Comparators
  - Eight Memory Map Decoder Inputs
  - Two Counters
  - One Sequencer
  - One 18-byte FIFO
- IEEE1149.1 JTAG Boundary Scan on all Digital Pins

# Block Diagram

Figure 9. AT91RM9200 Debug and Test Block Diagram

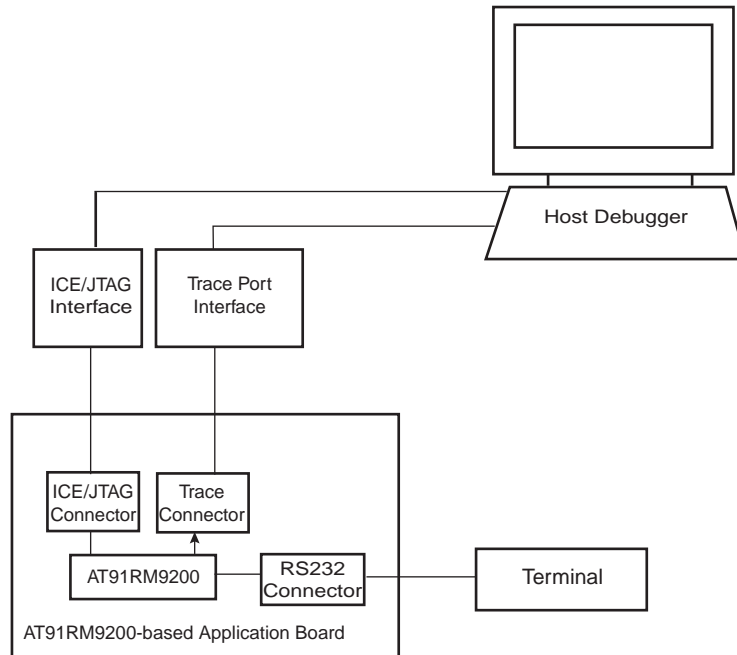


## Application Examples

### Debug Environment

Figure 10 on page 63 shows a complete debug environment example. The ICE/JTAG interface is used for standard debugging functions such as downloading code and single-stepping through the program. The Trace Port interface is used for tracing information. A software debugger running on a personal computer provides the user interface for configuring a Trace Port interface utilizing the ICE/JTAG interface.

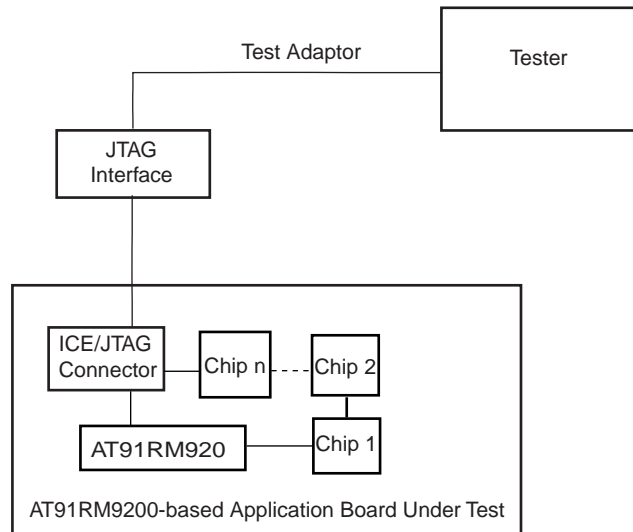
**Figure 10.** AT91RM9200-based Application Debug and Trace Environment Example



### Test Environment

Figure 11 below shows a test environment example. Test vectors are sent and interpreted by the tester. In this example, the “board under test” is designed using many JTAG compliant devices. These devices can be connected together to form a single scan chain.

**Figure 11.** AT91RM9200-based Application IEEE1149.1 Test Environment Example



## Debug and Test Pin Description

**Table 18.** Debug and Test Pin List

| Pin Name                                   | Function                     | Type   | Active Level |
|--|------------------------------|--------|--------------|
| <b>Reset/Test</b>                          |                              |        |              |
| NRST                                       | Microcontroller Reset        | Input  | Low          |
| TST0                                       | Test Mode Select             | Input  |              |
| TST1                                       | Test Mode Select             | Input  |              |
| <b>ICE and JTAG</b>                        |                              |        |              |
| TCK  | Test Clock                   | Input  |              |
| TDI  | Test Data In                 | Input  |              |
| TDO  | Test Data Out                | Output |              |
| TMS  | Test Mode Select             | Input  |              |
| NTRST                                      | Test Reset Signal            | Input  | Low          |
| JTAGSEL                                    | JTAG Selection               | Input  |              |
| <b>ETM (available only in BGA package)</b> |                              |        |              |
| TSYNC                                      | Trace Synchronization Signal | Output |              |
| TCLK                                       | Trace Clock                  | Output |              |
| TPS0- TPS2                                 | Trace ARM Pipeline Status    | Output |              |
| TPK0 - TPK15                               | Trace Packet Port            | Output |              |
| <b>Debug Unit</b>                          |                              |        |              |
| DRXD                                       | Debug Receive Data           | Input  | DRXD         |
| DTXD                                       | Debug Transmit Data          | Output | DTXD         |



## Functional Description

### Test Mode Pins

Two dedicated pins (TST1, TST0) are used to define the test mode of the device. The user must make sure that these pins are both tied at low level to ensure normal operating conditions. Other values associated to these pins are manufacturing test reserved.

### Embedded In-Circuit Emulator

The ARM9TDMI Embedded In-Circuit Emulator is supported via the ICE/JTAG port. It is connected to a host computer via an ICE interface. Debug support is implemented using an ARM9TDMI core embedded within the ARM920T. The internal state of the ARM920T is examined through an ICE/JTAG port which allows instructions to be serially inserted into the pipeline of the core without using the external data bus. Therefore, when in debug state, a store-multiple (STM) can be inserted into the instruction pipeline. This exports the contents of the ARM9TDMI registers. This data can be serially shifted out without affecting the rest of the system.

There are six scan chains inside the ARM920T processor which support testing, debugging, and programming of the Embedded ICE. The scan chains are controlled by the ICE/JTAG port.

Embedded ICE mode is selected when JTAGSEL is low. It is not possible to switch directly between ICE and JTAG operations. A chip reset must be performed (NRST and NTRST) after JTAGSEL is changed. The test reset input to the embedded ICE (NTRST) is provided separately to facilitate debug of the boot program.

For further details on the Embedded In-Circuit-Emulator, see the ARM920T Technical Reference Manual, ARM Ltd, - DDI 0151C.

### Debug Unit

The Debug Unit provides a two-pin (DXRD and TXRD) UART that can be used for several debug and trace purposes and offers an ideal means for in-situ programming solutions and debug monitor communication. Moreover, the link with two peripheral data controller channels provides packet handling of these tasks with processor time reduced to a minimum.

The Debug Unit also manages the interrupt handling of the COMMTX and COMMRX signals that come from the ICE and trace the activity of the Debug Communication Channel. The Debug Unit allows blockage of access to the system through the ICE interface.

The Debug Unit can be used to upload an application into internal SRAM. It is activated by the boot program when no valid application is detected. The protocol used to load the application is XMODEM.

A specific register, the Debug Unit Chip ID Register, informs about the product version and its internal configuration.

AT91RM9200 Debug Unit Chip ID value is: 0x09290781, on 32-bit width.

For further details on the Debug Unit, see the Debug Unit datasheet; Atmel literature number, 2641.

For further details on the Debug Unit and Boot program, see the Boot Program Specifications.

### Embedded Trace Macrocell

The AT91RM9200 features an Embedded Trace Macrocell (ETM), which is closely connected to the ARM9TDMI Processor. The Embedded Trace is a standard mid-level implementation and contains the following resources:

- Four pairs of address comparators
- Two data comparators



- Eight memory map decoder inputs
- Two counters
- One sequencer
- Four external inputs
- One external output
- One 18-byte FIFO

The Embedded Trace Macrocell of the AT91RM9200 works in half-rate clock mode and thus integrates a clock divider. This assures that the maximum frequency of all the trace port signals do not exceed one half of the ARM920T clock speed.

The Embedded Trace Macrocell input and output resources are not used in the AT91RM9200.

The Embedded Trace is a real-time trace module with the capability of tracing the ARM9TDMI instruction and data.

The Embedded Trace debug features are only accessible in the AT91RM9200 BGA package.

For further details on Embedded Trace Macrocell, see the ETM9 (Rev2a) Technical Reference Manual, ARM Ltd. -DDI 0157E.

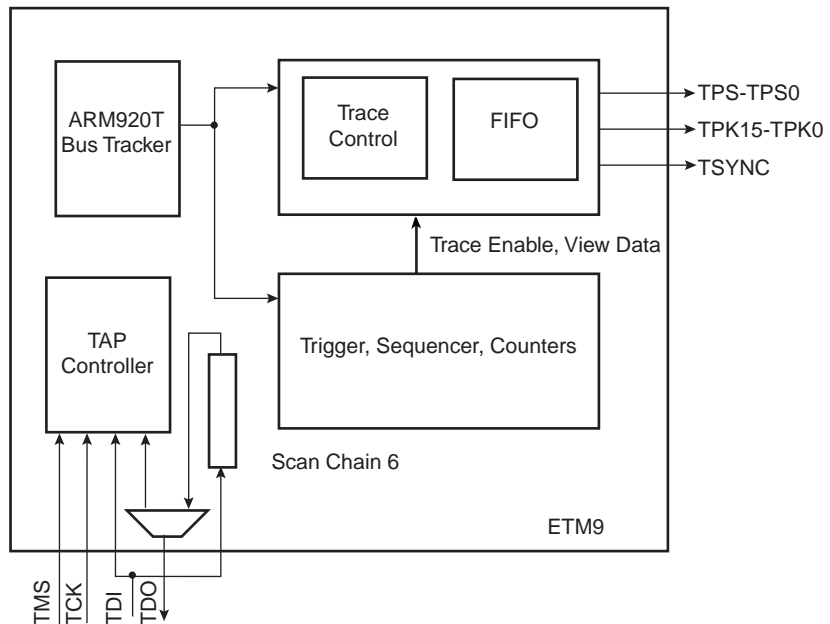
## Trace Port

The Trace Port is made up of the following pins:

- TSYNC - the synchronization signal (Indicates the start of a branch sequence on the trace packet port.)
- TCLK - the Trace Port clock, half-rate of the ARM920T processor clock.
- TPS0 to TPS2 - indicate the processor state at each trace clock edge.
- TPK0 to TPK15 - the Trace Packet data value.

The trace packet information (address, data) is associated with the processor state indicated by TPS. Some processor states have no additional data associated with the Trace Packet Port (i.e. failed condition code of an instruction). The packet is 8-bits wide, and up to two packets can be output per cycle.

**Figure 12.** ETM9 Block



## Implementation Details

This section gives an overview of the Embedded Trace resources. For further details, see the Embedded Trace Macrocell Specification, ARM Ltd. -IHI 0014H.

### Three-state Sequencer

The sequencer has three possible next states (one dedicated to itself and two others) and can change on every clock cycle. The state transition is controlled with internal events. If the user needs multiple-stage trigger schemes, the trigger event is based on a sequencer state.

### Address Comparator

In single mode, address comparators compare either the instruction address or the data address against the user-programmed address.

In range mode, the address comparators are arranged in pairs to form a virtual address range resource.

Details of the address comparator programming are:

- The first comparator is programmed with the range start address.
- The second comparator is programmed with the range end address.
- The resource matches if the address is within the following range:
  - (address >= range start address) AND (address < range end address)
- Unpredictable behavior occurs if the two address comparators are not configured in the same way.

### Data Comparator

Each full address comparator is associated with a specific data comparator. A data comparator is used to observe the data bus only when load and store operations occur.

A data comparator has both a value register and a mask register, therefore it is possible to compare only certain bits of a preprogrammed value against the data bus.

### Memory Decoder Inputs

The eight memory map decoder inputs are connected to custom address decoders. The address decoders divide the memory into regions of on-chip SRAM, on-chip ROM, and peripherals. The address decoders also optimize the ETM9 trace trigger.

**Table 19.** ETM Memory Map Inputs Layout

| Description       | Region   | Access type | start_address | end_address  |
|-------------------|----------|-------------|---------------|--------------|
| SRAM              | Internal | Data        | 0x00000000    | 0x000FFFFFFF |
| SRAM              | Internal | Fetch       | 0x00000000    | 0x000FFFFFFF |
| ROM               | Internal | Data        | 0x00100000    | 0x001FFFFFFF |
| ROM               | Internal | Fetch       | 0x00100000    | 0x001FFFFFFF |
| NCS0-NCS7         | External | Data        | 0x10000000    | 0x8FFFFFFF   |
| NCS0-NCS7         | External | Fetch       | 0x10000000    | 0x8FFFFFFF   |
| User Peripheral   | Internal | Data        | 0xF0000000    | 0xFFFFEFFF   |
| System Peripheral | Internal | Data        | 0xFFFFF000    | 0xFFFFFFF    |

### FIFO

An 18-byte FIFO is used to store data tracing. The FIFO is used to separate the pipeline status from the trace packet. So, the FIFO can be used to buffer trace packets.

A FIFO overflow is detected by the embedded trace macrocell when the FIFO is full or when the FIFO has less bytes than the user-programmed number.

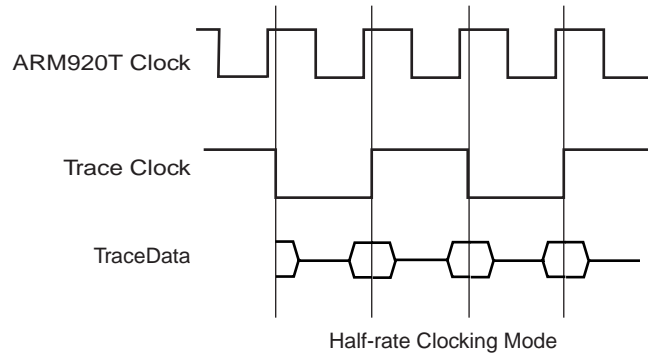
For further details, see the ETM9 (Rev2a) Technical Reference Manual, ARM Ltd. DDI 0157E.

*Half-rate Clocking Mode*

The ETM9 is implemented in half-rate mode that allows both rising and falling edge data tracing of the trace clock.

The half-rate mode is implemented to maintain the signal clock integrity of high speed systems (up to 100 Mhz).

**Figure 13.** Half-rate Clocking Mode



Care must be taken on the choice of the trace capture system as it needs to support half-rate clock functionality.

**Application Board Restriction**

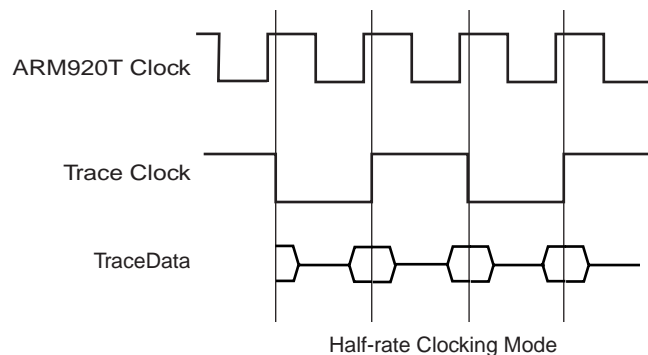
The TCLK signal needs to be set with care, some timing parameters are required.

Refer to AT91RM9200 “JTAG/ICE Timings” on page 621 and “ETM Timings” on page 624.

The specified target system connector is the AMP Mictor connector.

The connector must be oriented on the application board as described below in Figure 14. The view of the PCB is shown from above with the trace connector mounted near the edge of the board. This allows the Trace Port Analyzer to minimize the physical intrusiveness of the inter-connected target.

**Figure 14.** AMP Mictor Connector Orientation



## IEEE 1149.1 JTAG Boundary Scan

IEEE 1149.1 JTAG Boundary Scan allows pin-level access independent of the device packaging technology.

IEEE 1149.1 JTAG Boundary Scan is enabled when JTAGSEL is high. The SAMPLE, EXTEST and BYPASS functions are implemented. In ICE debug mode, the ARM processor responds with a non-JTAG chip ID that identifies the processor 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 JTAGSEL is changed.

Two Boundary Scan Descriptor Language (BSDL) files are provided to set up testing. Each BSDL file is dedicated to a specific packaging.

## JTAG Boundary Scan Register

The Boundary Scan Register (BSR) contains 449 bits which correspond to active pins and associated control signals.

Each AT91RM9200 input pin has a corresponding bit in the Boundary Scan Register for observability.

Each AT91RM9200 output pin has a corresponding 2-bit register in the BSR. The OUTPUT bit contains data which can be forced on the pad. The CTRL bit can put the pad into high impedance.

Each AT91RM9200 input/output 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 facilitates the observability of data applied to the pad. The CTRL bit selects the direction of the pad.

**Table 20.** JTAG Boundary Scan Register

| Bit Number | Pin Name                                     | Pin Type | Associated BSR Cells |
|------------|--|----------|----------------------|
| 449        | A19  | Output   | OUTPUT               |
| 448        | A[19:16]/BA0/BA1                             | Output   | CTRL                 |
| 447        | A20  | Output   | OUTPUT               |
| 446        | A[22:20]/NWE/NWR0                            | Output   | CTRL                 |
| 445        | A21  | Output   | OUTPUT               |
| 444        | A22  | Output   | OUTPUT               |
| 443        | PC7/A23                                      | I/O      | INPUT                |
| 442        |  |          | OUTPUT               |
| 441        |  |          | CTRL                 |
| 440        | PC8/A24                                      | I/O      | INPUT                |
| 439        |  |          | OUTPUT               |
| 438        |  |          | CTRL                 |
| 437        | PC9/A25/CFRNW                                | I/O      | INPUT                |
| 436        |  |          | OUTPUT               |
| 435        |  |          | CTRL                 |
| 434        | NCS0/BFCS                                    | Output   | OUTPUT               |
| 433        | NCS[1:0]/NOE/NRD/NUB/<br>NWR1/NBS1/BFCS/SDCS | Output   | CTRL                 |

**Table 20. JTAG Boundary Scan Register (Continued)**

| Bit Number | Pin Name               | Pin Type | Associated BSR Cells |
|------------|------------------------|----------|----------------------|
| 432        | NCS1/SDCS              | Output   | OUTPUT               |
| 431        | NCS2                   | Output   | OUTPUT               |
| 430        | NCS[2:3]/NBS3          | Output   | CTRL                 |
| 429        | NCS3                   | Output   | OUTPUT               |
| 428        | NOE/NRD                | Output   | OUTPUT               |
| 427        | NWE/NWR0               | Output   | INPUT                |
| 426        |                        |          | OUTPUT               |
| 425        | NUB/NWR1/NBS1          | Output   | INPUT                |
| 424        |                        |          | OUTPUT               |
| 423        | NBS3                   | Output   | OUTPUT               |
| 422        | SDCKE                  | Output   | OUTPUT               |
| 421        | SDCKE/RAS/CAS/WE/SDA10 | Output   | CTRL                 |
| 420        | RAS                    | Output   | OUTPUT               |
| 419        | CAS                    | Output   | OUTPUT               |
| 418        | WE                     | Output   | OUTPUT               |
| 417        | D0                     | I/O      | INPUT                |
| 416        |                        |          | OUTPUT               |
| 415        | D[3:0]                 | I/O      | CTRL                 |
| 414        | D1                     | I/O      | INPUT                |
| 413        |                        |          | OUTPUT               |
| 412        | D2                     | I/O      | INPUT                |
| 411        |                        |          | OUTPUT               |
| 410        | D3                     | I/O      | INPUT                |
| 409        |                        |          | OUTPUT               |
| 408        | D4                     | I/O      | INPUT                |
| 407        |                        |          | OUTPUT               |
| 406        | D[7:4]                 | I/O      | CTRL                 |
| 405        | D5                     | I/O      | INPUT                |
| 404        |                        |          | OUTPUT               |
| 403        | D6                     | I/O      | INPUT                |
| 402        |                        |          | OUTPUT               |
| 401        | D7                     | I/O      | INPUT                |
| 400        |                        |          | OUTPUT               |
| 399        | D8                     | I/O      | INPUT                |
| 398        |                        |          | OUTPUT               |

**Table 20.** JTAG Boundary Scan Register (Continued)

| Bit Number | Pin Name | Pin Type | Associated BSR Cells |
|------------|----------|----------|----------------------|
| 397        | D[11:8]  | I/O      | CTRL                 |
| 396        | D9       | I/O      | INPUT                |
| 395        |          |          | OUTPUT               |
| 394        | D10      | I/O      | INPUT                |
| 393        |          |          | OUTPUT               |
| 392        | D11      | I/O      | INPUT                |
| 391        |          |          | OUTPUT               |
| 390        | D12      | I/O      | INPUT                |
| 389        |          |          | OUTPUT               |
| 388        | D[15:12] | I/O      | CTRL                 |
| 387        | D13      | I/O      | INPUT                |
| 386        |          |          | OUTPUT               |
| 385        | D14      | I/O      | INPUT                |
| 384        |          |          | OUTPUT               |
| 383        | D15      | I/O      | INPUT                |
| 382        |          |          | OUTPUT               |
| 381        | PC16/D16 | I/O      | INPUT                |
| 380        |          |          | OUTPUT               |
| 379        |          |          | CTRL                 |
| 378        | PC17/D17 | I/O      | INPUT                |
| 377        |          |          | OUTPUT               |
| 376        |          |          | CTRL                 |
| 375        | PC18/D18 | I/O      | INPUT                |
| 374        |          |          | OUTPUT               |
| 373        |          |          | CTRL                 |
| 372        | PC19/D19 | I/O      | INPUT                |
| 371        |          |          | OUTPUT               |
| 370        |          |          | CTRL                 |
| 369        | PC20/D20 | I/O      | INPUT                |
| 368        |          |          | OUTPUT               |
| 367        |          |          | CTRL                 |
| 366        | PC21/D21 | I/O      | INPUT                |
| 365        |          |          | OUTPUT               |
| 364        |          |          | CTRL                 |

**Table 20.** JTAG Boundary Scan Register (Continued)

| Bit Number | Pin Name       | Pin Type | Associated BSR Cells |
|------------|----------------|----------|----------------------|
| 363        | PC22/D22       | I/O      | INPUT                |
| 362        |                |          | OUTPUT               |
| 361        |                |          | CTRL                 |
| 360        | PC23/D23       | I/O      | INPUT                |
| 359        |                |          | OUTPUT               |
| 358        |                |          | CTRL                 |
| 357        | PC24/D24       | I/O      | INPUT                |
| 356        |                |          | OUTPUT               |
| 355        |                |          | CTRL                 |
| 354        | PC25/D25       | I/O      | INPUT                |
| 353        |                |          | OUTPUT               |
| 352        |                |          | CTRL                 |
| 351        | PC26/D26       | I/O      | INPUT                |
| 350        |                |          | OUTPUT               |
| 349        |                |          | CTRL                 |
| 348        | PC27/D27       | I/O      | INPUT                |
| 347        |                |          | OUTPUT               |
| 346        |                |          | CTRL                 |
| 345        | PC28/D28       | I/O      | INPUT                |
| 344        |                |          | OUTPUT               |
| 343        |                |          | CTRL                 |
| 342        | PC29/D29       | I/O      | INPUT                |
| 341        |                |          | OUTPUT               |
| 340        |                |          | CTRL                 |
| 339        | PC30/D30       | I/O      | INPUT                |
| 338        |                |          | OUTPUT               |
| 337        |                |          | CTRL                 |
| 336        | PC31/D31       | I/O      | INPUT                |
| 335        |                |          | OUTPUT               |
| 334        |                |          | CTRL                 |
| 333        | PC10/NCS4/CFCS | I/O      | INPUT                |
| 332        |                |          | OUTPUT               |
| 331        |                |          | CTRL                 |



**Table 20.** JTAG Boundary Scan Register (Continued)

| Bit Number | Pin Name        | Pin Type | Associated BSR Cells |
|------------|-----------------|----------|----------------------|
| 330        | PC11/NCS5/CFCE1 | I/O      | INPUT                |
| 329        |                 |          | OUTPUT               |
| 328        |                 |          | CTRL                 |
| 327        | PC12/NCS6/CFCE2 | I/O      | INPUT                |
| 326        |                 |          | OUTPUT               |
| 325        |                 |          | CTRL                 |
| 324        | PC13/NCS7       | I/O      | INPUT                |
| 323        |                 |          | OUTPUT               |
| 322        |                 |          | CTRL                 |
| 321        | PC14            | I/O      | INPUT                |
| 320        |                 |          | OUTPUT               |
| 319        |                 |          | CTRL                 |
| 318        | PC15            | I/O      | INPUT                |
| 317        |                 |          | OUTPUT               |
| 316        |                 |          | CTRL                 |
| 315        | PC0/BCFK        | I/O      | INPUT                |
| 314        |                 |          | OUTPUT               |
| 313        |                 |          | CTRL                 |
| 312        | PC1/BFRDY/SMOE  | I/O      | INPUT                |
| 311        |                 |          | OUTPUT               |
| 310        |                 |          | CTRL                 |
| 309        | PC2/BFAVD       | I/O      | INPUT                |
| 308        |                 |          | OUTPUT               |
| 307        |                 |          | CTRL                 |
| 306        | PC3/BFBAA/SMWE  | I/O      | INPUT                |
| 305        |                 |          | OUTPUT               |
| 304        |                 |          | CTRL                 |
| 303        | PC4/BFOE        | I/O      | INPUT                |
| 302        |                 |          | OUTPUT               |
| 301        |                 |          | CTRL                 |
| 300        | PC5/BFWE        | I/O      | INPUT                |
| 299        |                 |          | OUTPUT               |
| 298        |                 |          | CTRL                 |

**Table 20.** JTAG Boundary Scan Register (Continued)

| Bit Number | Pin Name       | Pin Type | Associated BSR Cells |
|------------|----------------|----------|----------------------|
| 297        | PC6/NWAIT      | I/O      | INPUT                |
| 296        |                |          | OUTPUT               |
| 295        |                |          | CTRL                 |
| 294        | PA0/MISO/PCK3  | I/O      | INPUT                |
| 293        |                |          | OUTPUT               |
| 292        |                |          | CTRL                 |
| 291        | PA1/MOSI/PCK0  | I/O      | INPUT                |
| 290        |                |          | OUTPUT               |
| 289        |                |          | CTRL                 |
| 288        | PA2/SPCK/IRQ4  | I/O      | INPUT                |
| 287        |                |          | OUTPUT               |
| 286        |                |          | CTRL                 |
| 285        | PA3/NPCS0/IRQ5 | I/O      | INPUT                |
| 284        |                |          | OUTPUT               |
| 283        |                |          | CTRL                 |
| 282        | PA4/NPCS1/PCK1 | I/O      | INPUT                |
| 281        |                |          | OUTPUT               |
| 280        |                |          | CTRL                 |
| 279        | PA5/NPCS2/TXD3 | I/O      | INPUT                |
| 278        |                |          | OUTPUT               |
| 277        |                |          | CTRL                 |
| 276        | PD0/ETX0       | I/O      | INPUT                |
| 275        |                |          | OUTPUT               |
| 274        |                |          | CTRL                 |
| 273        | PD1/ETX1       | I/O      | INPUT                |
| 272        |                |          | OUTPUT               |
| 271        |                |          | CTRL                 |
| 270        | PD2/ETX2       | I/O      | INPUT                |
| 269        |                |          | OUTPUT               |
| 268        |                |          | CTRL                 |
| 267        | PD3/ETX3       | I/O      | INPUT                |
| 266        |                |          | OUTPUT               |
| 265        |                |          | CTRL                 |

**Table 20.** JTAG Boundary Scan Register (Continued)

| Bit Number | Pin Name              | Pin Type | Associated BSR Cells |
|------------|-----------------------|----------|----------------------|
| 264        | PD4/ETXEN             | I/O      | INPUT                |
| 263        |                       |          | OUTPUT               |
| 262        |                       |          | CTRL                 |
| 261        | PD5/ETXER             | I/O      | INPUT                |
| 260        |                       |          | OUTPUT               |
| 259        |                       |          | CTRL                 |
| 258        | PD6/DTXD              | I/O      | INPUT                |
| 257        |                       |          | OUTPUT               |
| 256        |                       |          | CTRL                 |
| 255        | PA6/NPCS3/RXD3        | I/O      | INPUT                |
| 254        |                       |          | OUTPUT               |
| 253        |                       |          | CTRL                 |
| 252        | PA7/ETXCK/EREFCK/PCK2 | I/O      | INPUT                |
| 251        |                       |          | OUTPUT               |
| 250        |                       |          | CTRL                 |
| 249        | PA8/ETXEN/MCCDB       | I/O      | INPUT                |
| 248        |                       |          | OUTPUT               |
| 247        |                       |          | CTRL                 |
| 246        | PA9/ETX0/MCDB0        | I/O      | INPUT                |
| 245        |                       |          | OUTPUT               |
| 244        |                       |          | CTRL                 |
| 243        | PA10/ETX1/MCDB1       | I/O      | INPUT                |
| 242        |                       |          | OUTPUT               |
| 241        |                       |          | CTRL                 |
| 240        | PA11/ECRS/ECRSV/MCDB2 | I/O      | INPUT                |
| 239        |                       |          | OUTPUT               |
| 238        |                       |          | CTRL                 |
| 237        | PA12/ERX0/MCDB3       | I/O      | INPUT                |
| 236        |                       |          | OUTPUT               |
| 235        |                       |          | CTRL                 |
| 234        | PA13/ERX1/TCLK0       | I/O      | INPUT                |
| 233        |                       |          | OUTPUT               |
| 232        |                       |          | CTRL                 |

**Table 20.** JTAG Boundary Scan Register (Continued)

| Bit Number | Pin Name         | Pin Type | Associated BSR Cells |
|------------|------------------|----------|----------------------|
| 231        | PA14/ERXER/TCLK1 | I/O      | INPUT                |
| 230        |                  |          | OUTPUT               |
| 229        |                  |          | CTRL                 |
| 228        | PA15/EMDC/TCLK2  | I/O      | INPUT                |
| 227        |                  |          | OUTPUT               |
| 226        |                  |          | CTRL                 |
| 225        | PA16/EMDIO/IRQ6  | I/O      | INPUT                |
| 224        |                  |          | OUTPUT               |
| 223        |                  |          | CTRL                 |
| 222        | PA17/TXD0/TIOA0  | I/O      | INPUT                |
| 221        |                  |          | OUTPUT               |
| 220        |                  |          | CTRL                 |
| 219        | PA18/RXD0/TIOB0  | I/O      | INPUT                |
| 218        |                  |          | OUTPUT               |
| 217        |                  |          | CTRL                 |
| 216        | PA19/SCK0/TIOA1  | I/O      | INPUT                |
| 215        |                  |          | OUTPUT               |
| 214        |                  |          | CTRL                 |
| 213        | PA20/CTS0/TIOB1  | I/O      | INPUT                |
| 212        |                  |          | OUTPUT               |
| 211        |                  |          | CTRL                 |
| 210        | PA21/RTS0/TIOA2  | I/O      | INPUT                |
| 209        |                  |          | OUTPUT               |
| 208        |                  |          | CTRL                 |
| 207        | PA22/RXD2/TIOB2  | I/O      | INPUT                |
| 206        |                  |          | OUTPUT               |
| 205        |                  |          | CTRL                 |
| 204        | PA23/TXD2/IRQ3   | I/O      | INPUT                |
| 203        |                  |          | OUTPUT               |
| 202        |                  |          | CTRL                 |
| 201        | PA24/SCK2/PCK1   | I/O      | INPUT                |
| 200        |                  |          | OUTPUT               |
| 199        |                  |          | CTRL                 |

**Table 20.** JTAG Boundary Scan Register (Continued)

| Bit Number | Pin Name         | Pin Type | Associated BSR Cells |
|------------|------------------|----------|----------------------|
| 198        | PA25/TWD/IRQ2    | I/O      | INPUT                |
| 197        |                  |          | OUTPUT               |
| 196        |                  |          | CTRL                 |
| 195        | PA26/TWCK/IRQ1   | I/O      | INPUT                |
| 194        |                  |          | OUTPUT               |
| 193        |                  |          | CTRL                 |
| 192        | PA27/MCCK/TCLK3  | I/O      | INPUT                |
| 191        |                  |          | OUTPUT               |
| 190        |                  |          | CTRL                 |
| 189        | PA28/MCCDA/TCLK4 | I/O      | INPUT                |
| 188        |                  |          | OUTPUT               |
| 187        |                  |          | CTRL                 |
| 186        | PA29/MCDA0/TCLK5 | I/O      | INPUT                |
| 185        |                  |          | OUTPUT               |
| 184        |                  |          | CTRL                 |
| 183        | PA30/DRXD/CTS2   | I/O      | INPUT                |
| 182        |                  |          | OUTPUT               |
| 181        |                  |          | CTRL                 |
| 180        | PA31/DTXD/RTS2   | I/O      | INPUT                |
| 179        |                  |          | OUTPUT               |
| 178        |                  |          | CTRL                 |
| 177        | PB0/TF0/RTS3     | I/O      | INPUT                |
| 176        |                  |          | OUTPUT               |
| 175        |                  |          | CTRL                 |
| 174        | PB1/TK0/CTS3     | I/O      | INPUT                |
| 173        |                  |          | OUTPUT               |
| 172        |                  |          | CTRL                 |
| 171        | PB2/TD0/SCK3     | I/O      | INPUT                |
| 170        |                  |          | OUTPUT               |
| 169        |                  |          | CTRL                 |
| 168        | PB3/RD0/MCDA1    | I/O      | INPUT                |
| 167        |                  |          | OUTPUT               |
| 166        |                  |          | CTRL                 |

**Table 20.** JTAG Boundary Scan Register (Continued)

| Bit Number | Pin Name       | Pin Type | Associated BSR Cells |
|------------|----------------|----------|----------------------|
| 165        | PB4/RK0/MCDA2  | I/O      | INPUT                |
| 164        |                |          | OUTPUT               |
| 163        |                |          | CTRL                 |
| 162        | PB5/RF0/MCDA3  | I/O      | INPUT                |
| 161        |                |          | OUTPUT               |
| 160        |                |          | CTRL                 |
| 159        | PB6/TF1/TIOA3  | I/O      | INPUT                |
| 158        |                |          | OUTPUT               |
| 157        |                |          | CTRL                 |
| 156        | PB7/TK1/TIOB3  | I/O      | INPUT                |
| 155        |                |          | OUTPUT               |
| 154        |                |          | CTRL                 |
| 153        | PB8/TD1/TIOA4  | I/O      | INPUT                |
| 152        |                |          | OUTPUT               |
| 151        |                |          | CTRL                 |
| 150        | PB9/RD1/TIOB4  | I/O      | INPUT                |
| 149        |                |          | OUTPUT               |
| 148        |                |          | CTRL                 |
| 147        | PB10/RK1/TIOA5 | I/O      | INPUT                |
| 146        |                |          | OUTPUT               |
| 145        |                |          | CTRL                 |
| 144        | PB11/RF1/TIOB5 | I/O      | INPUT                |
| 143        |                |          | OUTPUT               |
| 142        |                |          | CTRL                 |
| 141        | PB12/TF2/ETX2  | I/O      | INPUT                |
| 140        |                |          | OUTPUT               |
| 139        |                |          | CTRL                 |
| 138        | PB13/TK2/ETX3  | I/O      | INPUT                |
| 137        |                |          | OUTPUT               |
| 136        |                |          | CTRL                 |
| 135        | PB14/TD2/ETXER | I/O      | INPUT                |
| 134        |                |          | OUTPUT               |
| 133        |                |          | CTRL                 |

**Table 20.** JTAG Boundary Scan Register (Continued)

| Bit Number | Pin Name        | Pin Type | Associated BSR Cells |
|------------|-----------------|----------|----------------------|
| 132        | PB15/RD2/ERX2   | I/O      | INPUT                |
| 131        |                 |          | OUTPUT               |
| 130        |                 |          | CTRL                 |
| 129        | PB16/RK2/ERX3   | I/O      | INPUT                |
| 128        |                 |          | OUTPUT               |
| 127        |                 |          | CTRL                 |
| 126        | PD7/PCK0/TSYNC  | I/O      | INPUT                |
| 125        |                 |          | OUTPUT               |
| 124        |                 |          | CTRL                 |
| 123        | PD8/PCK1/TCLK   | I/O      | INPUT                |
| 122        |                 |          | OUTPUT               |
| 121        |                 |          | CTRL                 |
| 120        | PD9/PCK2/TPS0   | I/O      | INPUT                |
| 119        |                 |          | OUTPUT               |
| 118        |                 |          | CTRL                 |
| 117        | PD10/PCK3/TPS1  | I/O      | INPUT                |
| 116        |                 |          | OUTPUT               |
| 115        |                 |          | CTRL                 |
| 114        | PD11/TPS2       | I/O      | INPUT                |
| 113        |                 |          | OUTPUT               |
| 112        |                 |          | CTRL                 |
| 111        | PD12/TPK0       | I/O      | INPUT                |
| 110        |                 |          | OUTPUT               |
| 109        |                 |          | CTRL                 |
| 108        | PB17/RF2/ERXDV  | I/O      | INPUT                |
| 107        |                 |          | OUTPUT               |
| 106        |                 |          | CTRL                 |
| 105        | PB18/RI1/ECOL   | I/O      | INPUT                |
| 104        |                 |          | OUTPUT               |
| 103        |                 |          | CTRL                 |
| 102        | PB19/DTR1/ERXCK | I/O      | INPUT                |
| 101        |                 |          | OUTPUT               |
| 100        |                 |          | CTRL                 |

**Table 20. JTAG Boundary Scan Register (Continued)**

| Bit Number | Pin Name        | Pin Type | Associated BSR Cells |
|------------|-----------------|----------|----------------------|
| 99         | PB20/TXD1       | I/O      | INPUT                |
| 98         |                 |          | OUTPUT               |
| 97         |                 |          | CTRL                 |
| 96         | PB21/RXD1       | I/O      | INPUT                |
| 95         |                 |          | OUTPUT               |
| 94         |                 |          | CTRL                 |
| 93         | PB22/SCK1       | I/O      | INPUT                |
| 92         |                 |          | OUTPUT               |
| 91         |                 |          | CTRL                 |
| 90         | PD13/TPK1       | I/O      | INPUT                |
| 89         |                 |          | OUTPUT               |
| 88         |                 |          | CTRL                 |
| 87         | PD14/TPK2       | I/O      | INPUT                |
| 86         |                 |          | OUTPUT               |
| 85         |                 |          | CTRL                 |
| 84         | PD15/TD0/TPK3   | I/O      | INPUT                |
| 83         |                 |          | OUTPUT               |
| 82         |                 |          | CTRL                 |
| 81         | PB23/DCD1       | I/O      | INPUT                |
| 80         |                 |          | OUTPUT               |
| 79         |                 |          | CTRL                 |
| 78         | PB24/CTS1       | I/O      | INPUT                |
| 77         |                 |          | OUTPUT               |
| 76         |                 |          | CTRL                 |
| 75         | PB25/DSR1/EF100 | I/O      | INPUT                |
| 74         |                 |          | OUTPUT               |
| 73         |                 |          | CTRL                 |
| 72         | PB26/RTS1       | I/O      | INPUT                |
| 71         |                 |          | OUTPUT               |
| 70         |                 |          | CTRL                 |
| 69         | PB27/PCK0       | I/O      | INPUT                |
| 68         |                 |          | OUTPUT               |
| 67         |                 |          | CTRL                 |



**Table 20.** JTAG Boundary Scan Register (Continued)

| Bit Number | Pin Name        | Pin Type | Associated BSR Cells |
|------------|-----------------|----------|----------------------|
| 66         | PD16/TD1/TPK4   | I/O      | INPUT                |
| 65         |                 |          | OUTPUT               |
| 64         |                 |          | CTRL                 |
| 63         | PD17/TD2/TPK5   | I/O      | INPUT                |
| 62         |                 |          | OUTPUT               |
| 61         |                 |          | CTRL                 |
| 60         | PD18/NPCS1/TPK6 | I/O      | INPUT                |
| 59         |                 |          | OUTPUT               |
| 58         |                 |          | CTRL                 |
| 57         | PD19/NPCS2/TPK7 | I/O      | INPUT                |
| 56         |                 |          | OUTPUT               |
| 55         |                 |          | CTRL                 |
| 54         | PD20/NPCS3/TPK8 | I/O      | INPUT                |
| 53         |                 |          | OUTPUT               |
| 52         |                 |          | CTRL                 |
| 51         | PD21/RTS0/TPK9  | I/O      | INPUT                |
| 50         |                 |          | OUTPUT               |
| 49         |                 |          | CTRL                 |
| 48         | PD22/RTS1/TPK10 | I/O      | INPUT                |
| 47         |                 |          | OUTPUT               |
| 46         |                 |          | CTRL                 |
| 45         | PD23/RTS2/TPK11 | I/O      | INPUT                |
| 44         |                 |          | OUTPUT               |
| 43         |                 |          | CTRL                 |
| 42         | PD24/RTS3/TPK12 | I/O      | INPUT                |
| 41         |                 |          | OUTPUT               |
| 40         |                 |          | CTRL                 |
| 39         | PD25/DTR1/TPK13 | I/O      | INPUT                |
| 38         |                 |          | OUTPUT               |
| 37         |                 |          | CTRL                 |
| 36         | PD26/TPK14      | I/O      | INPUT                |
| 35         |                 |          | OUTPUT               |
| 34         |                 |          | CTRL                 |

**Table 20. JTAG Boundary Scan Register (Continued)**

| Bit Number | Pin Name                   | Pin Type | Associated BSR Cells |
|------------|----------------------------|----------|----------------------|
| 33         | PD27/TPK15                 | I/O      | INPUT                |
| 32         |                            |          | OUTPUT               |
| 31         |                            |          | CTRL                 |
| 30         | PB28/FIQ                   | I/O      | INPUT                |
| 29         |                            |          | OUTPUT               |
| 28         |                            |          | CTRL                 |
| 27         | PB29/IRQ0                  | I/O      | INPUT                |
| 26         |                            |          | OUTPUT               |
| 25         |                            |          | CTRL                 |
| 24         | A0/NLB/NBS0                | Output   | OUTPUT               |
| 23         | A[3:0]/NLB/NWR2/NBS0 /NBS2 | Output   | CTRL                 |
| 22         | A1/NWR2/NBS2               | Output   | OUTPUT               |
| 21         | A2                         | Output   | OUTPUT               |
| 20         | A3                         | Output   | OUTPUT               |
| 19         | A4                         | Output   | OUTPUT               |
| 18         | A[7:4]                     | Output   | CTRL                 |
| 17         | A5                         | Output   | OUTPUT               |
| 16         | A6                         | Output   | OUTPUT               |
| 15         | A7                         | Output   | OUTPUT               |
| 14         | A8                         | Output   | OUTPUT               |
| 13         | A[11:8]                    | Output   | CTRL                 |
| 12         | A9                         | Output   | OUTPUT               |
| 11         | A10                        | Output   | OUTPUT               |
| 10         | SDA10                      | Output   | OUTPUT               |
| 9          | A11                        | Output   | OUTPUT               |
| 8          | A12                        | Output   | OUTPUT               |
| 7          | A[15:12]                   | Output   | CTRL                 |
| 6          | A13                        | Output   | OUTPUT               |
| 5          | A14                        | Output   | OUTPUT               |
| 4          | A15                        | Output   | OUTPUT               |
| 3          | A16/BA0                    | Output   | OUTPUT               |
| 2          | A17/BA1                    | Output   | OUTPUT               |
| 1          | A18                        | Output   | OUTPUT               |

## AT91RM9200 ID Code Register

Access: Read-only

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

### VERSION[31:28]: Product Version Number

Set to 0x0 = JTAGSEL is low.

Set to 0x1 = JTAGSEL is high.

### PART NUMBER[27:14]: Product Part Number

Set to 0x5b02.

### MANUFACTURER IDENTITY[11:1]

Set to 0x01f.

### Bit [0]: Required by IEEE Std. 1149.1

Set to 1.

The AT91RM9200 ID Code value is 0x15b0203f (JTAGSEL is High).

The AT91RM9200 ID Code value is 0x05b0203f (JTAGSEL is Low).



## Boot Program

### Overview

The Boot Program downloads an application in any of the AT91 products integrating a ROM. It integrates a Bootloader and a boot Uploader to assure correct information download.

The Bootloader is activated first. It looks for a sequence of eight valid ARM exception vectors in a DataFlash connected to the SPI, an EEPROM connected to the Two-wire Interface (TWI) or an 8-bit memory device connected to the external bus interface (EBI) (if device integrates EBI). All these vectors must be B-branch or LDR load register instructions except for the sixth instruction. This vector is used to store information, such as the size of the image to download and the type of DataFlash device.

If a valid sequence is found, code is downloaded into the internal SRAM. This is followed by a remap and a jump to the first address of the SRAM.

If no valid ARM vector sequence is found, the boot Uploader is started. It initializes the Debug Unit serial port (DBGU) and the USB Device Port. It then waits for any transaction and downloads a piece of code into the internal SRAM via a Device Firmware Upgrade (DFU) protocol for USB and XMODEM protocol for the DBGU. After the end of the download, it branches to the application entry point at the first address of the SRAM.

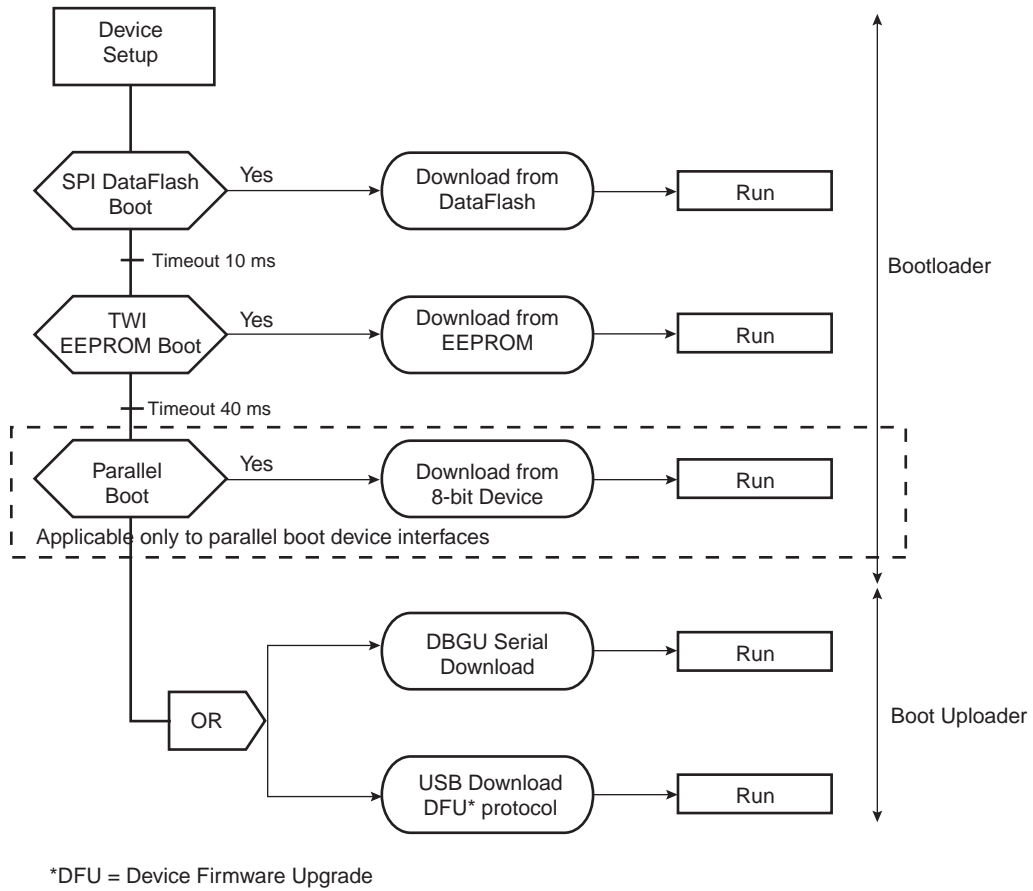
The main features of the Boot Program are:

- Default Boot Program stored in ROM-based products
- Downloads and runs an application from external storage media into internal SRAM
- Downloaded code size depends on embedded SRAM size
- Automatic detection of valid application
- Bootloader supporting a wide range of non-volatile memories
  - SPI DataFlash<sup>®</sup> connected on SPI NPCS0
  - Two-wire EEPROM
  - 8-bit parallel memories on NCS0 (if device integrates EBI)
- Boot Uploader in case no valid program is detected in external NVM and supporting several communication media
- Serial communication on a DBGU (XModem protocol)
- USB Device Port (DFU Protocol)

## Flow Diagram

The Boot Program implements the algorithm presented in Figure 15.

**Figure 15.** Boot Program Algorithm Flow Diagram



## Bootloader

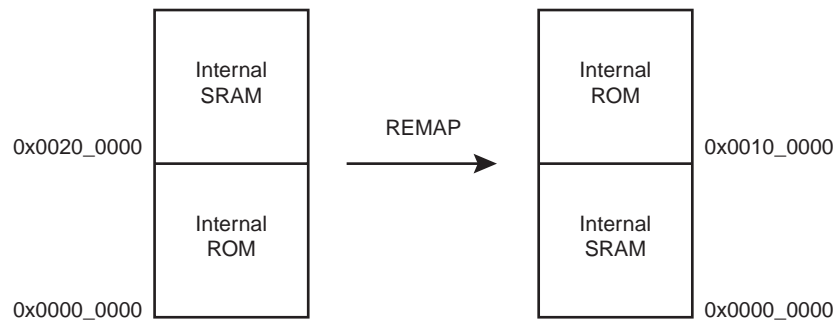
The Boot Program is started from address 0x0000\_0000 (ARM reset vector) when the on-chip boot mode is selected (BMS high during the reset, only on devices with EBI integrated). The first operation is the search for a valid program in the off-chip non-volatile memories. If a valid application is found, this application is loaded into internal SRAM and executed by branching at address 0x0000\_0000 after remap. This application may be the application code or a second-level Bootloader.

To optimize the downloaded application code size, the Boot Program embeds several functions that can be reused by the application. The Boot Program is linked at address 0x0010\_0000 but the internal ROM is mapped at both 0x0000\_0000 and 0x0010\_0000 after reset. All the call to functions is PC relative and does not use absolute addresses. The ARM vectors are present at both addresses, 0x0000\_0000 and 0x0010\_0000.

To access the functions in ROM, a structure containing chip descriptor and function entry points is defined at a fixed address in ROM.

If no valid application is detected, the debug serial port or the USB device port must be connected to allow the upload. A specific application provided by Atmel (DFU uploader) loads the application into internal SRAM through the USB. To load the application through the debug serial port, a terminal application (HyperTerminal) running the Xmodem protocol is required.

**Figure 16.** Remap Action after Download Completion



After reset, the code in internal ROM is mapped at both addresses 0x0000\_0000 and 0x0010\_0000:

|        |          |     |                  |    |          |     |                  |
|--------|----------|-----|------------------|----|----------|-----|------------------|
| 100000 | ea00000b | B   | 0x2c             | 00 | ea00000b | B   | 0x2c             |
| 100004 | e59ff014 | LDR | PC, [PC, 20]     | 04 | e59ff014 | LDR | PC, [PC, 20]     |
| 100008 | e59ff014 | LDR | PC, [PC, 20]     | 08 | e59ff014 | LDR | PC, [PC, 20]     |
| 10000c | e59ff014 | LDR | PC, [PC, 20]     | 0c | e59ff014 | LDR | PC, [PC, 20]     |
| 100010 | e59ff014 | LDR | PC, [PC, 20]     | 10 | e59ff014 | LDR | PC, [PC, 20]     |
| 100014 | 00001234 | LDR | PC, [PC, 20]     | 14 | 00001234 | LDR | PC, [PC, 20]     |
| 100018 | e51fff20 | LDR | PC, [PC, -0xf20] | 18 | e51fff20 | LDR | PC, [PC, -0xf20] |
| 10001c | e51fff20 | LDR | PC, [PC, -0xf20] | 1c | e51fff20 | LDR | PC, [PC, -0xf20] |

## Valid Image Detection

The Bootloader software looks for a valid application by analyzing the first 32 bytes corresponding to the ARM exception vectors. These bytes must implement ARM instructions for either branch or load PC with PC relative addressing. The sixth vector, at offset 0x18, contains the size of the image to download and the DataFlash parameters.

The user must replace this vector with his own vector.

Figure 17. LDR Opcode

|    |    |    |    |    |    |    |    |    |    |    |   |    |    |  |
|----|----|----|----|----|----|----|----|----|----|----|---|----|----|--|
| 31 | 28 | 27 | 24 | 23 | 20 | 19 | 16 | 15 | 12 | 11 | 0 |    |    |  |
| 1  | 1  | 1  | 0  | 1  | 1  | I  | P  | U  | 1  | W  | 0 | Rn | Rd |  |

Figure 18. B Opcode

|    |    |    |    |    |   |   |   |                  |  |  |  |
|----|----|----|----|----|---|---|---|------------------|--|--|--|
| 31 | 28 | 27 | 24 | 23 | 0 |   |   |                  |  |  |  |
| 1  | 1  | 1  | 0  | 1  | 0 | 1 | 0 | Offset (24 bits) |  |  |  |

Unconditional instruction: 0xE for bits 31 to 28

Load PC with PC relative addressing instruction:

- Rn = Rd = PC = 0xF
- I==1
- P==1
- U offset added (U==1) or subtracted (U==0)
- W==1

## Example

An example of valid vectors:

|     |          |     |                 |                           |
|-----|----------|-----|-----------------|---------------------------|
| 00  | ea00000b | B   | 0x2c            |                           |
| 004 | e59ff014 | LDR | PC, [PC,20]     |                           |
| 08  | e59ff014 | LDR | PC, [PC,20]     |                           |
| 0c  | e59ff014 | LDR | PC, [PC,20]     |                           |
| 10  | e59ff014 | LDR | PC, [PC,20]     |                           |
| 14  | 00001234 | LDR | PC, [PC,20]     | <- Code size = 4660 bytes |
| 18  | e51fff20 | LDR | PC, [PC,-0xf20] |                           |
| 1c  | e51fff20 | LDR | PC, [PC,-0xf20] |                           |

In download mode (DataFlash, EEPROM or 8-bit memory in device with EBI integrated), the size of the image to load into SRAM is contained in the location of the sixth ARM vector. Thus the user must replace this vector by the correct vector for his application.



## Structure of ARM Vector 6

The ARM exception vector 6 is used to store information needed by the Boot ROM downloader. This information is described below.

**Figure 19.** Structure of the ARM vector 6

|                     |    |                 |    |          |   |                                    |   |
|---------------------|----|-----------------|----|----------|---|------------------------------------|---|
| 31                  | 17 | 16              | 13 | 12       | 8 | 7                                  | 0 |
| DataFlash page size |    | Number of pages |    | Reserved |   | Nb of 512 bytes blocks to download |   |

The first eight bits contain the number of blocks to download. The size of a block is 512 bytes, allowing download of up to 128K bytes.

The bits 13 to 16 determine the DataFlash page number.

– DataFlash page number =  $2^{(\text{Nb of pages})}$

The last 15 bits contain the DataFlash page size.

**Table 21.** DataFlash Device

| Device     | Density   | Page Size (bytes) | Number of pages |
|------------|-----------|-------------------|-----------------|
| AT45DB011B | 1 Mbit    | 264               | 512             |
| AT45DB021B | 2 Mbits   | 264               | 1024            |
| AT45DB041B | 4 Mbits   | 264               | 2048            |
| AT45DB081B | 8 Mbits   | 264               | 4096            |
| AT45DB161B | 16 Mbits  | 528               | 4096            |
| AT45DB321B | 32 Mbits  | 528               | 8192            |
| AT45DB642  | 64 Mbits  | 1056              | 8192            |
| AT45DB1282 | 128 Mbits | 1056              | 16384           |
| AT45DB2562 | 256 Mbits | 2112              | 16384           |

### Example

The following vector contains the information to describe a AT45DB642 DataFlash which contains 11776 bytes to download.

Vector 6 is 0x0841A017 (00001000010000011010000000010111b):

Size to download:  $0x17 * 512 \text{ bytes} = 11776 \text{ bytes}$

Number pages (1101b): 13 ==> Number of DataFlash pages =  $2^{13} = 8192$

DataFlash page size(000010000100000b) = 1056

For download in the EEPROM or 8-bit external memory (if device integrates EBI), only the size to be downloaded is decoded.

## Bootloader Sequence

The Boot Program performs device initialization followed by the download procedure. If unsuccessful, the upload is done via the USB or debug serial port.

### Device Initialization

Initialization follows the steps described below:

1. PLL setup
  - PLLB is initialized to generate a 48 MHz clock necessary to use the USB Device. A register located in the Power Management Controller (PMC) determines the frequency of the main oscillator and thus the correct factor for the PLLB. Table 22 defines the crystals supported by the Boot Program.

**Table 22.** Crystals Supported by Software Auto-detection (MHz)

|          |           |        |          |          |
|----------|-----------|--------|----------|----------|
| 3.0      | 3.2768    | 3.6864 | 3.84     | 4.0      |
| 4.433619 | 4.9152    | 5.0    | 5.24288  | 6.0      |
| 6.144    | 6.4       | 6.5536 | 7.159090 | 7.3728   |
| 7.864320 | 8.0       | 9.8304 | 10.0     | 11.05920 |
| 12.0     | 12.288    | 13.56  | 14.31818 | 14.7456  |
| 16.0     | 17.734470 | 18.432 | 20.0     | 24.0     |
| 25.0     | 28.224    | 32.0   | 33.0     |          |

2. Stacks setup for each ARM mode
3. Main oscillator frequency detection
4. Interrupt controller setup
5. C variables initialization
6. Branch main function

### Download Procedure

The download procedure checks for a valid boot on several devices. The first device checked is a serial DataFlash connected to the NPCS0 of the SPI, followed by the serial EEPROM connected to the TWI and by an 8-bit parallel memory connected on NCS0 of the External Bus Interface (if EBI is implemented in the product).

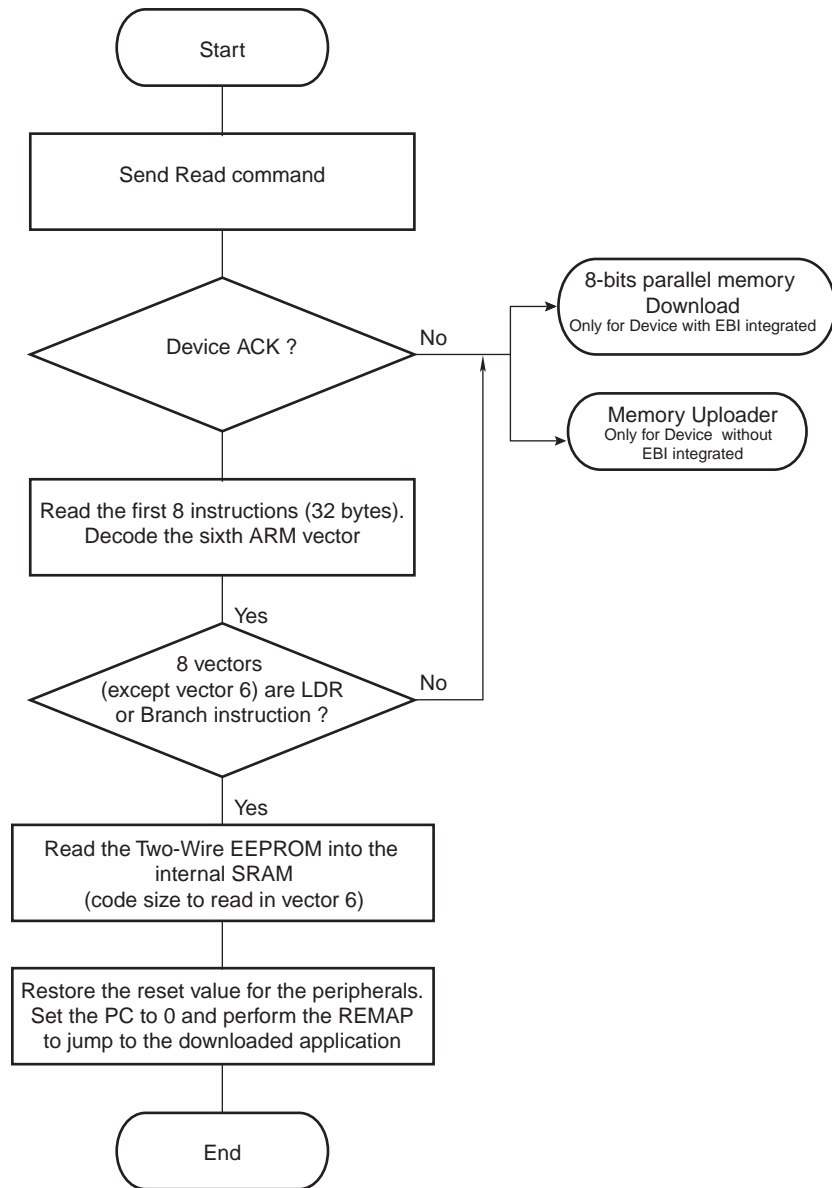
**Serial DataFlash Download**

The Boot Program supports all Atmel DataFlash devices. Table 21 summarizes the parameters to include in the ARM vector 6 for all devices.

The DataFlash has a Status Register that determines all the parameters required to access the device.

Thus, to be compatible with the future design of the DataFlash, parameters are coded in the ARM vector 6.

**Figure 20.** Serial DataFlash Download



## Serial Two-wire EEPROM Download

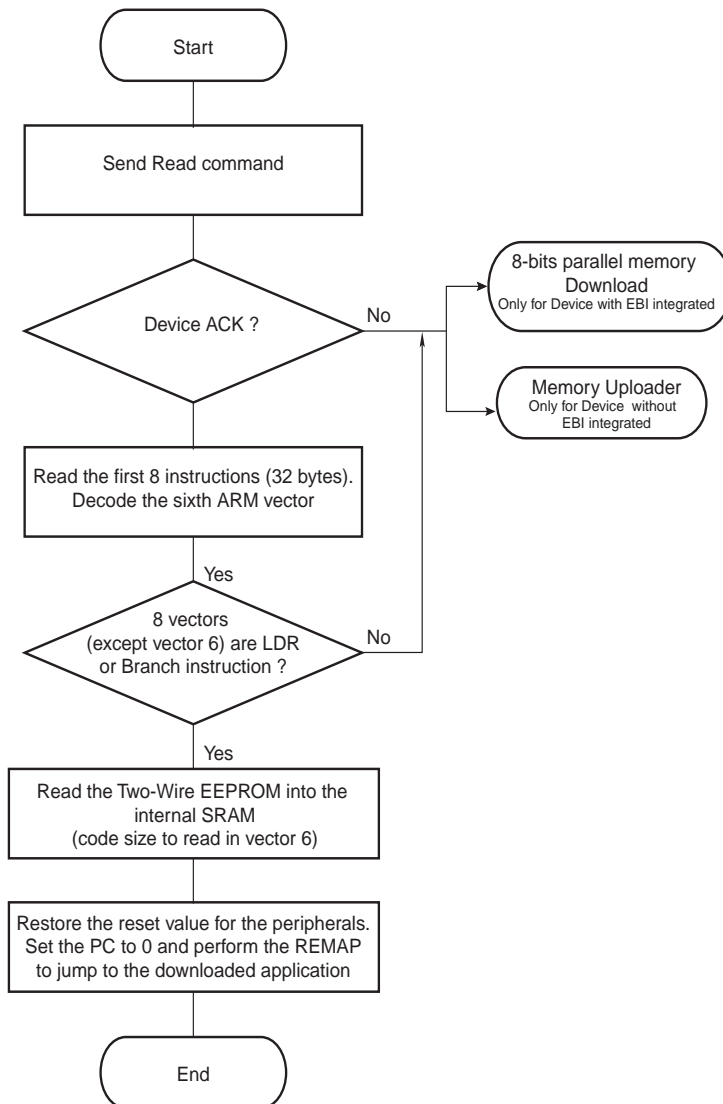
Generally, serial EEPROMs have no identification code. The bootloader checks for an acknowledgment on the first read. The device address on the two-wire bus must be 0x0.

The bootloader supports the devices listed in Table 23.

**Table 23.** Supported EEPROM Devices

| Device   | Size      | Organization         |
|----------|-----------|----------------------|
| AT24C16A | 16 Kbits  | 16 bytes page write  |
| AT24C164 | 16 Kbits  | 16 bytes page write  |
| AT24C32  | 32 Kbits  | 32 bytes page write  |
| AT24C64  | 64 Kbits  | 32 bytes page write  |
| AT24C128 | 128 Kbits | 64 bytes page write  |
| AT24C256 | 256 Kbits | 64 bytes page write  |
| AT24C512 | 528 Kbits | 128 bytes page write |

**Figure 21.** Serial Two-Wire EEPROM Download

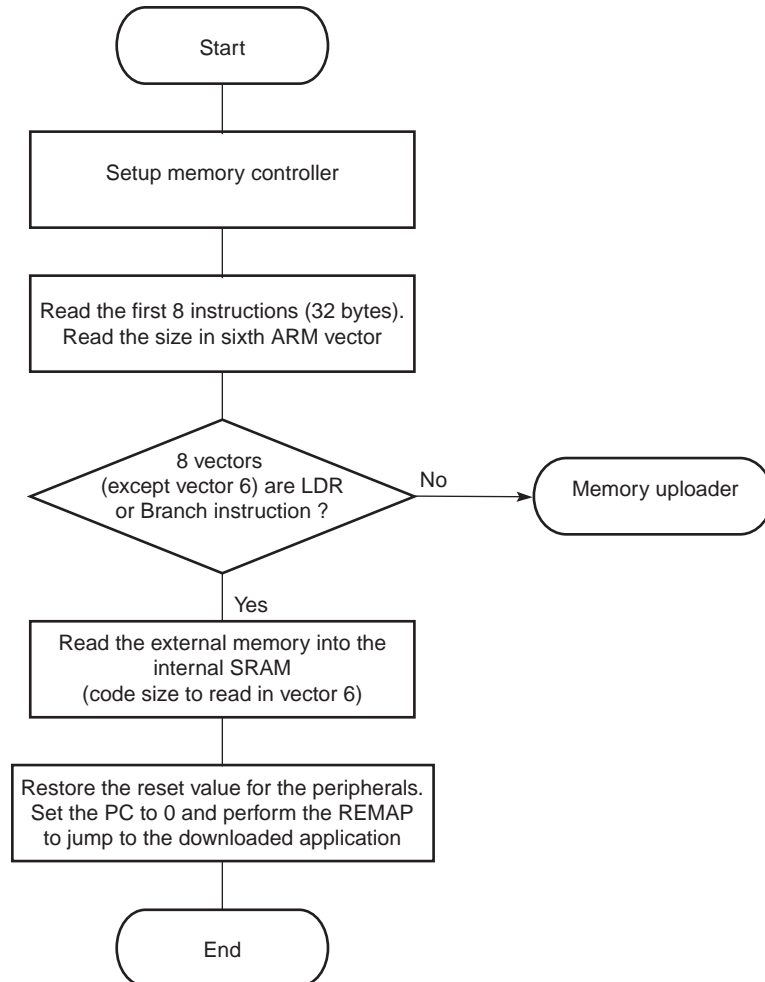


**8-bit Parallel Flash Download (Applicable to Devices with EBI)**

Eight-bit parallel Flash download is supported if the product integrates an External Bus Interface (EBI).

All 8-bit memory devices supported by the EBI when NCS0 is configured in 8-bit data bus width are supported by the bootloader.

**Figure 22.** 8-bit Parallel Flash Download



## Boot Uploader

If no valid boot device has been found during the Bootloader sequence, initialization of serial communication devices (DBGU and USB device ports) is performed.

- Initialization of the DBGU serial port (115200 bauds, 8, N, 1) and Xmodem protocol start
- Initialization of the USB Device Port and DFU protocol start
- Download of the application

The boot Uploader performs the DFU and Xmodem protocols to upload the application into internal SRAM at address 0x0020\_0000.

The Boot Program uses a piece of internal SRAM for variables and stacks. To prevent any upload error, the size of the application to upload must be less than the SRAM size minus 3K bytes.

After the download, the peripheral registers are reset, the interrupts are disabled and the remap is performed. After the remap, the internal SRAM is at address 0x0000\_0000 and the internal ROM at address 0x0010\_0000. The instruction setting the PC to 0 is the one just after the remap command. This instruction is fetched in the pipe before doing the remap and executed just after. This fetch cycle executes the downloaded image.

## External Communication Channels

### DBGU Serial Port

The upload is performed through the DBGU serial port initialized to 115200 Baud, 8, n, 1.

The DBGU sends the character 'C' (0x43) to start an Xmodem protocol. Any terminal performing this protocol can be used to send the application file to the target. The size of the binary file to send depends on the SRAM size embedded in the product (Refer to the microcontroller datasheet to determine SRAM size embedded in the microcontroller). In all cases, the size of the binary file must be lower than SRAM size because the Xmodem protocol requires some SRAM memory to work.

### Xmodem Protocol

The Xmodem protocol supported is the 128-byte length block. This protocol uses a two character CRC-16 to guarantee detection of a maximum bit error.

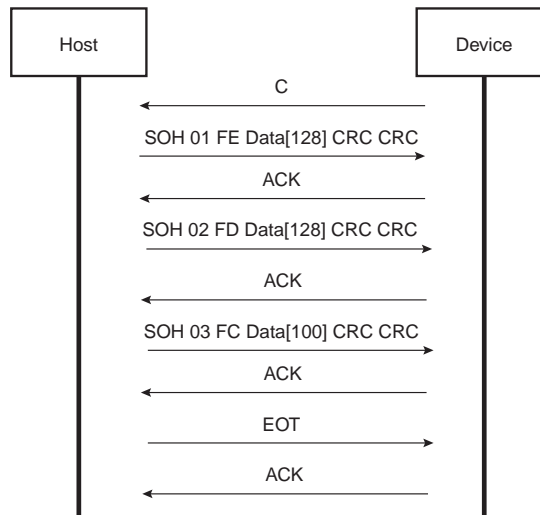
Xmodem protocol with CRC is accurate provided both sender and receiver report successful transmission. Each block of the transfer looks like:

<SOH><blk #><255-blk #><--128 data bytes--><checksum> in which:

- <SOH> = 01 hex
- <blk #> = binary number, starts at 01, increments by 1, and wraps 0FFH to 00H (not to 01)
- <255-blk #> = 1's complement of the blk#.
- <checksum> = 2 bytes CRC16

Figure 23 shows a transmission using this protocol.

Figure 23. Xmodem Transfer Example



**USB Device Port**

A 48 MHz USB clock is necessary to use USB Device port. It has been programmed earlier in the device initialization with PLLB configuration.

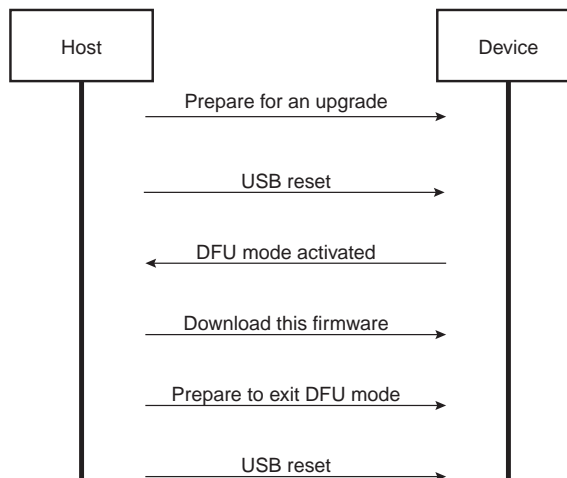
**DFU Protocol**

The DFU allows upgrade of the firmware of USB devices. The DFU algorithm is a part of the USB specification. For more details, refer to “USB Device Firmware Upgrade Specification, Rev. 1.0”.

There are four distinct steps when carrying out a firmware upgrade:

1. Enumeration: The device informs the host of its capabilities.
2. Reconfiguration: The host and the device agree to initiate a firmware upgrade.
3. Transfer: The host transfers the firmware image to the device. Status requests are employed to maintain synchronization between the host and the device.
4. Manifestation: Once the device reports to the host that it has completed the reprogramming operations, the host issues a reset and the device executes the upgraded firmware.

Figure 24. DFU Protocol



## Hardware and Software Constraints

The software limitations of the Boot Program are:

- The downloaded code size is less than the SRAM size embedded in the product.
- The device address of the EEPROM must be 0 on the TWI bus.
- The code is always downloaded from the device address 0x0000\_0000 (DataFlash, EEPROM) to the address 0x0000\_0000 of the internal SRAM (after remap).
- The downloaded code must be position-independent or linked at address 0x0000\_0000.

The hardware limitations of the Boot Program are:

- The DataFlash must be connected to NPCS0 of the SPI.
- The 8-bit parallel Flash must be connected to NCS0 of the EBI (applicable for devices with integrated EBI).

The SPI and TWI drivers use several PIOs in alternate functions to communicate with devices. Care must be taken when these PIOs are used by the application. The devices connected could be unintentionally driven at boot time, and electrical conflicts between SPI or TWI output pins and the connected devices may appear.

To assure correct functionality, it is recommended to plug in critical devices to other pins or to boot on an external 16-bit parallel memory (if product integrates an EBI) by setting bit BMS.

Table 24 contains a list of pins that are driven during the Boot Program execution. These pins are driven during the boot sequence for a period of about 6 ms if no correct boot program is found. The download through the TWI takes about 5 sec for 64K bytes due to the TWI bit rate (100 Kbits/s).

For the DataFlash driven by SPCK signal at 12 MHz, the time to download 64K bytes is reduced to 66 ms.

Before performing the jump to the application in internal SRAM, all the PIOs and peripherals used in the Boot Program are set to their reset state.

**Table 24.** Pins Driven during Boot Program Execution

| Pin Used             | SPI (Dataflash) | TWI (EEPROM) |
|----------------------|-----------------|--------------|
| MOSI <sup>(1)</sup>  | O               | X            |
| SPCK <sup>(1)</sup>  | O               | X            |
| NPCS0 <sup>(1)</sup> | O               | X            |
| TWD <sup>(1)</sup>   | X               | I/O          |
| TWCK <sup>(1)</sup>  | X               | O            |

Note: 1. See “Peripheral Multiplexing on PIO Lines” on page 18.



## Embedded Software Services

### Overview

An embedded software service is an independent software object that drives device resources for frequently implemented tasks. The object-oriented approach of the software provides an easy way to access services to build applications.

An AT91 service has several purposes:

- It gives software examples dedicated to the AT91 devices.
- It can be used on several AT91 device families.
- It offers an interface to the software stored in the ROM.

The main features of the software services are:

- Compliant with ATPCS
- Compliant with ANSI/ISO Standard C
- Compiled in ARM/Thumb Interworking
- ROM Entry Service
- Tempo, Xmodem and DataFlash services
- CRC and Sine tables

### Service Definition

#### Service Structure

##### Structure Definition

A service structure is defined in C header files.

This structure is composed of data members and pointers to functions (methods) and is similar to a class definition. There is no protection of data access or methods access. However, some functions can be used by the customer application or other services and so be considered as public methods. Similarly, other functions are not invoked by them. They can be considered as private methods. This is also valid for data.

##### Methods

In the service structure, pointers to functions are supposed to be initialized by default to the standard functions. Only the default standard functions reside in ROM. Default methods can be overloaded by custom application methods.

Methods do not declare any static variables nor invoke global variables. All methods are invoked with a pointer to the service structure. A method can access and update service data without restrictions.

Similarly, there is no polling in the methods. In fact, there is a method to start the functionality (a read to give an example), a method to get the status (is the read achieved?), and a callback, initialized by the start method. Thus, using service, the client application carries out a synchronous read by starting the read and polling the status, or an asynchronous read specifying a callback when starting the read operation.

##### Service Entry Point

Each AT91 service, except for the **ROM Entry Service** (see page 101), defines a function named `AT91F_Open_<Service>`. It is the only entry point defined for a service. Even if the functions `AT91F_Open_<Service>` may be compared with object constructors, they do not act as constructors in that they initiate the service structure but they do not allocate it. Thus the customer application must allocate it.

### Example

```
// Allocation of the service structure
AT91S_Pipe pipe;
// Opening of the service
AT91PS_Pipe pPipe = AT91F_OpenPipe(&pipe, ...);
```

Method pointers in the service structure are initialized to the default methods defined in the AT91 service. Other fields in the service structure are initialized to default values or with the arguments of the function `AT91F_Open_<Service>`.

In summary, an application must know what the service structure is and where the function `AT91F_Open_<Service>` is.

The default function `AT91F_Open_<Service>` may be redefined by the application or comprised in an application-defined function.

## Using a Service

### Opening a Service

The entry point to a service is established by initializing the service structure. An open function is associated with each service structure, except for the ROM Entry Service (see page 101). Thus, only the functions `AT91F_Open_<service>` are visible from the user side. Access to the service methods is made via function pointers in the service structure.

The function `AT91F_Open_<service>` has at least one argument: a pointer to the service structure that must be allocated elsewhere. It returns a pointer to the base service structure or a pointer to this service structure.

The function `AT91F_Open_<service>` initializes all data members and method pointers. All function pointers in the service structure are set to the service's functions.

The advantage of this method is to offer a single entry point for a service. The methods of a service are initialized by the open function and each member can be overloaded.

### Overloading a Method

Default methods are defined for all services provided in ROM. These methods may not be adapted to a project requirement. It is possible to overload default methods by methods defined in the project.

A method is a pointer to a function. This pointer is initialized by the function `AT91F_Open_<Service>`. To overload one or several methods in a service, the function pointer must be updated to the new method.

It is possible to overload just one method of a service or all the methods of a service. In this latter case, the functionality of the service is user-defined, but still works on the same data structure.

Note: Calling the default function `AT91F_Open_<Service>` ensures that all methods and data are initialized.

This can be done by writing a new function `My_OpenService()`. This new Open function must call the library-defined function `AT91F_Open_<Service>`, and then update one or several function pointers:

**Table 25.** Overloading a Method with the Overloading of the Open Service Function

| Default service behavior in ROM   | Overloading AT91F_ChildMethod by My_ChildMethod  |
|---|--|
| <pre> // Defined in <i>embedded_services.h</i> typedef struct _AT91S_Service {     char data;     char (*MainMethod) ();     char (*ChildMethod) (); } AT91S_Service, * AT91PS_Service;  // Defined in <i>obj_service.c (in ROM)</i> char AT91F_MainMethod () { }  char AT91F_ChildMethod () { }  // Init the service with default methods AT91PS_Service AT91F_OpenService( AT91PS_Service pService) {     pService-&gt;data = 0;     pService-&gt;MainMethod =AT91F_MainMethod;     pService-&gt;ChildMethod=AT91F_ChildMethod;     return pService; } </pre> | <pre> // <i>My_ChildMethod</i> will replace <i>AT91F_ChildMethod</i> char My_ChildMethod () { }  // <i>Overloading Open Service Method</i> AT91PS_Service My_OpenService( AT91PS_Service pService) {     AT91F_OpenService(pService);  // <i>Overloading ChildMethod default value</i>     pService-&gt;ChildMethod= My_ChildMethod;     return pService; }  // <i>Allocation of the service structure</i> AT91S_Service service;  // <i>Opening of the service</i> AT91PS_Service pService = My_OpenService(&amp;service); </pre> |

This also can be done directly by overloading the method after the use of `AT91F_Open_<Service>` method:

**Table 26.** Overloading a Method without the Overloading of the Open Service Function.

| Default service behavior in ROM   | Overloading <code>AT91F_ChildMethod</code> by <code>My_ChildMethod</code>   |
|---|---|
| <pre> // Defined in embedded_services.h typedef struct _AT91S_Service {     char data;     char (*MainMethod) ();     char (*ChildMethod) (); } AT91S_Service, * AT91PS_Service;  // Defined in obj_service.c (in ROM) char AT91F_MainMethod () { }  char AT91F_ChildMethod () { }  // Init the service with default methods AT91PS_Service AT91F_OpenService( AT91PS_Service pService) {     pService-&gt;data = 0;     pService-&gt;MainMethod =AT91F_MainMethod;     pService-&gt;ChildMethod=AT91F_ChildMethod;     return pService; } </pre> | <pre> // My_ChildMethod will replace AT91F_ChildMethod char My_ChildMethod () { }  // Allocation of the service structure AT91S_Service service;  // Opening of the service AT91PS_Service pService = AT91F_OpenService(&amp;service);  // Overloading ChildMethod default value pService-&gt;ChildMethod= My_ChildMethod; </pre> |

## Embedded Software Services

### Definition

Several AT91 products embed ROM. In most cases, the ROM integrates a bootloader and several services that may speed up the application and reduce the application code size.

When software is fixed in the ROM, the address of each object (function, constant, table, etc.) must be related to a customer application. This is done by providing an address table to the linker. For each version of ROM, a new address table must be provided and all client applications must be recompiled.

The Embedded Software Services offer another solution to access objects stored in ROM. For each embedded service, the customer application requires only the address of the Service Entry Point (see page 97).

Even if these services have only one entry point (AT91F\_Open\_<Service> function), they must be specified to the linker. The Embedded Software Services solve this problem by providing a dedicated service: the ROM Entry Service.

The goal of this product-dedicated service is to provide just one address to access all ROM functionalities.

### ROM Entry Service

The ROM Entry Service of a product is a structure named AT91S\_RomBoot. Some members of this structure point to the open functions of all services stored in ROM (function AT91F\_Open\_<Service>) but also the CRC and Sine Arrays. Thus, only the address of the AT91S\_RomBoot has to be published.

**Table 27.** Initialization of the ROM Entry Service and Use with an Open Service Method

| Application Memory Space   | ROM Memory Space   |
|--|--|
| <pre>// Init the ROM Entry Service AT91S_RomBoot const *pAT91; pAT91 = AT91C_ROM_BOOT_ADDRESS;  // Allocation of the service structure AT91S_CtlTempo tempo;  // Call the Service Open method pAT91-&gt;OpenCtlTempo(&amp;tempo, ...);  // Use of tempo methods tempo.CtlTempoCreate(&amp;tempo, ...);</pre> | <pre>AT91S_TempoStatus AT91F_OpenCtlTempo(     AT91PS_CtlTempo pCtlTempo,     void const *pTempoTimer ) {     ... }  AT91S_TempoStatus AT91F_CtlTempoCreate (     AT91PS_CtlTempo pCtrl,     AT91PS_SvcTempo pTempo) {     ... }</pre> |

The application obtains the address of the ROM Entry Service and initializes an instance of the AT91S\_RomBoot structure. To obtain the Open Service Method of another service stored in ROM, the application uses the appropriate member of the AT91S\_RomBoot structure.

The address of the AT91S\_RomBoot can be found at the beginning of the ROM, after the exception vectors.

## Tempo Service

### Presentation

The Tempo Service allows a single hardware system timer to support several software timers running concurrently. This works as an object notifier.

There are two objects defined to control the Tempo Service: `AT91S_CtlTempo` and `AT91S_SvcTempo`.

The application declares one instance of `AT91S_CtlTempo` associated with the hardware system timer. Additionally, it controls a list of instances of `AT91S_SvcTempo`.

Each time the application requires another timer, it asks the `AT91S_CtlTempo` to create a new instance of `AT91S_SvcTempo`, then the application initializes all the settings of `AT91S_SvcTempo`.

### Tempo Service Description

Table 28. Tempo Service Methods

| Associated Function Pointers & Methods Used by Default  | Description   |
|---|---|
| <pre>// Typical Use: pAT91-&gt;OpenCtlTempo(...);  // Default Method: AT91S_TempoStatus AT91F_OpenCtlTempo( AT91PS_CtlTempo pCtlTempo, void const *pTempoTimer)</pre>                         | <p>Member of <code>AT91S_RomBoot</code> structure.<br/>Corresponds to the Open Service Method for the Tempo Service.</p> <p><u>Input Parameters:</u><br/>Pointer on a Control Tempo Object.<br/>Pointer on a System Timer Descriptor Structure.</p> <p><u>Output Parameters:</u><br/>Returns 0 if <code>OpenCtlTempo</code> successful.<br/>Returns 1 if not.</p> |
| <pre>// Typical Use: AT91S_CtlTempo ctlTempo; ctlTempo.CtlTempoStart(...);  // Default Method: AT91S_TempoStatus AT91F_STStart(void * pTimer)</pre>   | <p>Member of <code>AT91S_CtlTempo</code> structure.<br/>Start of the Hardware System Timer associated.</p> <p><u>Input Parameters:</u><br/>Pointer on a Void Parameter corresponding to a System Timer Descriptor Structure.</p> <p><u>Output Parameters:</u><br/>Returns 2.</p>  |
| <pre>// Typical Use: AT91S_CtlTempo ctlTempo; ctlTempo.CtlTempoIsStart(...);  // Default Method: AT91S_TempoStatus AT91F_STIsStart( AT91PS_CtlTempo pCtrl)</pre>                              | <p>Member of <code>AT91S_CtlTempo</code> structure.</p> <p><u>Input Parameters:</u><br/>Pointer on a Control Tempo Object.</p> <p><u>Output Parameters:</u><br/>Returns the Status Register of the System Timer.</p>  |
| <pre>// Typical Use: AT91S_CtlTempo ctlTempo; ctlTempo.CtlTempoCreate(...);  // Default Method: AT91S_TempoStatus AT91F_CtlTempoCreate ( AT91PS_CtlTempo pCtrl, AT91PS_SvcTempo pTempo)</pre> | <p>Member of <code>AT91S_CtlTempo</code> structure.<br/>Insert a software timer in the <code>AT91S_SvcTempo</code>'s list.</p> <p><u>Input Parameters:</u><br/>Pointer on a Control Tempo Object.<br/>Pointer on a Service Tempo Object to insert.</p> <p><u>Output Parameters:</u><br/>Returns 0 if the software tempo was created.<br/>Returns 1 if not.</p>    |

**Table 28.** Tempo Service Methods (Continued)

| Associated Function Pointers & Methods Used by Default   | Description  |
|--|--|
| <pre>// Typical Use: AT91S_CtlTempo ctlTempo; ctlTempo.CtlTempoRemove(...);  // Default Method: AT91S_TempoStatus AT91F_CtlTempoRemove (AT91PS_CtlTempo pCtrl, AT91PS_SvcTempo pTempo)</pre>   | <p>Member of AT91S_CtlTempo structure.<br/>Remove a software timer in the list.</p> <p><u>Input Parameters:</u><br/>Pointer on a Control Tempo Object.<br/>Pointer on a Service Tempo Object to remove.</p> <p><u>Output Parameters:</u><br/>Returns 0 if the tempo was created.<br/>Returns 1 if not.</p>   |
| <pre>// Typical Use: AT91S_CtlTempo ctlTempo; ctlTempo.CtlTempoTick(...);  // Default Method: AT91S_TempoStatus AT91F_CtlTempoTick (AT91PS_CtlTempo pCtrl)</pre>   | <p>Member of AT91S_CtlTempo structure.<br/>Refresh all the software timers in the list. Update their timeout and check if callbacks have to be launched. So, for example, this function has to be used when the hardware timer starts a new periodic interrupt if period interval timer is used.</p> <p><u>Input Parameters:</u><br/>Pointer on a Control Tempo Object.</p> <p><u>Output Parameters:</u><br/>Returns 1.</p>  |
| <pre>// Typical Use: AT91S_SvcTempo svcTempo; svcTempo.Start(...);  // Default Method: AT91S_TempoStatus AT91F_SvcTempoStart ( AT91PS_SvcTempo pSvc, unsigned int timeout, unsigned int reload, void (*callback) (AT91S_TempoStatus, void *), void *pData)</pre> | <p>Member of AT91S_SvcTempo structure.<br/>Start a software timer.</p> <p><u>Input Parameters:</u><br/>Pointer on a Service Tempo Object.<br/>Timeout to apply.<br/>Number of times to reload the tempo after timeout completed for periodic execution.<br/>Callback on a method to launch once the timeout completed.<br/>Allows to have a hook on the current service.</p> <p><u>Output Parameters:</u><br/>Returns 1.</p> |
| <pre>// Typical Use: AT91S_SvcTempo svcTempo; svcTempo.Stop(...);  // Default Method: AT91S_TempoStatus AT91F_SvcTempoStop ( AT91PS_SvcTempo pSvc)</pre>   | <p>Member of AT91S_SvcTempo structure.<br/>Force to stop a software timer.</p> <p><u>Input Parameters:</u><br/>Pointer on a Service Tempo Object.</p> <p><u>Output Parameters:</u><br/>Returns 1.</p>  |

**Note:** AT91S\_TempoStatus corresponds to an unsigned int.

## Using the Service

The first step is to find the address of the open service method `AT91F_OpenCtlTempo` using the ROM Entry Service.

Allocate one instance of `AT91S_CtlTempo` and `AT91S_SvcTempo` in the application memory space:

```
// Allocate the service and the control tempo
AT91S_CtlTempo ctlTempo;
AT91S_SvcTempo svcTempo1;
```

Initialize the `AT91S_CtlTempo` instance by calling the `AT91F_OpenCtlTempo` function:

```
// Initialize service
pAT91->OpenCtlTempo(&ctlTempo, (void *) &(pAT91->SYSTIMER_DESC));
```

At this stage, the application can use the `AT91S_CtlTempo` service members.

If the application wants to overload an object member, it can be done now. For example, if `AT91F_CtlTempoCreate(&ctlTempo, &svcTempo1)` method is to be replaced by the application defined as `my_CtlTempoCreate(...)`, the procedure is as follows:

```
// Overload AT91F_CtlTempoCreate
ctlTempo.CtlTempoCreate = my_CtlTempoCreate;
```

In most cases, initialize the `AT91S_SvcTempo` object by calling the `AT91F_CtlTempoCreate` method of the `AT91S_CtlTempo` service:

```
// Init the svcTempo1, link it to the AT91S_CtlTempo object
ctlTempo.CtlTempoCreate(&ctlTempo, &svcTempo1);
```

Start the timeout by calling `Start` method of the `svcTempo1` object. Depending on the function parameters, either a callback is started at the end of the countdown or the status of the timeout is checked by reading the `TickTempo` member of the `svcTempo1` object.

```
// Start the timeout
svcTempo1.Start(&svcTempo1, 100, 0, NULL, NULL);
// Wait for the timeout of 100 (unity depends on the timer programming)
// No repetition and no callback.
while (svcTempo1.TickTempo);
```

When the application needs another software timer to control a timeout, it:

- Allocates one instance of `AT91S_SvcTempo` in the application memory space

```
// Allocate the service
AT91S_SvcTempo svcTempo2;
```

- Initializes the `AT91S_SvcTempo` object calling the `AT91F_CtlTempoCreate` method of the `AT91S_CtlTempo` service:

```
// Init the svcTempo2, link it to the AT91S_CtlTempo object
ctlTempo.CtlTempoCreate(&ctlTempo, &svcTempo2);
```



## Xmodem Service

### Presentation

The Xmodem service is an application of the communication pipe abstract layer. This layer is media-independent (USART, USB, etc.) and gives entry points to carry out reads and writes on an abstract media, the pipe.

### Communication Pipe Service

The pipe communication structure is a virtual structure that contains all the functions required to read and write a buffer, regardless of the communication media and the memory management.

The pipe structure defines:

- a pointer to a communication service structure `AT91PS_SvcComm`
- a pointer to a buffer manager structure `AT91PS_Buffer`
- pointers on read and write functions
- pointers to callback functions associated to the read and write functions

The following structure defines the pipe object:

```
typedef struct _AT91S_Pipe
{
    // A pipe is linked with a peripheral and a buffer
    AT91PS_SvcComm pSvcComm;
    AT91PS_Buffer  pBuffer;

    // Callback functions with their arguments
    void (*WriteCallback) (AT91S_PipeStatus, void *);
    void (*ReadCallback)  (AT91S_PipeStatus, void *);
    void *pPrivateReadData;
    void *pPrivateWriteData;

    // Pipe methods
    AT91S_PipeStatus (*Write) (
        struct _AT91S_Pipe *pPipe,
        char const *      pData,
        unsigned int      size,
        void               (*callback) (AT91S_PipeStatus, void *),
        void               *privateData);
    AT91S_PipeStatus (*Read) (
        struct _AT91S_Pipe *pPipe,
        char               *pData,
        unsigned int      size,
        void               (*callback) (AT91S_PipeStatus, void *),
        void               *privateData);
    AT91S_PipeStatus (*AbortWrite) (struct _AT91S_Pipe *pPipe);
    AT91S_PipeStatus (*AbortRead)  (struct _AT91S_Pipe *pPipe);
    AT91S_PipeStatus (*Reset)      (struct _AT91S_Pipe *pPipe);
    char (*IsWritten) (struct _AT91S_Pipe *pPipe, char const *pVoid);
    char (*IsReceived) (struct _AT91S_Pipe *pPipe, char const *pVoid);
} AT91S_Pipe, *AT91PS_Pipe;
```

The Xmodem protocol implementation demonstrates how to use the communication pipe.

### Description of the Buffer Structure

The AT91PS\_Buffer is a pointer to the AT91S\_Buffer structure manages the buffers. This structure embeds the following functions:

- pointers to functions that manage the read buffer
- pointers to functions that manage the write buffer

All the functions can be overloaded by the application to adapt buffer management.

A simple implementation of buffer management for the **Xmodem Service** is provided in the boot ROM source code.

```
typedef struct _AT91S_Buffer
{
    struct _AT91S_Pipe *pPipe;
    void *pChild;

    // Functions invoked by the pipe
    AT91S_BufferStatus (*SetRdBuffer)      (struct _AT91S_Buffer *pSBuffer, char
    *pBuffer, unsigned int Size);
    AT91S_BufferStatus (*SetWrBuffer)      (struct _AT91S_Buffer *pSBuffer, char const
    *pBuffer, unsigned int Size);
    AT91S_BufferStatus (*RstRdBuffer)      (struct _AT91S_Buffer *pSBuffer);
    AT91S_BufferStatus (*RstWrBuffer)      (struct _AT91S_Buffer *pSBuffer);
    char (*MsgWritten)      (struct _AT91S_Buffer *pSBuffer, char const *pBuffer);
    char (*MsgRead)         (struct _AT91S_Buffer *pSBuffer, char const *pBuffer);

    // Functions invoked by the peripheral
    AT91S_BufferStatus (*GetWrBuffer)      (struct _AT91S_Buffer *pSBuffer, char const
    **pData, unsigned int *pSize);
    AT91S_BufferStatus (*GetRdBuffer)      (struct _AT91S_Buffer *pSBuffer, char
    **pData, unsigned int *pSize);
    AT91S_BufferStatus (*EmptyWrBuffer)    (struct _AT91S_Buffer *pSBuffer, unsigned
    int size);
    AT91S_BufferStatus (*FillRdBuffer)     (struct _AT91S_Buffer *pSBuffer, unsigned
    int size);
    char (*IsWrEmpty)      (struct _AT91S_Buffer *pSBuffer);
    char (*IsRdFull)       (struct _AT91S_Buffer *pSBuffer);
} AT91S_Buffer, *AT91PS_Buffer;
```

### Description of the SvcComm Structure

The SvcComm structure provides the interface between low-level functions and the pipe object.

It contains pointers of functions initialized to the lower level functions (e.g. SvcXmodem).

The Xmodem Service implementation gives an example of SvcComm use.

```
typedef struct _AT91S_Service
{
    // Methods:
    AT91S_SvcCommStatus (*Reset) (struct _AT91S_Service *pService);
    AT91S_SvcCommStatus (*StartTx)(struct _AT91S_Service *pService);
    AT91S_SvcCommStatus (*StartRx)(struct _AT91S_Service *pService);
    AT91S_SvcCommStatus (*StopTx) (struct _AT91S_Service *pService);
    AT91S_SvcCommStatus (*StopRx) (struct _AT91S_Service *pService);
    char (*TxReady)(struct _AT91S_Service *pService);
    char (*RxReady)(struct _AT91S_Service *pService);
    // Data:
    struct _AT91S_Buffer *pBuffer; // Link to a buffer object
    void *pChild;
} AT91S_SvcComm, *AT91PS_SvcComm;
```



## Description of the SvcXmodem Structure

The SvcXmodem service is a reusable implementation of the Xmodem protocol. It supports only the 128-byte packet format and provides read and write functions. The SvcXmodem structure defines:

- a pointer to a handler initialized to readHandler or writeHandler
- a pointer to a function that processes the xmodem packet crc
- a pointer to a function that checks the packet header
- a pointer to a function that checks data

With this structure, the Xmodem protocol can be used with all media (USART, USB, etc.). Only private methods may be overloaded to adapt the Xmodem protocol to a new media.

The default implementation of the Xmodem uses a USART to send and receive packets. Read and write functions implement peripheral data controller facilities to reduce interrupt overhead. It assumes the USART is initialized, the memory buffer allocated and the interrupts programmed.

A periodic timer is required by the service to manage timeouts and the periodic transmission of the character "C" (Refer to Xmodem protocol). This feature is provided by the Tempo Service.

The following structure defines the Xmodem Service:

```
typedef struct _AT91PS_SvcXmodem {

    // Public Methods:
    AT91S_SvcCommStatus (*Handler) (struct _AT91PS_SvcXmodem *, unsigned int);
    AT91S_SvcCommStatus (*StartTx) (struct _AT91PS_SvcXmodem *, unsigned int);
    AT91S_SvcCommStatus (*StopTx) (struct _AT91PS_SvcXmodem *, unsigned int);

    // Private Methods:
    AT91S_SvcCommStatus (*ReadHandler) (struct _AT91PS_SvcXmodem *, unsigned int
csr);
    AT91S_SvcCommStatus (*WriteHandler) (struct _AT91PS_SvcXmodem *, unsigned int
csr);
    unsigned short      (*GetCrc)      (char *ptr, unsigned int count);
    char                (*CheckHeader) (unsigned char currentPacket, char *packet);
    char                (*CheckData)  (struct _AT91PS_SvcXmodem *);

    AT91S_SvcComm parent; // Base class
    AT91PS_USART pUsart;

    AT91S_SvcTempo tempo; // Link to a AT91S_Tempo object

    char *pData;
    unsigned int dataSize; // = XMODEM_DATA_STX or XMODEM_DATA_SOH
    char packetDesc[AT91C_XMODEM_PACKET_SIZE];
    unsigned char packetId; // Current packet
    char packetStatus;
    char isPacketDesc;
    char eot; // end of transmission
} AT91S_SvcXmodem, *AT91PS_SvcXmodem
```

## Xmodem Service Description

**Table 29.** Xmodem Service Methods

| Associated Function Pointers & Methods Used by Default  | Description  |
|---|--|
| <pre>// Typical Use: pAT91-&gt;OpenSvcXmodem(...);  // Default Method: AT91PS_SvcComm AT91F_OpenSvcXmodem(     AT91PS_SvcXmodem pSvcXmodem,     AT91PS_USART pUsart,     AT91PS_CtlTempo pCtlTempo)</pre>   | <p>Member of AT91S_RomBoot structure.<br/>Corresponds to the Open Service Method for the Xmodem Service.</p> <p><u>Input Parameters:</u><br/>Pointer on SvcXmodem structure.<br/>Pointer on a USART structure.<br/>Pointer on a CtlTempo structure.</p> <p><u>Output Parameters:</u><br/>Returns the Xmodem Service Pointer Structure.</p> |
| <pre>// Typical Use: AT91S_SvcXmodem svcXmodem; svcXmodem.Handler(...);  // Default read handler: AT91S_SvcCommStatus AT91F_SvcXmodemReadHandler(AT91PS_SvcXmodem     pSvcXmodem, unsigned int csr)  // Default write handler: AT91S_SvcCommStatus AT91F_SvcXmodemWriteHandler(AT91PS_SvcXmodem     pSvcXmodem, unsigned int csr)</pre> | <p>Member of AT91S_SvcXmodem structure.<br/>interrupt handler for xmodem read or write fonctionnalities</p> <p><u>Input Parameters:</u><br/>Pointer on a Xmodem Service Structure.<br/>csr: usart channel status register .</p> <p><u>Output Parameters:</u><br/>Status for xmodem read or write.</p>                                      |

## Using the Service

The following steps show how to initialize and use the Xmodem Service in an application:

Variables definitions:

```

AT91S_RomBoot const *pAT91; // struct containing Openservice functions
AT91S_SBuffer   sXmBuffer; // Xmodem Buffer allocation
AT91S_SvcXmodem svcXmodem; // Xmodem service structure allocation
AT91S_Pipe      xmodemPipe; // xmodem pipe communication struct
AT91S_CtlTempo  ctlTempo; // Tempo struct
AT91PS_Buffer  pXmBuffer; // Pointer on a buffer structure
AT91PS_SvcComm pSvcXmodem; // Pointer on a Media Structure

Initialisations
// Call Open methods:
pAT91 = AT91C_ROM_BOOT_ADDRESS;
// OpenCtlTempo on the system timer
pAT91->OpenCtlTempo(&ctlTempo, (void *) &(pAT91->SYSTIMER_DESC));
ctlTempo.CtlTempoStart((void *) &(pAT91->SYSTIMER_DESC));
// Xmodem buffer initialisation
pXmBuffer      = pAT91->OpenSBuffer(&sXmBuffer);
pSvcXmodem     = pAT91->OpenSvcXmodem(&svcXmodem, AT91C_BASE_DBGU, &ctlTempo);
// Open communication pipe on the xmodem service
pAT91->OpenPipe(&xmodemPipe, pSvcXmodem, pXmBuffer);
// Init the DBGU peripheral
// Open PIO for DBGU
AT91F_DBGU_CfgPIO();
// Configure DBGU
AT91F_US_Configure (
    (AT91PS_USART) AT91C_BASE_DBGU, // DBGU base address
    MCK, // Master Clock
    AT91C_US_ASYNC_MODE, // mode Register to be programmed
    BAUDRATE, // baudrate to be programmed
    0); // timeguard to be programmed
// Enable Transmitter
AT91F_US_EnableTx((AT91PS_USART) AT91C_BASE_DBGU);
// Enable Receiver
AT91F_US_EnableRx((AT91PS_USART) AT91C_BASE_DBGU);
// Initialize the Interrupt for System Timer and DBGU (shared interrupt)
// Initialize the Interrupt Source 1 for SysTimer and DBGU
AT91F_AIC_ConfigureIt(AT91C_BASE_AIC,
    AT91C_ID_SYS,
    AT91C_AIC_PRIOR_HIGHEST,
    AT91C_AIC_SRCTYPE_INT_LEVEL_SENSITIVE,
    AT91F_ASM_ST_DBGU_Handler);

// Enable SysTimer and DBGU interrupt
AT91F_AIC_EnableIt(AT91C_BASE_AIC, AT91C_ID_SYS);

xmodemPipe.Read(&xmodemPipe, (char *) BASE_LOAD_ADDRESS, MEMORY_SIZE,
XmodemProtocol, (void *) BASE_LOAD_ADDRESS);

```

## DataFlash Service

### Presentation

The DataFlash Service allows the Serial Peripheral Interface (SPI) to support several Serial DataFlash and DataFlash Cards for reading, programming and erasing operations.

This service is based on SPI interrupts that are managed by a specific handler. It also uses the corresponding PDC registers.

For more information on the commands available in the DataFlash Service, refer to the relevant DataFlash documentation.

### DataFlash Service Description

**Table 30.** DataFlash Service Methods

| Associated Function Pointers & Methods Used by Default  | Description   |
|---|---|
| <pre>// Typical Use: pAT91-&gt;OpenSvcDataFlash(...);  // Default Method: AT91PS_SvcDataFlash AT91F_OpenSvcDataFlash ( const AT91PS_PMC pApmc, AT91PS_SvcDataFlash pSvcDataFlash)</pre>         | <p>Member of <code>AT91S_RomBoot</code> structure.</p> <p>Corresponds to the Open Service Method for the DataFlash Service.</p> <p><u>Input Parameters:</u></p> <p>Pointer on a PMC Register Description Structure.</p> <p>Pointer on a DataFlash Service Structure.</p> <p><u>Output Parameters:</u></p> <p>Returns the DataFlash Service Pointer Structure.</p>   |
| <pre>// Typical Use: AT91S_SvcDataFlash svcDataFlash; svcDataFlash.Handler(...);  // Default Method: void AT91F_DataFlashHandler( AT91PS_SvcDataFlash pSvcDataFlash, unsigned int status)</pre> | <p>Member of <code>AT91S_SvcDataFlash</code> structure.</p> <p>SPI Fixed Peripheral C interrupt handler.</p> <p><u>Input Parameters:</u></p> <p>Pointer on a DataFlash Service Structure.</p> <p>Status: corresponds to the interruptions detected and validated on SPI (SPI Status Register masked by SPI Mask Register).</p> <p>Has to be put in the Interrupt handler for SPI.</p> <p><u>Output Parameters:</u></p> <p>None.</p> |
| <pre>// Typical Use: AT91S_SvcDataFlash svcDataFlash; svcDataFlash.Status(...);  // Default Method: AT91S_SvcDataFlashStatus AT91F_DataFlashGetStatus(AT91PS_DataflashDesc pDesc)</pre>         | <p>Member of <code>AT91S_SvcDataFlash</code> structure.</p> <p>Read the status register of the DataFlash.</p> <p><u>Input Parameters:</u></p> <p>Pointer on a DataFlash Descriptor Structure (member of the service structure).</p> <p><u>Output Parameters:</u></p> <p>Returns 0 if DataFlash is Busy.</p> <p>Returns 1 if DataFlash is Ready.</p>   |
| <pre>// Typical Use: AT91S_SvcDataFlash svcDataFlash; svcDataFlash.AbortCommand(...);  // Default Method: void AT91F_DataFlashAbortCommand(AT91PS_DataflashDesc pDesc)</pre>                    | <p>Member of <code>AT91S_SvcDataFlash</code> structure</p> <p>Allows to reset PDC &amp; Interrupts.</p> <p><u>Input Parameters:</u></p> <p>Pointer on a DataFlash Descriptor Structure (member of the service structure).</p> <p><u>Output Parameters:</u></p> <p>None.</p>   |

**Table 30. DataFlash Service Methods (Continued)**

| Associated Function Pointers & Methods Used by Default  | Description  |
|---|--|
| <pre>// Typical Use: AT91S_SvcDataFlash svcDataFlash; svcDataFlash.<b>PageRead</b>(...);  // Default Method: AT91S_SvcDataFlashStatus AT91F_DataFlashPageRead ( AT91PS_SvcDataFlash pSvcDataFlash, unsigned int src, unsigned char *dataBuffer, int sizeToRead )</pre>  | <p>Member of AT91S_SvcDataFlash structure<br/>Read a Page in DataFlash.</p> <p><u>Input Parameters:</u><br/>Pointer on DataFlash Service Structure.<br/>DataFlash address.<br/>Data buffer destination pointer.<br/>Number of bytes to read.</p> <p><u>Output Parameters:</u><br/>Returns 0 if DataFlash is Busy.<br/>Returns 1 if DataFlash Ready.</p>  |
| <pre>// Typical Use: AT91S_SvcDataFlash svcDataFlash; svcDataFlash.<b>ContinuousRead</b>(...);  // Default Method: AT91S_SvcDataFlashStatus AT91F_DataFlashContinuousRead ( AT91PS_SvcDataFlash pSvcDataFlash, int src, unsigned char *dataBuffer, int sizeToRead )</pre>   | <p>Member of AT91S_SvcDataFlash structure.<br/>Continuous Stream Read.</p> <p><u>Input Parameters:</u><br/>Pointer on DataFlash Service Structure.<br/>DataFlash address.<br/>Data buffer destination pointer.<br/>Number of bytes to read.</p> <p><u>Output Parameters:</u><br/>Returns 0 if DataFlash is Busy.<br/>Returns 1 if DataFlash is Ready.</p>  |
| <pre>// Typical Use: AT91S_SvcDataFlash svcDataFlash; svcDataFlash.<b>ReadBuffer</b>(...);  // Default Method: AT91S_SvcDataFlashStatus AT91F_DataFlashReadBuffer ( AT91PS_SvcDataFlash pSvcDataFlash, unsigned char BufferCommand, unsigned int bufferAddress, unsigned char *dataBuffer, int sizeToRead )</pre> | <p>Member of AT91S_SvcDataFlash structure.<br/>Read the Internal DataFlash SRAM Buffer 1 or 2.</p> <p><u>Input Parameters:</u><br/>Pointer on DataFlash Service Structure.<br/>Choose Internal DataFlash Buffer 1 or 2 command.<br/>DataFlash address.<br/>Data buffer destination pointer.<br/>Number of bytes to read.</p> <p><u>Output Parameters:</u><br/>Returns 0 if DataFlash is Busy.<br/>Returns 1 if DataFlash is Ready.<br/>Returns 4 if DataFlash Bad Command.<br/>Returns 5 if DataFlash Bad Address.</p> |
| <pre>// Typical Use: AT91S_SvcDataFlash svcDataFlash; svcDataFlash.<b>MainMemoryToBufferTransfert</b>(...);  // Default Method: AT91S_SvcDataFlashStatus AT91F_MainMemoryToBufferTransfert( AT91PS_SvcDataFlash pSvcDataFlash, unsigned char BufferCommand, unsigned int page)</pre>                              | <p>Member of AT91S_SvcDataFlash structure<br/>Read a Page in the Internal SRAM Buffer 1 or 2.</p> <p><u>Input Parameters:</u><br/>Pointer on DataFlash Service Structure.<br/>Choose Internal DataFlash Buffer 1 or 2 command.<br/>Page to read.</p> <p><u>Output Parameters:</u><br/>Returns 0 if DataFlash is Busy.<br/>Returns 1 if DataFlash is Ready.<br/>Returns 4 if DataFlash Bad Command.</p>   |



**Table 30. DataFlash Service Methods (Continued)**

| Associated Function Pointers & Methods Used by Default  | Description  |
|---|--|
| <pre>// Typical Use: AT91S_SvcDataFlash svcDataFlash; svcDataFlash.<b>PageRead</b>(...);  // Default Method: AT91S_SvcDataFlashStatus AT91F_DataFlashPageRead ( AT91PS_SvcDataFlash pSvcDataFlash, unsigned int src, unsigned char *dataBuffer, int sizeToRead )</pre>  | <p>Member of AT91S_SvcDataFlash structure<br/>Read a Page in DataFlash.</p> <p><u>Input Parameters:</u><br/>Pointer on DataFlash Service Structure.<br/>DataFlash address.<br/>Data buffer destination pointer.<br/>Number of bytes to read.</p> <p><u>Output Parameters:</u><br/>Returns 0 if DataFlash is Busy.<br/>Returns 1 if DataFlash Ready.</p>  |
| <pre>// Typical Use: AT91S_SvcDataFlash svcDataFlash; svcDataFlash.<b>ContinuousRead</b>(...);  // Default Method: AT91S_SvcDataFlashStatus AT91F_DataFlashContinuousRead ( AT91PS_SvcDataFlash pSvcDataFlash, int src, unsigned char *dataBuffer, int sizeToRead )</pre>   | <p>Member of AT91S_SvcDataFlash structure.<br/>Continuous Stream Read.</p> <p><u>Input Parameters:</u><br/>Pointer on DataFlash Service Structure.<br/>DataFlash address.<br/>Data buffer destination pointer.<br/>Number of bytes to read.</p> <p><u>Output Parameters:</u><br/>Returns 0 if DataFlash is Busy.<br/>Returns 1 if DataFlash is Ready.</p>  |
| <pre>// Typical Use: AT91S_SvcDataFlash svcDataFlash; svcDataFlash.<b>ReadBuffer</b>(...);  // Default Method: AT91S_SvcDataFlashStatus AT91F_DataFlashReadBuffer ( AT91PS_SvcDataFlash pSvcDataFlash, unsigned char BufferCommand, unsigned int bufferAddress, unsigned char *dataBuffer, int sizeToRead )</pre> | <p>Member of AT91S_SvcDataFlash structure.<br/>Read the Internal DataFlash SRAM Buffer 1 or 2.</p> <p><u>Input Parameters:</u><br/>Pointer on DataFlash Service Structure.<br/>Choose Internal DataFlash Buffer 1 or 2 command.<br/>DataFlash address.<br/>Data buffer destination pointer.<br/>Number of bytes to read.</p> <p><u>Output Parameters:</u><br/>Returns 0 if DataFlash is Busy.<br/>Returns 1 if DataFlash is Ready.<br/>Returns 4 if DataFlash Bad Command.<br/>Returns 5 if DataFlash Bad Address.</p> |
| <pre>// Typical Use: AT91S_SvcDataFlash svcDataFlash; svcDataFlash.<b>MainMemoryToBufferTransfert</b>(...);  // Default Method: AT91S_SvcDataFlashStatus AT91F_MainMemoryToBufferTransfert( AT91PS_SvcDataFlash pSvcDataFlash, unsigned char BufferCommand, unsigned int page)</pre>                              | <p>Member of AT91S_SvcDataFlash structure<br/>Read a Page in the Internal SRAM Buffer 1 or 2.</p> <p><u>Input Parameters:</u><br/>Pointer on DataFlash Service Structure.<br/>Choose Internal DataFlash Buffer 1 or 2 command.<br/>Page to read.</p> <p><u>Output Parameters:</u><br/>Returns 0 if DataFlash is Busy.<br/>Returns 1 if DataFlash is Ready.<br/>Returns 4 if DataFlash Bad Command.</p>   |

**Table 30. DataFlash Service Methods (Continued)**

| Associated Function Pointers & Methods Used by Default   | Description  |
|--|--|
| <pre>// Typical Use: AT91S_SvcDataFlash svcDataFlash; svcDataFlash.<b>PagePgmBuf</b>(...);  // Default Method: AT91S_SvcDataFlashStatus AT91F_DataFlashPagePgmBuf( AT91PS_SvcDataFlash pSvcDataFlash, unsigned char BufferCommand, unsigned char *src, unsigned int dest, unsigned int SizeToWrite)</pre>            | <p>Member of <code>AT91S_SvcDataFlash</code> structure<br/> Page Program through Internal SRAM Buffer 1 or 2.</p> <p><u>Input Parameters:</u><br/> Pointer on DataFlash Service Structure.<br/> Choose Internal DataFlash Buffer 1 or 2 command.<br/> Source buffer.<br/> DataFlash destination address.<br/> Number of bytes to write.</p> <p><u>Output Parameters:</u><br/> Returns 0 if DataFlash is Busy.<br/> Returns 1 if DataFlash is Ready.<br/> Returns 4 if DataFlash Bad Command.</p>   |
| <pre>// Typical Use: AT91S_SvcDataFlash svcDataFlash; svcDataFlash.<b>WriteBuffer</b>(...);  // Default Method: AT91S_SvcDataFlashStatus AT91F_DataFlashWriteBuffer ( AT91PS_SvcDataFlash pSvcDataFlash, unsigned char BufferCommand, unsigned char *dataBuffer, unsigned int bufferAddress, int SizeToWrite )</pre> | <p>Member of <code>AT91S_SvcDataFlash</code> structure.<br/> Write data to the Internal SRAM buffer 1 or 2.</p> <p><u>Input Parameters:</u><br/> Pointer on DataFlash Service Structure.<br/> Choose Internal DataFlash Buffer 1 or 2 command.<br/> Pointer on data buffer to write.<br/> Address in the internal buffer.<br/> Number of bytes to write.</p> <p><u>Output Parameters:</u><br/> Returns 0 if DataFlash is Busy.<br/> Returns 1 if DataFlash is Ready.<br/> Returns 4 if DataFlash Bad Command.<br/> Returns 5 if DataFlash Bad Address.</p> |
| <pre>// Typical Use: AT91S_SvcDataFlash svcDataFlash; svcDataFlash.<b>WriteBufferToMain</b>(...);  // Default Method: AT91S_SvcDataFlashStatus AT91F_WriteBufferToMain ( AT91PS_SvcDataFlash pSvcDataFlash, unsigned char BufferCommand, unsigned int dest )</pre>   | <p>Member of <code>AT91S_SvcDataFlash</code> structure.<br/> Write Internal Buffer to the DataFlash Main Memory.</p> <p><u>Input Parameters:</u><br/> Pointer on DataFlash Service Structure.<br/> Choose Internal DataFlash Buffer 1 or 2 command.<br/> Main memory address on DataFlash.</p> <p><u>Output Parameters:</u><br/> Returns 0 if DataFlash is Busy.<br/> Returns 1 if DataFlash is Ready.</p>   |

**Table 30.** DataFlash Service Methods (Continued)

| Associated Function Pointers & Methods Used by Default   | Description  |
|--|--|
| <pre>// Typical Use: AT91S_SvcDataFlash svcDataFlash; svcDataFlash.<b>PageErase</b>(...);  // Default Method: AT91S_SvcDataFlashStatus AT91F_PageErase ( AT91PS_SvcDataFlash pSvcDataFlash, unsigned int PageNumber)</pre>   | <p>Member of AT91S_SvcDataFlash structure.<br/>Erase a page in DataFlash.</p> <p><i>Input Parameters:</i><br/>Pointer on a Service DataFlash Object.<br/>Page to erase.</p> <p><i>Output Parameters:</i><br/>Returns 0 if DataFlash is Busy.<br/>Returns 1 if DataFlash Ready.</p>   |
| <pre>// Typical Use: AT91S_SvcDataFlash svcDataFlash; svcDataFlash.<b>BlockErase</b>(...);  // Default Method: AT91S_SvcDataFlashStatus AT91F_BlockErase ( AT91PS_SvcDataFlash pSvcDataFlash, unsigned int BlockNumber )</pre>   | <p>Member of AT91S_SvcDataFlash structure.<br/>Erase a block of 8 pages.</p> <p><i>Input Parameters:</i><br/>Pointer on a Service DataFlash Object.<br/>Block to erase.</p> <p><i>Output Parameters:</i><br/>Returns 0 if DataFlash is Busy.<br/>Returns 1 if DataFlash Ready.</p>   |
| <pre>// Typical Use: AT91S_SvcDataFlash svcDataFlash; svcDataFlash.<b>MainMemoryToBufferCompare</b>(...);  // Default Method: AT91S_SvcDataFlashStatus AT91F_MainMemoryToBufferCompare( AT91PS_SvcDataFlash pSvcDataFlash, unsigned char BufferCommand, unsigned int page)</pre> | <p>Member of AT91S_SvcDataFlash structure.<br/>Compare the contents of a Page and one of the Internal SRAM buffer.</p> <p><i>Input Parameters:</i><br/>Pointer on a Service DataFlash Object.<br/>Internal SRAM DataFlash Buffer to compare command.<br/>Page to compare.</p> <p><i>Output Parameters:</i><br/>Returns 0 if DataFlash is Busy.<br/>Returns 1 if DataFlash Ready.<br/>Returns 4 if DataFlash Bad Command.</p> |

Note: AT91S\_SvcDataFlashStatus corresponds to an unsigned int.

## Using the Service

The first step is to find the address of the open service method `AT91F_OpenSvcDataFlash` using the ROM Entry Service.

1. Allocate one instance of `AT91S_SvcDataFlash` and `AT91S_Dataflash` in the application memory space:

```
// Allocate the service and a device structure.
AT91S_SvcDataFlash svcDataFlash;
AT91S_Dataflash Device; // member of AT91S_SvcDataFlash service
```

Then initialize the `AT91S_SvcDataFlash` instance by calling the `AT91F_OpenSvcDataFlash` function:

```
// Initialize service
pAT91->OpenSvcDataFlash (AT91C_BASE_PMC, &svcDataFlash);
```

2. Initialize the SPI Interrupt:

```
// Initialize the SPI Interrupt
at91_irq_open ( AT91C_BASE_AIC, AT91C_ID_SPI, 3,
               AT91C_AIC_SRCTYPE_INT_LEVEL_SENSITIVE , AT91F_spi_asm_handler);
```

3. Configure the DataFlash structure with its correct features and link it to the device structure in the `AT91S_SvcDataFlash` service structure:

```
// Example with an ATMEL AT45DB321B DataFlash
Device.pages_number = 8192;
Device.pages_size = 528;
Device.page_offset = 10;
Device.byte_mask = 0x300;
// Link to the service structure
svcDataFlash.pDevice = &Device;
```

4. Now the different methods can be used. Following is an example of a Page Read of 528 bytes on page 50:

```
// Result of the read operation in RxBufferDataFlash
unsigned char RxBufferDataFlash[528];
svcDataFlash.PageRead(&svcDataFlash,
                    (50*528), RxBufferDataFlash, 528);
```

## CRC Service

### Presentation

This “service” differs from the preceding ones in that it is structured differently: it is composed of an array and some methods directly accessible via the `AT91S_RomBoot` structure.

### CRC Service Description

**Table 31.** CRC Service Description

| Methods and Array Available   | Description  |
|---|--|
| <pre>// Typical Use: pAT91-&gt;CRC32(...);  // Default Method: void CalculateCrc32( const unsigned char *address, unsigned int size, unsigned int *crc)</pre>           | <p>This function provides a table driven 32bit CRC generation for byte data. This CRC is known as the CCITT CRC32.</p> <p><u>Input Parameters:</u><br/>           Pointer on the data buffer.<br/>           The size of this buffer.<br/>           A pointer on the result of the CRC.</p> <p><u>Output Parameters:</u><br/>           None.</p>                     |
| <pre>// Typical Use: pAT91-&gt;CRC16(...);  // Default Method: void CalculateCrc16( const unsigned char *address, unsigned int size, unsigned short *crc)</pre>         | <p>This function provides a table driven 16bit CRC generation for byte data. This CRC is calculated with the POLYNOME 0x8005</p> <p><u>Input Parameters:</u><br/>           Pointer on the data buffer.<br/>           The size of this buffer.<br/>           A pointer on the result of the CRC.</p> <p><u>Output Parameters:</u><br/>           None.</p>           |
| <pre>// Typical Use: pAT91-&gt;CRCHDLC(...);  // Default Method: void CalculateCrcHdlc( const unsigned char *address, unsigned int size, unsigned short *crc)</pre>     | <p>This function provides a table driven 16bit CRC generation for byte data. This CRC is known as the HDLC CRC.</p> <p><u>Input Parameters:</u><br/>           Pointer on the data buffer.<br/>           The size of this buffer.<br/>           A pointer on the result of the CRC.</p> <p><u>Output Parameters:</u><br/>           None.</p>                        |
| <pre>// Typical Use: pAT91-&gt;CRCCCITT(...);  // Default Method: void CalculateCrc16ccitt( const unsigned char *address, unsigned int size, unsigned short *crc)</pre> | <p>This function provides a table driven 16bit CRC generation for byte data. This CRC is known as the CCITT CRC16 (POLYNOME = 0x1021).</p> <p><u>Input Parameters:</u><br/>           Pointer on the data buffer.<br/>           The size of this buffer.<br/>           A pointer on the result of the CRC.</p> <p><u>Output Parameters:</u><br/>           None.</p> |
| <pre>// Typical Use: char reverse_byte; reverse_byte = pAT91-&gt;Bit_Reverse_Array[...];  // Array Embedded: const unsigned char bit_rev[256]</pre>                     | <p>Bit Reverse Array: array which allows to reverse one octet. Frequently used in mathematical algorithms.</p> <p>Used for example in the CRC16 calculation.</p>   |

## Using the Service

Compute the CRC16 CCITT of a 256-byte buffer and save it in the crc16 variable:

```
// Compute CRC16 CCITT
unsigned char BufferToCompute[256];
short crc16;
... (BufferToCompute Treatment)
pAT91->CRCCITT(&BufferToCompute, 256, &crc16);
```

## Sine Service

### Presentation

This “service” differs from the preceding one in that it is structured differently: it is composed of an array and a method directly accessible through the `AT91S_RomBoot` structure.

### Sine Service Description

**Table 32.** Sine Service Description

| Method and Array Available  | Description   |
|---|---|
| <pre>// Typical Use: pAT91-&gt;Sine(...);  // Default Method: short AT91F_Sinus(int step)</pre>                                 | <p>This function returns the amplitude coded on 16 bits, of a sine waveform for a given step.</p> <p><u>Input Parameters:</u><br/>Step of the sine. Corresponds to the precision of the amplitude calculation. Depends on the Sine Array used. Here, the array has 256 values (thus 256 steps) of amplitude for 180 degrees.</p> <p><u>Output Parameters:</u><br/>Amplitude of the sine waveform.</p> |
| <pre>// Typical Use: short sinus; sinus = pAT91-&gt;SineTab[...];  // Array Embedded: const short AT91C_SINUS180_TAB[256]</pre> | <p>Sine Array with a resolution of 256 values for 180 degrees.</p>  |

# AT91RM9200 Reset Controller

## Overview

This chapter describes the AT91RM9200 reset signals and how to use them in order to assure correct operation of the device.

The AT91RM9200 has two reset input lines called NRST and NTRST. Each line provides, respectively:

- Initialization of the User Interface registers (defined in the user interface of each peripheral) and:
  - Sample the signals needed at bootup
  - Compel the processor to fetch the next instruction at address zero.
- Initialization of the embedded ICE TAP controller.

The NRST signal must be considered as the System Reset signal and the reader must take care when designing the logic to drive this reset signal. NTRST is typically used by the hardware debug interface which uses the In-Circuit Emulator unit and Initializes it without affecting the normal operation of the ARM® processor. This line shall also be driven by an on board logic.

Both NRST and NTRST are active low signals that asynchronously reset the logic in the AT91RM9200.

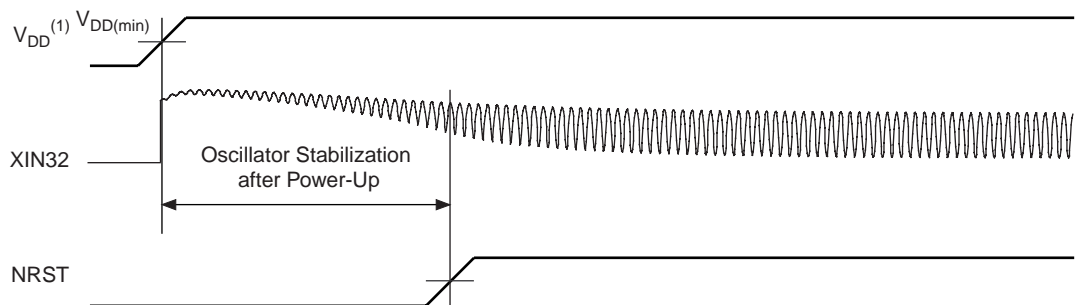
## Reset Conditions

### NRST Conditions

NRST is the active low reset input. When power is first applied to the system, a power-on reset (also denominated as “cold” reset) must be applied to the AT91RM9200. During this transient state, it is mandatory to hold the reset signal low long enough for the power supply to reach a working nominal level and for the oscillator to reach a stable operating frequency. Typically, these features are provided by every power supply supervisor which, under a threshold voltage limit, the electrical environment is considered as not nominal. Power-up is not the only event to be considered as power-down or a brownout are also occurrences that assert the NRST signal. The threshold voltage must be selected according to the minimum operating voltage of the AT91RM9200 power supply lines marked as VDD in Figure 25. (See “DC Characteristics” on page 596.)

The choice of the reset holding delay depends on the start-up time of the low frequency oscillator as shown below in Figure 25. (See “32 kHz Oscillator Characteristics” on page 599.)

**Figure 25.** Cold Reset and Oscillator Start-up relationship



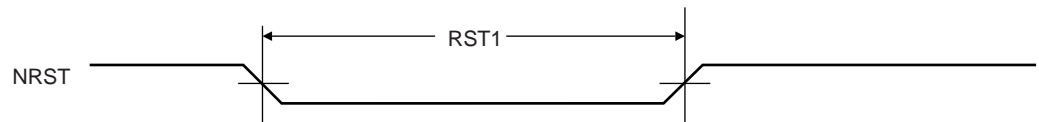
Note: 1. VDD is applicable to VDDIOM, VDDIOP, VDDPLL, VDDOSC and VDDCORE

NRST can also be asserted in circumstances other than the power-up sequence, such as a manual command. This assertion can be performed asynchronously, but exit from reset is synchronized internally to the default active clock. During normal operation, NRST must be active for a minimum delay time to ensure correct behavior. See Figure 26 and Table 33.

**Table 33.** Reset Minimum Pulse Width

| Symbol | Parameter                | Min. Pulse Width | Unit |
|--------|--------------------------|------------------|------|
| RST1   | NRST Minimum Pulse Width | 92               | μs   |

**Figure 26.** NRST assertion



### NTRST Assertion

As with the NRST signal, at power-up, the NTRST signal must be valid while the power supply has not obtained the the minimum recommended working level. (See “DC Characteristics” on page 596.). A clock on TCK is not required to validate this reset request.

As with the NRST signal, NTRST can also be asserted in circumstances other than the power-up sequence, such as a manual command or an ICE Interface action. This assertion and de-assertion can be performed asynchronously but must be active for a minimum delay time. (See “JTAG/ICE Timings” on page 621.)

## Reset Management

### System Reset

The system reset functionality is provided through the NRST signal.

This Reset signal is used to compel the microcontroller unit to assume a set of initial conditions:

- Sample the Boot Mode Select (BMS) logical state.
- Restore the default states (default values) of the user interface.
- Require the processor to perform the next instruction fetch from address zero.

With the exception of the program counter and the Current Program Status Register, the processor’s registers do not have defined reset states. When the microcontroller’s NRST input is asserted, the processor immediately stops execution of the current instruction independently of the clock.

The system reset circuitry must take two types of reset requests into account:

- The cold reset needed for the power-up sequence.
- The user reset request.

Both have the same effect but can have different assertion time requirements regarding the NRST pin. In fact, the cold reset assertion has to overlap the start-up time of the system. The user reset request requires a shorter assertion delay time than does cold reset.

### Test Reset

Test reset functionality is provided through the NTRST signal.



The NTRST control pin initializes the selected TAP controller. The TAP controller involved in this reset is determined according to the initial logical state applied on the JTAGSEL pin after the last valid NRST.

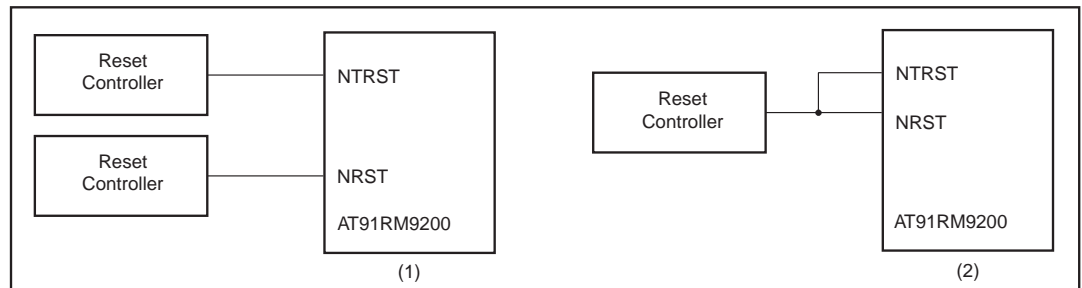
In Boundary Scan Mode, after a NTRST assertion, the IDCODE instruction is set onto the output of the instruction register in the Test-Logic-Reset controller state.

Otherwise, in ICE Mode, the reset action is as follows:

- The core exits from Debug Mode.
- The IDCORE instruction is requested.

In either Boundary Scan or ICE Mode a reset can be performed from the same or different circuitry, as shown in Figure 27 below, upon system reset at power-up or upon user request.

**Figure 27. Separate or Common Reset Management**



- Notes:
1. NRST and NTRST handling in Debug Mode during development.
  2. NRST and NTRST handling during production.

In order to benefit the most regarding the separation of NRST and NTRST during the Debug phase of development, the user must independently manage both signals as shown in example (1) of Figure 27 above. However, once Debug is completed, both signals are easily managed together during production as shown in example (2) of Figure 27 above.

## Required Features for the Reset Controller

The following table presents the features required of a reset controller in order to obtain an optimal system with the AT91RM9200 processor.

**Table 34. Reset Controller Functions Synthesis**

| Feature                     | Description   |
|-----------------------------|---|
| Power Supply Monitoring     | Overlaps the transient state of the system during power-up/down and brownout.   |
| Reset Active Timeout Period | Overlaps the start-up time of the boot-up oscillator by holding the reset signal during this delay.   |
| Manual Reset Command        | Asserts the reset signal from a logic command and holds the reset signal with a shorter delay than that of the "Reset Active Timeout Period". |



## Memory Controller(MC)

### Overview

The Memory Controller (MC) manages the ASB bus and controls access by up to four masters. It features a bus arbiter and an address decoder that splits the 4G bytes of address space into areas to access the embedded SRAM and ROM, the embedded peripherals and the external memories through the External Bus Interface (EBI). It also features an abort status and a misalignment detector to assist in application debug.

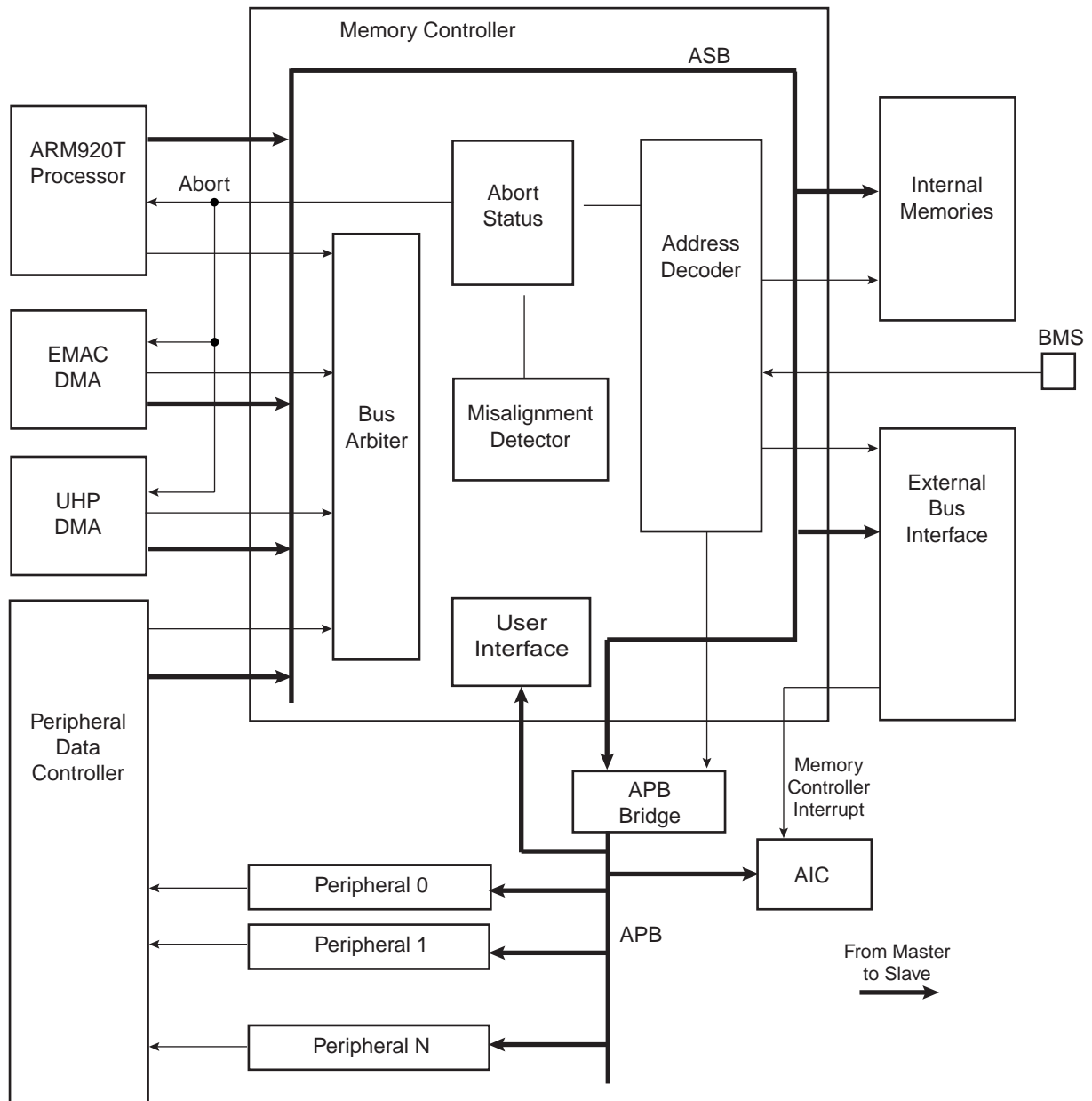
The Memory Controller allows booting from the embedded ROM or from an external non-volatile memory connected to the Chip Select 0 of the EBI. The Remap command switches addressing of the ARM vectors (0x0 - 0x20) on the embedded SRAM.

Key Features of the RM9200 Memory Controller are:

- Programmable Bus Arbiter Handling Four Masters
  - Internal Bus is Shared by ARM920T, PDC, USB Host Port and Ethernet MAC Masters
  - Each Master Can Be Assigned a Priority Between 0 and 7
- Address Decoder Provides Selection For
  - Eight External 256-Mbyte Memory Areas
  - Four Internal 1-Mbyte Memory Areas
  - One 256-Mbyte Embedded Peripheral Area
- Boot Mode Select Option
  - Non-volatile Boot Memory Can Be Internal or External
  - Selection is Made By BMS Pin Sampled at Reset
- Abort Status Registers
  - Source, Type and All Parameters of the Access Leading to an Abort are Saved
- Misalignment Detector
  - Alignment Checking of All Data Accesses
  - Abort Generation in Case of Misalignment
- Remap Command
  - Provides Remapping of an Internal SRAM in Place of the Boot NVM

# Block Diagram

Figure 28. Memory Controller Block Diagram



## Functional Description

The Memory Controller (MC) handles the internal ASB bus and arbitrates the accesses of up to four masters.

It is made up of:

- A bus arbiter
- An address decoder
- An abort status
- A misalignment detector

The Memory Controller handles only little-endian mode accesses. All masters must work in little-endian mode only.

## Bus Arbiter

The Memory Controller has a user-programmable bus arbiter. Each master can be assigned a priority between 0 and 7, where 7 is the highest level. The bus arbiter is programmed in the register MC\_MPR (Master Priority Register).

The same priority level can be assigned to more than one master. If requests occur from two masters having the same priority level, the following default priority is used by the bus arbiter to determine the first to serve: Master 0, Master 1, Master 2, Master 3.

The masters are:

- the ARM920T as the Master 0
- the Peripheral Data Controller as the Master 1
- the USB Host Port as the Master 2
- the Ethernet MAC as the Master 3

## Address Decoder

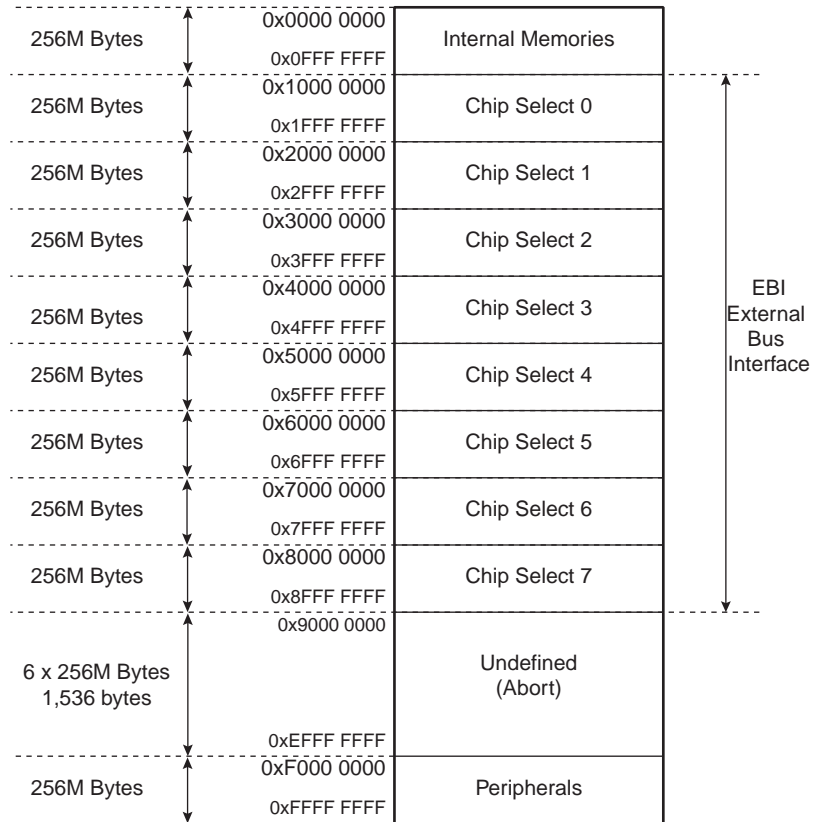
The Memory Controller features an Address Decoder that first decodes the four highest bits of the 32-bit address bus and defines 11 separate areas:

- One 256-Mbyte address space for the internal memories
- Eight 256-Mbyte address spaces, each assigned to one of the eight chip select lines of the External Bus Interface
- One 256-Mbyte address space reserved for the embedded peripherals
- An undefined address space of 1536M bytes that returns an Abort if accessed

## External Memory Areas

Figure 29 shows the assignment of the 256-Mbyte memory areas.

**Figure 29.** External Memory Areas



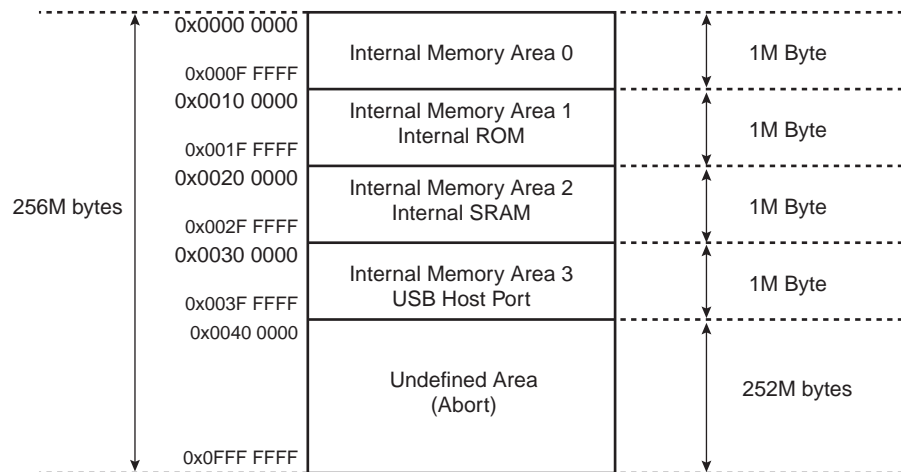
## Internal Memory Mapping

Within the Internal Memory address space, the Address Decoder of the Memory Controller decodes eight more address bits to allocate 1-Mbyte address spaces for the embedded memories.

The allocated memories are accessed all along the 1-Mbyte address space and so are repeated n times within this address space, n equaling 1M byte divided by the size of the memory.

When the address of the access is undefined within the internal memory area, i.e. over the address 0x0040 0000, the Address Decoder returns an Abort to the master.

**Figure 30.** Internal Memory Mapping After Remap



## Internal Memory Area 0

Depending on the BMS pin state at reset and as a function of the remap command, the memory mapped at address 0x0 is different. Before execution of the remap command the on-chip ROM (BMS = 1) or the non-volatile memory connected to external chip select zero (BMS = 0) is mapped into Internal Memory Area 0. After the remap command, the internal SRAM at address 0x0020 0000 is mapped into Internal Memory Area 0. The memory mapped into Internal Memory Area 0 is accessible in both its original location and at address 0x0.

The first 32 bytes of Internal Memory Area 0 contain the ARM processor exception vectors.

**Table 35.** Internal Memory Area Depending on BMS and the Remap Command

| BMS State              | Before Remap |                        | After Remap   |
|------------------------|--------------|------------------------|---------------|
|                        | 1            | 0                      | X             |
| Internal Memory Area 0 | Internal ROM | External Memory Area 0 | Internal SRAM |

## Boot Mode Select

The BMS pin state allows the device to boot out of an internal or external boot memory, actually the input level on the BMS pin during the last 2 clock cycles, before the reset selects the type of boot memory according to the following conditions:

- If high, the Internal ROM, which is generally mapped within the Internal Memory Area 1, is also accessible through the Internal Memory Area 0
- If low, the External Memory Area 0, which is generally accessible from address 0x1000 0000, is also accessible through the Internal Memory Area 0.

The BMS pin is multiplexed with an I/O line. After reset, this pin can be used as any standard PIO line.

## Remap Command

After execution, the Remap Command causes the Internal SRAM to be accessed through the Internal Memory Area 0.

As the ARM vectors (Reset, Abort, Data Abort, Prefetch Abort, Undefined Instruction, Interrupt, and Fast Interrupt) are mapped from address 0x0 to address 0x20, the Remap Command allows the user to redefine dynamically these vectors under software control.

The Remap Command is accessible through the Memory Controller User Interface by writing the MC\_RCR (Remap Control Register) RCB field to one.

The Remap Command can be cancelled by writing the MC\_RCR RCB field to one, which acts as a toggling command. This allows easy debug of the user-defined boot sequence by offering a simple way to put the chip in the same configuration as just after a reset.

Table 35 on page 127 is provided to summarize the effect of these two key features on the nature of the memory mapped to the address 0x0.

## Abort Status

There are two reasons for an abort to occur:

- an access to an undefined address
- an access to a misaligned address.

When an abort occurs, a signal is sent back to all the masters, regardless of which one has generated the access. However, only the master having generated the access leading to the abort takes this signal into account.

The abort signal generates directly an abort on the ARM9TDMI. Note that, from the processor perspective, an abort can also be generated by the Memory Management Unit of the ARM920T, but this is obviously not managed by the Memory Controller and not discussed in this section.

The Peripheral Data Controller does not handle the abort input signal (and that's why the connection is not represented in Figure 28). The UHP reports an unrecoverable error in the HcInterruptStatus register and resets its operations. The EMAC reports the Abort to the user through the ABT bit in its Status Register, which might generate an interrupt.

To facilitate debug or for fault analysis by an operating system, the Memory Controller integrates an Abort Status register set.

The full 32-bit wide abort address is saved in the Abort Address Status Register (MC\_AASR). Parameters of the access are saved in the Abort Status Register (MC\_ASR) and include:

- the size of the request (ABTSZ field)
- the type of the access, whether it is a data read or write or a code fetch (ABTTYP field)
- whether the access is due to accessing an undefined address (UNDADD bit) or a misaligned address (MISADD bit)
- the source of the access leading to the last abort (MST0, MST1, MST2 and MST3 bits)
- whether or not an abort occurred for each master since the last read of the register (SVMST0, SVMST1, SVMST2 and SVMST3 bits) except if it is traced in the MST bits.

In case of Data Abort from the processor, the address of the data access is stored. This is probably the most useful, as finding which address has generated the abort would require disassembling the instruction and full knowledge of the processor context.

However, in case of prefetch abort, the address might have changed, as the prefetch abort is pipelined in the ARM processor. The ARM processor takes the prefetch abort into account only if the read instruction is actually executed and it is probable that several aborts have occurred during this time. So, in this case, it is preferable to use the content of the Abort Link register of the ARM processor.

## Misalignment Detector

The Memory Controller features a Misalignment Detector that checks the consistency of the accesses.

For each access, regardless of the master, the size of access and the 0 and 1 bits of the address bus are checked. If the type of access is a word (32-bit) and the 0 and 1 bits are not 0, or if the type of the access is a half-word (16-bit) and the 0 bit is not 0, an abort is returned to the master and the access is cancelled. Note that the accesses of the ARM processor when it is fetching instructions are not checked.



The misalignments are generally due to software errors leading to wrong pointer handling. These errors are particularly difficult to detect in the debug phase.

As the requested address is saved in the Abort Status and the address of the instruction generating the misalignment is saved in the Abort Link Register of the processor, detection and correction of this kind of software error is simplified.

## Memory Controller Interrupt

The Memory Controller itself does not generate any interrupt. However, as indicated in Figure 28, the Memory Controller receives an interrupt signal from the External Bus Interface, which might be activated in case of Refresh Error detected by the SDRAM Controller. This interrupt signal just transits through the Memory Controller, which can neither enable/disable it nor return its activity.

This Memory Controller interrupt signal is ORed with the other System Peripheral interrupt lines (RTC, ST, DBGU, PMC) to provide the System Interrupt on Source 1 of the Advanced Interrupt Controller.

## User Interface

**Base Address:** 0xFFFFFFF00

**Table 36.** RM9200 Memory Controller Memory Map

| Offset      | Register                         | Name                                      | Access     | Reset State |
|-------------|----------------------------------|---|------------|-------------|
| 0x00        | MC Remap Control Register        | MC_RCR                                    | Write-only |             |
| 0x04        | MC Abort Status Register         | MC_ASR                                    | Read-only  | 0x0         |
| 0x08        | MC Abort Address Status Register | MC_AASR                                   | Read-only  | 0x0         |
| 0x0C        | MC Master Priority Register      | MC_MPR                                    | Read/Write | 0x3210      |
| 0x10 - 0x5C | Reserved                         |   |            |             |
| 0x60        | EBI Configuration Registers      | See EBI Datasheet, literature number 1759 |            |             |

## MC Remap Control Register

**Register Name:** MC\_RCR

**Access Type:** Write-only

**Absolute Address:** 0xFFFF FF00

|    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|-----|
| 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**

0: No effect.

1: This Command Bit acts on a toggle basis: writing a 1 alternatively cancels and restores the remapping of the page zero memory devices.

## MC Abort Status Register

**Register Name:** MC\_ASR  
**Access Type:** Read-only  
**Reset Value:** 0x0  
**Absolute Address:** 0xFFFF FF04

|    |    |    |    |        |        |        |        |
|----|----|----|----|--------|--------|--------|--------|
| 31 | 30 | 29 | 28 | 27     | 26     | 25     | 24     |
| –  | –  | –  | –  | SVMST3 | SVMST2 | SVMST1 | SVMST0 |
| 23 | 22 | 21 | 20 | 19     | 18     | 17     | 16     |
| –  | –  | –  | –  | MST3   | MST2   | MST1   | MST0   |
| 15 | 14 | 13 | 12 | 11     | 10     | 9      | 8      |
| –  | –  | –  | –  | ABTTYP |        | ABTSZ  |        |
| 7  | 6  | 5  | 4  | 3      | 2      | 1      | 0      |
| –  | –  | –  | –  | –      | –      | MISADD | UNDADD |

- **UNDADD: Undefined Address Abort Status**

0: The last abort was not due to the access of an undefined address in the address space.

1: The last abort was due to the access of an undefined address in the address space.

- **MISADD: Misaligned Address Abort Status**

0: The last aborted access was not due to an address misalignment.

1: The last aborted access was due to an address misalignment.

- **ABTSZ: Abort Size Status**

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

- **ABTTYP: Abort Type Status**

| ABTTYP |   | Abort Type |
|--------|---|------------|
| 0      | 0 | Data Read  |
| 0      | 1 | Data Write |
| 1      | 0 | Code Fetch |
| 1      | 1 | Reserved   |

- **MST0: ARM920T Abort Source**

0: The last aborted access was not due to the ARM920T.

1: The last aborted access was due to the ARM920T.

- **MST1: PDC Abort Source**

0: The last aborted access was not due to the PDC.

1: The last aborted access was due to the PDC.

- **MST2: UHP Abort Source**

0: The last aborted access was not due to the UHP.

1: The last aborted access was due to the UHP.

- **MST3: EMAC Abort Source**

0: The last aborted access was not due to the EMAC.

1: The last aborted access was due to the EMAC.

- **SVMST0: Saved ARM920T Abort Source**

0: No abort due to the ARM920T occurred since the last read of MC\_ASR or it is notified in the bit MST0.

1: At least one abort due to the ARM920T occurred since the last read of MC\_ASR.

- **SVMST1: Saved PDC Abort Source**

0: No abort due to the PDC occurred since the last read of MC\_ASR or it is notified in the bit MST1.

1: At least one abort due to the PDC occurred since the last read of MC\_ASR.

- **SVMST2: Saved UHP Abort Source**

0: No abort due to the UHP occurred since the last read of MC\_ASR or it is notified in the bit MST2.

1: At least one abort due to the UHP occurred since the last read of MC\_ASR.

- **SVMST3: Saved EMAC Abort Source**

0: No abort due to the EMAC occurred since the last read of MC\_ASR or it is notified in the bit MST3.

1: At least one abort due to the EMAC occurred since the last read of MC\_ASR.

### MC Abort Address Status Register

Register Name: MC\_AASR  
 Access Type: Read-only  
 Reset Value: 0x0  
 Absolute Address: 0xFFFF FF08

|        |    |    |    |    |    |    |    |
|--------|----|----|----|----|----|----|----|
| 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 of the last aborted access.

## MC Master Priority Register

Register Name: MC\_MPR

Access Type: Read/Write

Reset Value: 0x3210

Absolute Address: 0xFFFF FF0C

|    |       |    |    |    |       |    |    |
|----|-------|----|----|----|-------|----|----|
| 31 | 30    | 29 | 28 | 27 | 26    | 25 | 24 |
| -  | -     | -  | -  | -  | -     | -  | -  |
| 23 | 22    | 21 | 20 | 19 | 18    | 17 | 16 |
| -  | -     | -  | -  | -  | -     | -  | -  |
| 15 | 14    | 13 | 12 | 11 | 10    | 9  | 8  |
| -  | MSTP3 |    |    | -  | MSTP2 |    |    |
| 7  | 6     | 5  | 4  | 3  | 2     | 1  | 0  |
| -  | MSTP1 |    |    | -  | MSTP0 |    |    |

- **MSTP0: ARM920T Priority**
- **MSTP1: PDC Priority**
- **MSTP2: UHP Priority**
- **MSTP3: EMAC Priority**

000: Lowest priority

111: Highest priority

In the case of equal priorities, Master 0 has highest and Master 3 has lowest priority.

## External Bus Interface (EBI)

### Overview

The External Bus Interface (EBI) is designed to ensure the successful data transfer between several external devices and the embedded Memory Controller of an ARM®-based device. The Static Memory, SDRAM and Burst Flash Controllers are all featured external Memory Controllers on the EBI. These external Memory Controllers are capable of handling several types of external memory and peripheral devices, such as SRAM, PROM, EPROM, EEPROM, Flash, SDRAM and Burst Flash.

The EBI also supports the CompactFlash and the SmartMedia protocols via integrated circuitry that greatly reduces the requirements for external components. Furthermore, the EBI handles data transfers with up to eight external devices, each assigned to eight address spaces defined by the embedded Memory Controller. Data transfers are performed through a 16-bit or 32-bit data bus, an address bus of up to 26 bits, up to eight chip select lines (NCS[7:0]) and several control pins that are generally multiplexed between the different external Memory Controllers.

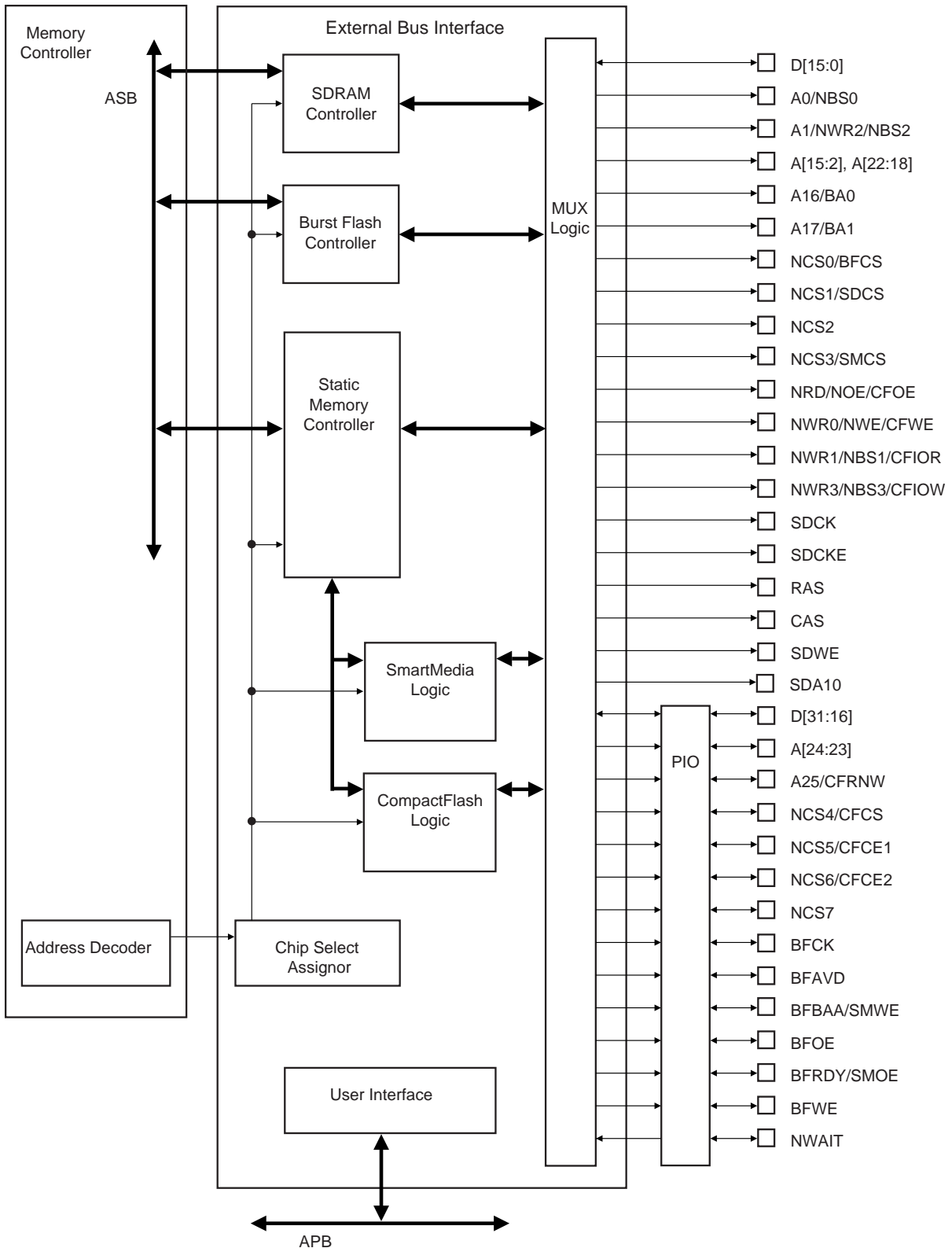
Features of the EBI are:

- Integrates Three External Memory Controllers:
  - Static Memory Controller
  - SDRAM Controller
  - Burst Flash Controller
- Additional Logic for SmartMedia™ and CompactFlash™ Support
- Optimized External Bus:
  - 16- or 32-bit Data Bus
  - Up to 26-bit Address Bus, Up to 64-Mbytes Addressable
  - Up to 8 Chip Selects, Each Reserved to One of the Eight Memory Areas
  - Optimized Pin Multiplexing to Reduce Latencies on External Memories
- Configurable Chip Select Assignment:
  - Burst Flash Controller or Static Memory Controller on NCS0
  - SDRAM Controller or Static Memory Controller on NCS1
  - Static Memory Controller on NCS3, Optional SmartMedia Support
  - Static Memory Controller on NCS4 - NCS6, Optional CompactFlash Support
  - Static Memory Controller on NCS7

# Block Diagram

Figure 31 below shows the organization of the External Bus Interface.

**Figure 31.** Organization of the External Bus Interface





## I/O Lines Description

Table 37. I/O Lines Description

| Name                                | Function                           | Type   | Active Level |
|-------------------------------------|------------------------------------|--------|--------------|
| <b>EBI</b>                          |                                    |        |              |
| D[31:0]                             | Data Bus                           | I/O    |              |
| A[25:0]                             | Address Bus                        | Output |              |
| <b>SMC</b>                          |                                    |        |              |
| NCS[7:0]                            | Chip Select Lines                  | Output | Low          |
| NWR[1:0]                            | Write Signals                      | Output | Low          |
| NOE                                 | Output Enable                      | Output | Low          |
| NRD                                 | Read Signal                        | Output | Low          |
| NBS1                                | NUB: Upper Byte Select             | Output | Low          |
| NBS0                                | NLB: Lower Byte Select             | Output | Low          |
| NWE                                 | Write Enable                       | Output | Low          |
| <b>EBI for CompactFlash Support</b> |                                    |        |              |
| CFCE[2:1]                           | CompactFlash Chip Enable           | Output | Low          |
| CFOE                                | CompactFlash Output Enable         | Output | Low          |
| CFWE                                | CompactFlash Write Enable          | Output | Low          |
| CFIOR                               | CompactFlash I/O Read Signal       | Output | Low          |
| CFIOW                               | CompactFlash I/O Write Signal      | Output | Low          |
| CFRNW                               | CompactFlash Read Not Write Signal | Output |              |
| CFCS                                | CompactFlash Chip Select Line      | Output | Low          |
| NWAIT                               | CompactFlash Wait Signal           | Input  | Low          |
| <b>EBI for SmartMedia Support</b>   |                                    |        |              |
| SMCS                                | SmartMedia Chip Select Line        | Output | Low          |
| SMOE                                | SmartMedia Output Enable           | Output | Low          |
| SMWE                                | SmartMedia Write Enable            | Output | Low          |
| <b>SDRAM Controller</b>             |                                    |        |              |
| SDCK                                | SDRAM Clock                        | Output |              |
| SDCKE                               | SDRAM Clock Enable                 | Output | High         |
| SDCS                                | SDRAM Controller Chip Select Line  | Output | Low          |
| BA[1:0]                             | Bank Select                        | Output |              |
| SDWE                                | SDRAM Write Enable                 | Output | Low          |
| RAS - CAS                           | Row and Column Signal              | Output | Low          |
| NWR[3:0]                            | Write Signals                      | Output | Low          |
| NBS[3:0]                            | Byte Mask Signals                  | Output | Low          |
| SDA10                               | SDRAM Address 10 Line              | Output |              |

**Table 37.** I/O Lines Description (Continued)

| Name                          | Function                           | Type   | Active Level |
|-------------------------------|------------------------------------|--------|--------------|
| <b>Burst Flash Controller</b> |                                    |        |              |
| BFCK                          | Burst Flash Clock                  | Output |              |
| BFCS                          | Burst Flash Chip Select Line       | Output | Low          |
| BFAVD                         | Burst Flash Address Valid Signal   | Output | Low          |
| BFBA                          | Burst Flash Address Advance Signal | Output | Low          |
| BFOE                          | Burst Flash Output Enable          | Output | Low          |
| BFRDY                         | Burst Flash Ready Signal           | Input  | High         |
| BFWE                          | Burst Flash Write Enable           | Output | Low          |

The connection of some signals through the Mux logic is not direct and depends on the Memory Controller in use at the moment.

Table 38 below details the connections between the three Memory Controllers and the EBI pins.

**Table 38.** EBI Pins and Memory Controllers I/O Line Connections

| EBI Pins        | SDRAMC I/O Lines | BFC I/O Lines | SMC I/O Lines |
|-----------------|------------------|---------------|---------------|
| NWR1/NBS1/CFIOR | NBS1             | Not Supported | NWR1/NUB      |
| A0/NBS0         | Not Supported    | Not Supported | A0/NLB        |
| A1              | Not Supported    | A0            | A1            |
| A[11:2]         | A[9:0]           | A[10:1]       | A[11:2]       |
| SDA10           | A10              | Not Supported | Not Supported |
| A12             | Not Supported    | A11           | A12           |
| A[14:13]        | A[12:11]         | A[13:12]      | A[14:13]      |
| A[25:15]        | Not Supported    | A[24:14]      | A[25:15]      |
| D[31:16]        | D[31:16]         | Not Supported | Not Supported |
| D[15:0]         | D[15:0]          | D[15:0]       | D[15:0]       |

## Application Example

**Hardware Interface** Table 39 below details the connections to be applied between the EBI pins and the external devices for each Memory Controller.

**Table 39.** EBI Pins and External Device Connections

| Pin             | Pins of the Interfaced Device |                          |                      |                    |           |                      |                          |
|-----------------|-------------------------------|--------------------------|----------------------|--------------------|-----------|----------------------|--------------------------|
|                 | 8-bit Static Device           | 2 x 8-bit Static Devices | 16-bit Static Device | Burst Flash Device | SDRAM     | CompactFlash         | SmartMedia or NAND Flash |
| Controller      | SMC                           |                          |                      | BFC                | SDRAMC    | SMC                  |                          |
| D0 - D7         | D0 - D7                       | D0 - D7                  | D0 - D7              | D0 - D7            | D0 - D7   | D0 - D7              | AD0 - AD7                |
| D8 - D15        | –                             | D8 - D15                 | D8 - D15             | D8 - D15           | D8 - D15  | D8 - 15              | –                        |
| D16 - D31       | –                             | –                        | –                    | –                  | D16 - D31 | –                    | –                        |
| A0/NBS0         | A0                            | –                        | NLB                  | –                  | DQM0      | A0                   | –                        |
| A1/NWR2/NBS2    | A1                            | A0                       | A0                   | A0                 | DQM2      | A1                   | –                        |
| A2 - A9         | A2 - A9                       | A1 - A8                  | A1 - A8              | A1 - A8            | A0 - A7   | A2 - A9              | –                        |
| A10             | A10                           | A9                       | A9                   | A9                 | A8        | A10                  | –                        |
| A11             | A11                           | A10                      | A10                  | A10                | A9        | –                    | –                        |
| SDA10           | –                             | –                        | –                    | –                  | A10       | –                    | –                        |
| A12             | A12                           | A11                      | A11                  | A11                | –         | –                    | –                        |
| A13 - A15       | A13 - A15                     | A12 - A14                | A12 - A14            | A12 - A14          | A11 - A13 | –                    | –                        |
| A16/BA0         | A16                           | A15                      | A15                  | A15                | BA0       | –                    | –                        |
| A17/BA1         | A17                           | A16                      | A16                  | A16                | BA1       | –                    | –                        |
| A18 - A20       | A18 - A20                     | A17 - A19                | A17 - A19            | A17 - A19          | –         | –                    | –                        |
| A21             | A21                           | A20                      | A20                  | A20                | –         | –                    | CLE <sup>(4)</sup>       |
| A22             | A22                           | A21                      | A21                  | A21                | –         | REG <sup>(3)</sup>   | ALE <sup>(4)</sup>       |
| A23 - A24       | A23 - A24                     | A22 - A23                | A22 - A23            | A22 - A23          | –         | –                    | –                        |
| A25             | A25                           | A24                      | A24                  | A24                | –         | CFRNW <sup>(1)</sup> | –                        |
| NCS0/BFCS       | CS                            | CS                       | CS                   | CS                 | –         | –                    | –                        |
| NCS1/SDCS       | CS                            | CS                       | CS                   | –                  | CS        | –                    | –                        |
| NCS2            | CS                            | CS                       | CS                   | –                  | –         | –                    | –                        |
| NCS3/SMCS       | CS                            | CS                       | C S                  | –                  | –         | –                    | –                        |
| NCS4/CFCS       | CS                            | CS                       | CS                   | –                  | –         | CFCS <sup>(1)</sup>  | –                        |
| NCS5/CFCE1      | CS                            | CS                       | CS                   | –                  | –         | CE1                  | –                        |
| NCS6/CFCE2      | CS                            | CS                       | CS                   | –                  | –         | CE2                  | –                        |
| NRD/NOE/CFOE    | OE                            | OE                       | OE                   | –                  | –         | OE                   | –                        |
| NWR0/NWE/CFWE   | WE                            | WE <sup>(5)</sup>        | WE                   | –                  | –         | WE                   | –                        |
| NWR1/NBS1/CFIOR | WE                            | WE <sup>(5)</sup>        | NUB                  | –                  | DQM1      | IOR                  | –                        |



**Table 39.** EBI Pins and External Device Connections (Continued)

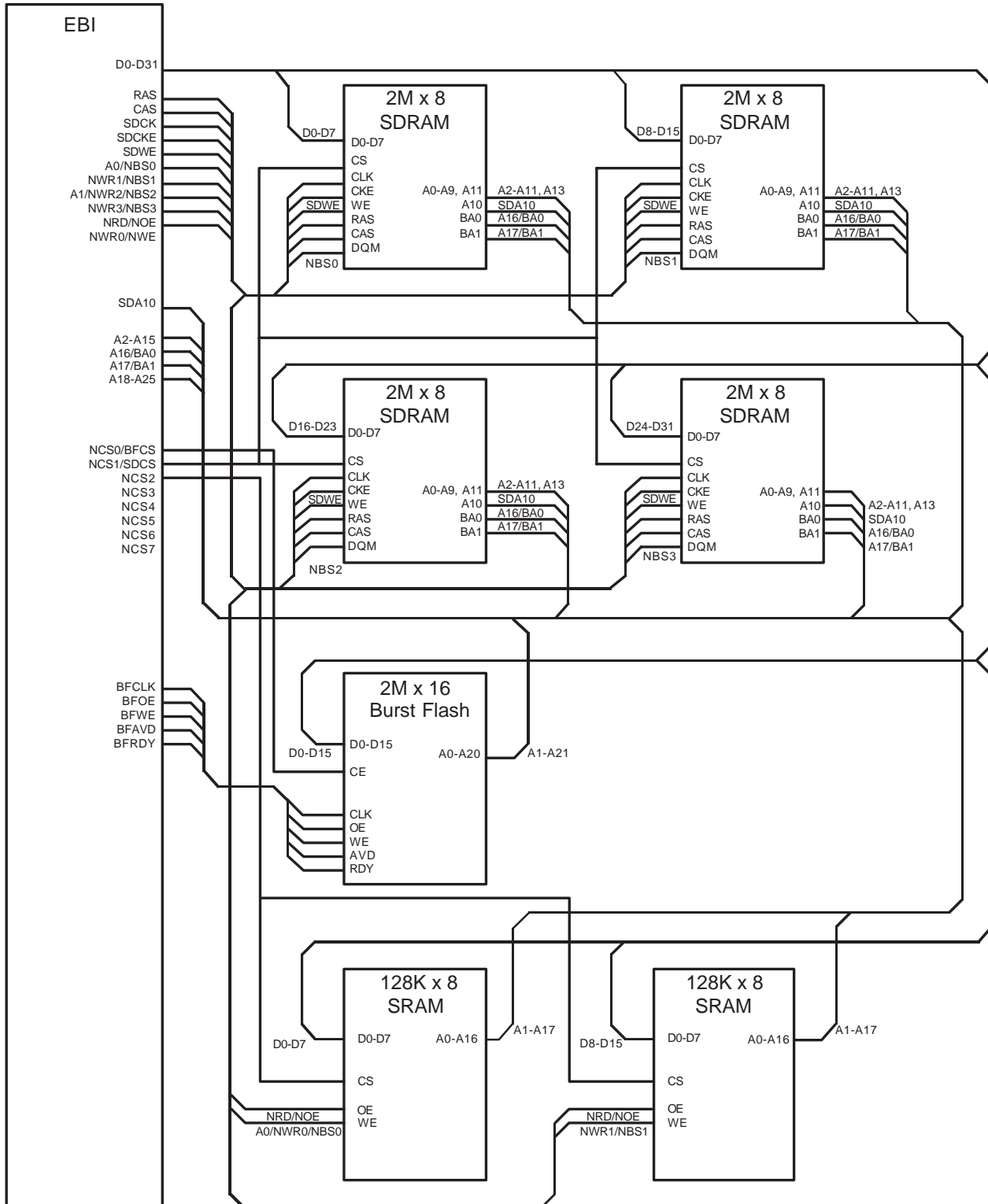
| Pin                | Pins of the Interfaced Device |                          |                      |                    |        |              |                          |
|--------------------|-------------------------------|--------------------------|----------------------|--------------------|--------|--------------|--------------------------|
|                    | 8-bit Static Device           | 2 x 8-bit Static Devices | 16-bit Static Device | Burst Flash Device | SDRAM  | CompactFlash | SmartMedia or NAND Flash |
| Controller         | SMC                           |                          |                      | BFC                | SDRAMC | SMC          |                          |
| NWR3/NBS3/CFIOW    | –                             | –                        | –                    | –                  | DQM3   | IOW          | –                        |
| BFCK               | –                             | –                        | –                    | CK                 | –      | –            | –                        |
| BFAVD              | –                             | –                        | –                    | AVD                | –      | –            | –                        |
| BFBA/SMWE          | –                             | –                        | –                    | BAA                | –      | –            | WE                       |
| BFOE               | –                             | –                        | –                    | OE                 | –      | –            | –                        |
| BFRDY/SMOE         | –                             | –                        | –                    | RDY                | –      | –            | OE                       |
| BFWE               | –                             | –                        | –                    | WE                 | –      | –            | –                        |
| SDCK               | –                             | –                        | –                    | –                  | CLK    | –            | –                        |
| SDCKE              | –                             | –                        | –                    | –                  | CKE    | –            | –                        |
| RAS                | –                             | –                        | –                    | –                  | RAS    | –            | –                        |
| CAS                | –                             | –                        | –                    | –                  | CAS    | –            | –                        |
| SDWE               | –                             | –                        | –                    | –                  | WE     | –            | –                        |
| NWAIT              | –                             | –                        | –                    | –                  | –      | WAIT         | –                        |
| Pxx <sup>(2)</sup> | –                             | –                        | –                    | –                  | –      | CD1 or CD2   | –                        |
| Pxx <sup>(2)</sup> | –                             | –                        | –                    | –                  | –      | –            | CE                       |
| Pxx <sup>(2)</sup> | –                             | –                        | –                    | –                  | –      | –            | RDY                      |

- Notes:
1. Not directly connected to the CompactFlash slot. Permits the control of the bidirectional buffer between the EBI data bus and the CompactFlash slot.
  2. Any PIO line.
  3. The REG signal of the CompactFlash can be driven by any of the following address bits: A24, A22 to A11. For details, see “CompactFlash Support” on page 143.
  4. The CLE and ALE signals of the SmartMedia device may be driven by any address bit. For details, see “SmartMedia and NAND Flash Support” on page 146.
  5. NWR1 enables upper byte writes. NWR0 enables lower byte writes.

## Connection Examples

Figure 32 below shows an example of connections between the EBI and external devices.

**Figure 32.** EBI Connections to Memory Devices



## Product Dependencies

### I/O Lines

The pins used for interfacing the External Bus Interface may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the External Bus Interface pins to their peripheral function. If I/O lines of the External Bus Interface are not used by the application, they can be used for other purposes by the PIO Controller.

### Functional Description

The EBI transfers data between the internal ASB Bus (handled by the Memory Controller) and the external memories or peripheral devices. It controls the waveforms and the parameters of the external address, data and control busses and is composed of the following elements:

- The Static Memory Controller (SMC)
- The SDRAM Controller (SDRAMC)
- The Burst Flash Controller (BFC)
- A chip select assignment feature that assigns an ASB address space to the external devices.
- A multiplex controller circuit that shares the pins between the different Memory Controllers.
- Programmable CompactFlash support logic
- Programmable SmartMedia and NAND Flash support logic

### Bus Multiplexing

The EBI offers a complete set of control signals that share the 32-bit data lines, the address lines of up to 26 bits and the control signals through a multiplex logic operating in function of the memory area requests.

Multiplexing is specifically organized in order to guarantee the maintenance of the address and output control lines at a stable state while no external access is being performed. Multiplexing is also designed to respect the data float times defined in the Memory Controllers. Furthermore, refresh cycles of the SDRAM are executed independently by the SDRAM Controller without delaying the other external Memory Controller accesses. Lastly, it prevents burst accesses on the same page of a burst Flash from being interrupted which avoids the need to restart a high-latency first access.

### Pull-up Control

The EBI permits enabling of on-chip pull-up resistors on the data bus lines not multiplexed with the PIO Controller lines. The pull-up resistors are enabled after reset. Setting the DBPUC bit disables the pull-up resistors on the D0 to D15 lines. Enabling the pull-up resistor on the D16 - D31 lines can be performed by programming the appropriate PIO controller.

### Static Memory Controller

For information on the Static Memory Controller, refer to the SMC “Overview” on page 151.

### SDRAM Controller

For information on the SDRAM Controller, refer to the SDRAMC description on “Overview” on page 135.

### Burst Flash Controller

For information on the Burst Flash Controller, refer to the BFC “Overview” on page 209.

## CompactFlash Support

The External Bus Interface integrates circuitry that interfaces to CompactFlash devices.

The CompactFlash logic is driven by the Static Memory Controller (SMC) on the NCS4 address space. Programming the CS4A field of the Chip Select Assignment Register (See “EBI Chip Select Assignment Register” on page 149.) to the appropriate value enables this logic. Access to an external CompactFlash device is then made by accessing the address space reserved to NCS4 (i.e., between 0x5000 0000 and 0x5FFF FFFF).

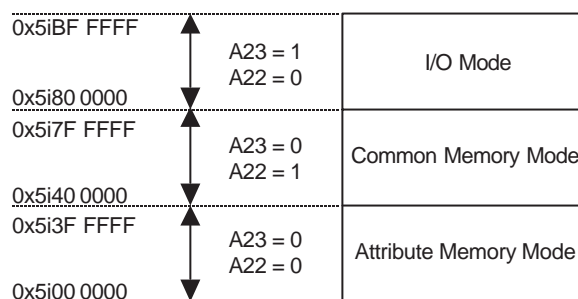
When multiplexed with CFCE1 and CFCE2 signals, the NCS5 and NCS6 signals become unavailable. Performing an access within the address space reserved to NCS5 and NCS6 (i.e., between 0x6000 0000 and 0x7FFF FFFF) may lead to an unpredictable outcome.

The True IDE Mode is not supported and in I/O Mode, the signal `_IOIS16` is not managed.

## I/O Mode, Common Memory Mode and Attribute Memory Mode

Within the NCS4 address space, the current transfer address is used to distinguish I/O mode, common memory mode and attribute memory mode. More precisely, the A23 bit of the transfer address is used to select I/O Mode. Any EBI address bit not required by the CompactFlash device (i.e., bit A24 or bits A22 to A11) can be used to separate common memory mode and attribute memory mode. Using the A22 bit, for example, leads to the address map in Figure 33 below. In this figure, “i” stands for any hexadecimal digit.

**Figure 33.** Address Map Example



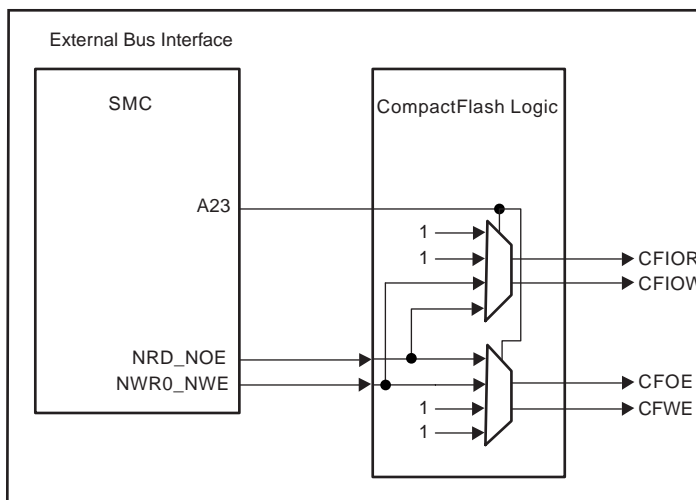
Note: In the above example, the A22 pin of the EBI can be used to drive the REG signal of the CompactFlash Device.

## Read/Write Signals

In I/O mode, the CompactFlash logic drives the read and write command signals of the SMC on CFIOR and CFIOW signals, while the CFOE and CFWE signals are deactivated. Likewise, in common memory mode and attribute memory mode, the SMC signals are driven on the CFOE and CFWE signals, while the CFIOR and CFIOW are deactivated. Figure 34 on page 144 demonstrates a schematic representation of this logic.

Attribute memory mode, common memory mode and I/O mode are supported by setting the address setup and hold time on the NCS4 chip select to the appropriate values. For details on these signal waveforms, please refer to the section: “Setup and Hold Cycles” on page 164 of the Static Memory Controller documentation.

**Figure 34.** CompactFlash Read/Write Control Signals



**Access Type**

The CFCE1 and CFCE2 signals enable upper- and lower-byte access on the data bus of the CompactFlash device in accordance with Table 40 below. The odd byte access on the D[7:0] bus is only possible when the SMC is configured to drive 8-bit memory devices on the NCS4 pin. The Chip Select Register (DBW field in “SMC Chip Select Registers” on page 186) of the NCS4 address space must be set as shown in Table 40 to enable the required access type. The CFCE1 and CFCE2 waveforms are identical to the NCS4 waveform. For details on these waveforms and timings, refer to the Static Memory Controller “Overview” on page 151.

**Table 40.** Upper- and Lower-byte Access

| Access               | CFCE2 | CFCE1 | A0 | D[15:8]           | D[7:0]            | SMC_CSR4 (DBW)  |
|----------------------|-------|-------|----|-------------------|-------------------|-----------------|
| Byte R/W Access      | 1     | 0     | 0  | Don't Care/High Z | Even Byte         | 8-bit or 16-bit |
|                      | 1     | 0     | 1  | Don't Care/High Z | Odd Byte          | 8-bit           |
| Odd Byte R/W Access  | 0     | 1     | X  | Odd Byte          | Don't Care/High Z | 16-bit          |
| Half-word R/W Access | 0     | 0     | X  | Odd Byte          | Even Byte         | 16-bit          |

**Multiplexing of CompactFlash Signals on EBI Pins**

Table 41 below and Table 42 on page 145 illustrate the multiplexing of the CompactFlash logic signals with other EBI signals on the EBI pins. The EBI pins in Table 41 are strictly dedicated to the CompactFlash interface as soon as the CS4A field of the Chip Select Assignment Register is set (See “EBI Chip Select Assignment Register” on page 149.). These pins must not be used to drive any other memory devices.

The EBI pins in Table 42 on page 145 remain shared between all memory areas when the CompactFlash interface is enabled (CS4A = 1).

**Table 41.** Dedicated CompactFlash Interface Multiplexing

| Pins       | CS4A = 1             | CS4A = 0    |
|------------|----------------------|-------------|
|            | CompactFlash Signals | EBI Signals |
| NCS4/CFCS  | CFCS                 | NCS4        |
| NCS5/CFCE1 | CFCE1                | NCS5        |
| NCS6/CFCE2 | CFCE2                | NCS6        |



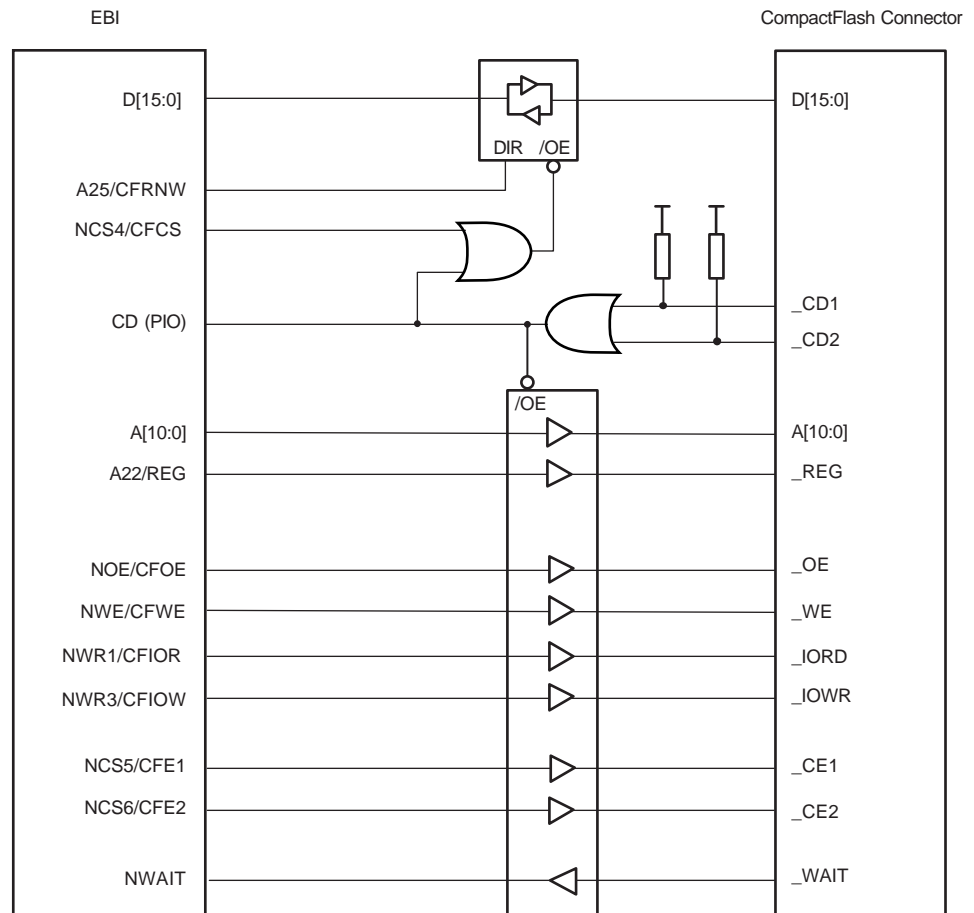
**Table 42.** Shared CompactFlash Interface Multiplexing

| Pins            | Access to CompactFlash Device | Access to Other EBI Devices |
|-----------------|-------------------------------|-----------------------------|
|                 | CompactFlash Signals          | EBI Signals                 |
| NOE/NRD/CFOE    | CFOE                          | NRD/NOE                     |
| NWR0/NWE/CFWE   | CFWE                          | NWR0/NWE                    |
| NWR1/NBS1/CFIOR | CFIOR                         | NWR1/NBS1                   |
| NWR3/NBS3/CFIOW | CFIOW                         | NWR3/NBS3                   |
| A25/CFRNW       | CFRNW                         | A25                         |

**CompactFlash Application Example**

Figure 35 below illustrates an example of a CompactFlash application. CFCS and CFRNW signals are not directly connected to the CompactFlash slot, but do control the direction and the output enable of the buffers between the EBI and the CompactFlash Device. The timing of the CFCS signal is identical to the NCS4 signal. Moreover, the CFRNW signal remains valid throughout the transfer, as does the address bus. The CompactFlash \_WAIT signal is connected to the NWAIT input of the Static Memory Controller. For details on these waveforms and timings, refer to the Static Memory Controller “Overview” on page 135.

**Figure 35.** CompactFlash Application Example



## SmartMedia and NAND Flash Support

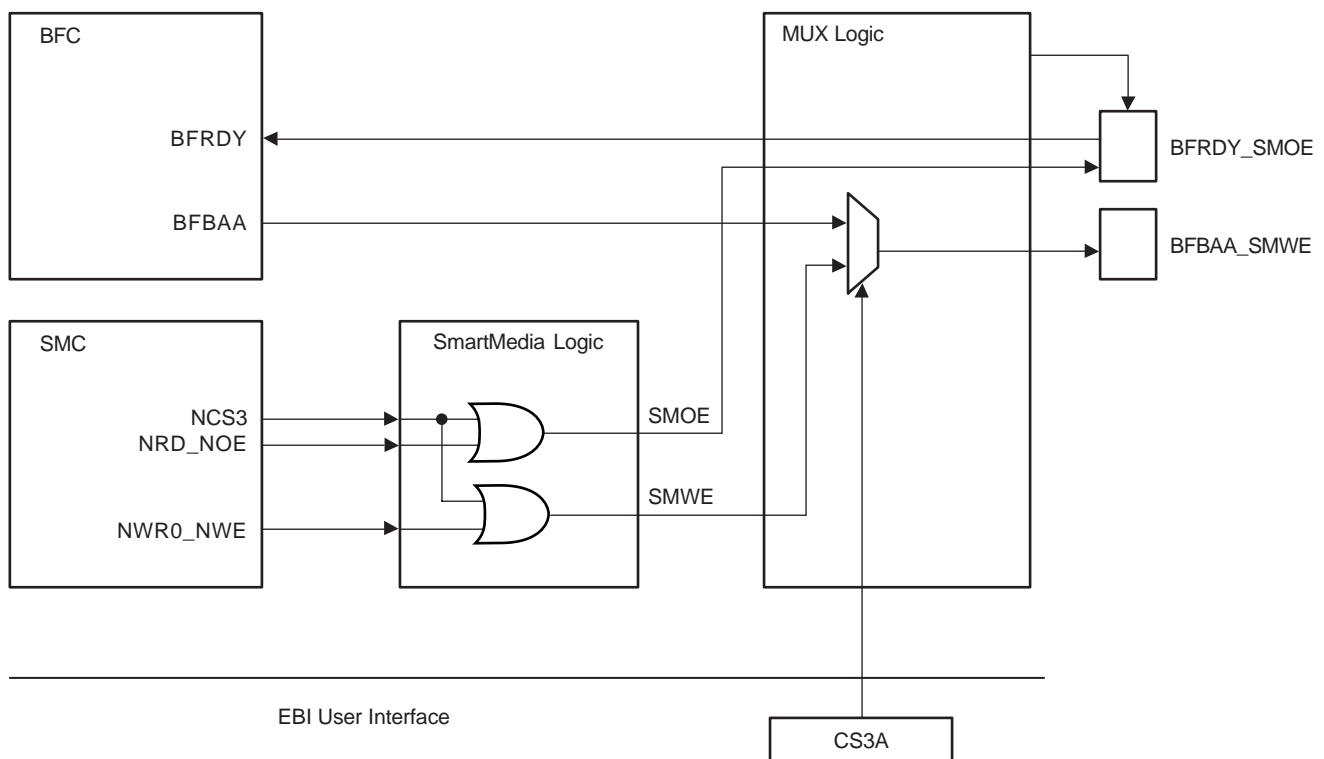
The EBI integrates circuitry that interfaces to SmartMedia and NAND Flash devices.

The SmartMedia logic is driven by the Static Memory Controller on the NCS3 address space. Programming the CS3A field in the Chip Select Assignment Register to the appropriate value enables the SmartMedia logic (See “EBI Chip Select Assignment Register” on page 149.). Access to an external SmartMedia device is then made by accessing the address space reserved to NCS3 (i.e., between 0x4000 0000 and 0x4FFF FFFF).

The SmartMedia Logic drives the read and write command signals of the SMC on the SMOE and SMWE signals when the NCS3 signal is active. SMOE and SMWE are invalidated as soon as the transfer address fails to lie in the NCS3 address space. For details on these waveforms, refer to the Static Memory Controller “Overview” on page 151.

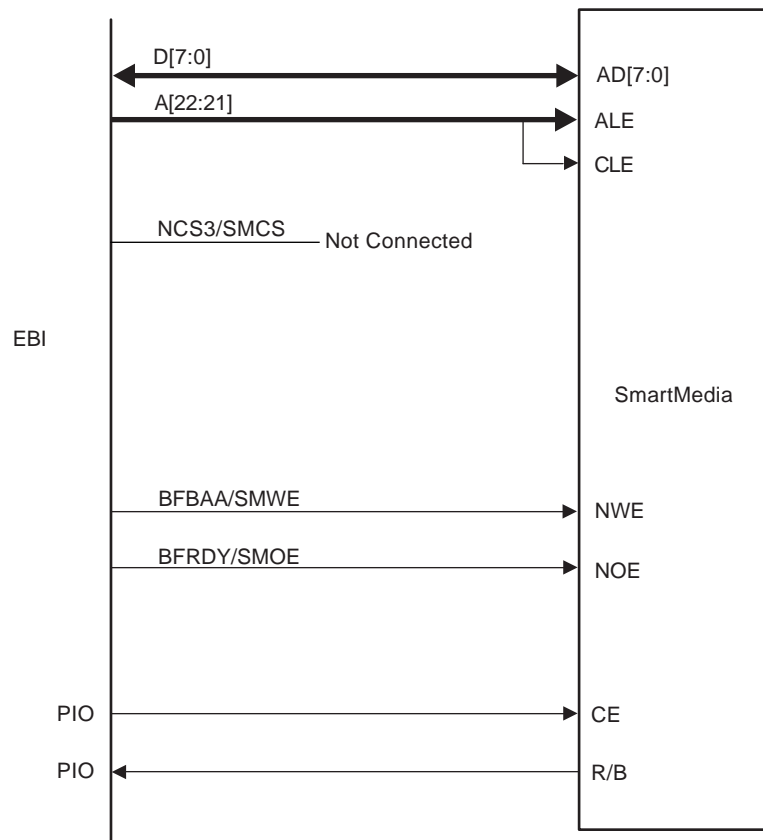
The SMWE and SMOE signals are multiplexed with BFRDY and BFBAAsignals of the Burst Flash Controller. This multiplexing is controlled in the MUX logic part of the EBI by the CS3A field of the Chip Select Assignment Register (See “EBI Chip Select Assignment Register” on page 149.). This logic also controls the direction of the BFRDY/SMOE pad.

**Figure 36.** SmartMedia Signal Multiplexing on EBI Pins



The address latch enable and command latch enable signals on the SmartMedia device are driven by address bits A22 and A21 of the EBI address bus. The user should note that any bit on the EBI address bus can also be used for this purpose. The command, address or data words on the data bus of the SmartMedia device are distinguished by using their address within the NCS3 address space. The chip enable (CE) signal of the device and the ready/busy (R/B) signals are connected to PIO lines. The CE signal then remains asserted even when NCS3 is not selected, preventing the device from returning to standby mode. Some functional limitation with the supported burst Flash device will occur when the SmartMedia device is activated due to the fact that the SMOE and SMWE signals are multiplexed with BFRDY and BFBAAsignals respectively.

Figure 37. SmartMedia Application Example



## External Bus Interface (EBI) User Interface

AT91RM9200 EBI User Interface Base Address: 0xFFFF FF60

**Table 43.** External Bus Interface Memory Map

| Offset      | Register                        | Name  | Access     | Reset State |
|-------------|---------------------------------|---|------------|-------------|
| 0x00        | Chip Select Assignment Register | EBI_CSA   | Read/Write | 0x0         |
| 0x04        | Configuration Register          | EBI_CFGR  | Read/Write | 0x0         |
| 0x08        | Reserved                        |   | –          |             |
| 0x0C        | Reserved                        |   | –          |             |
| 0x10 - 0x2C | SMC User Interface              | See “Static Memory Controller (SMC) User Interface” on page 185 |            |             |
| 0x30 - 0x5C | SDRAMC User Interface           | See “SDRAM Controller (SDRAMC) User Interface” on page 201.     |            |             |
| 0x60        | BFC User Interface              | See “Burst Flash Controller (BFC) User Interface” on page 221.  |            |             |
| 0x64 - 0x9C | Reserved                        |   |            |             |

### EBI Chip Select Assignment Register

Register Name: EBI\_CSA  
 Access Type: Read/write  
 Reset Value: 0x0  
 Offset: 0x0  
 Absolute Address: 0xFFFF FF60

|    |    |    |      |      |    |      |      |
|----|----|----|------|------|----|------|------|
| 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    |
| –  | –  | –  | CS4A | CS3A | –  | CS1A | CS0A |

• **CS0A: Chip Select 0 Assignment**

0 = Chip Select 0 is assigned to the Static Memory Controller.  
 1 = Chip Select 0 is assigned to the Burst Flash Controller.

• **CS1A: Chip Select 1 Assignment**

0 = Chip Select 1 is assigned to the Static Memory Controller.  
 1 = Chip Select 1 is assigned to the SDRAM Controller.

• **CS3A: Chip Select 3 Assignment**

0 = Chip Select 3 is only assigned to the Static Memory Controller and NCS3 behaves as defined by the SMC.  
 1 = Chip Select 3 is assigned to the Static Memory Controller and the SmartMedia Logic is activated.

• **CS4A: Chip Select 4 Assignment**

0 = Chip Select 4 is assigned to the Static Memory Controller and NCS4, NCS5 and NCS6 behave as defined by the SMC.  
 1 = Chip Select 4 is assigned to the Static Memory Controller and the CompactFlash Logic is activated.

Accessing the address space reserved to NCS5 and NCS6 may lead to an unpredictable outcome.



## EBI Configuration Register

**Register Name:** EBI\_CFGR  
**Access Type:** Read/write  
**Reset Value:** 0x0  
**Offset:** 0x04  
**Absolute Address:** 0xFFFF FF64

|    |    |    |    |    |    |    |       |
|----|----|----|----|----|----|----|-------|
| 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     |
| –  | –  | –  | –  | –  | –  |    | DBPUC |

- **DBPUC: Data Bus Pull-Up Configuration**

0 = [D15:0] Data Bus bits are internally pulled-up to the VDDIOM power supply.

1 = [D15:0] Data Bus bits are not internally pulled-up.

## Static Memory Controller (SMC)

### Overview

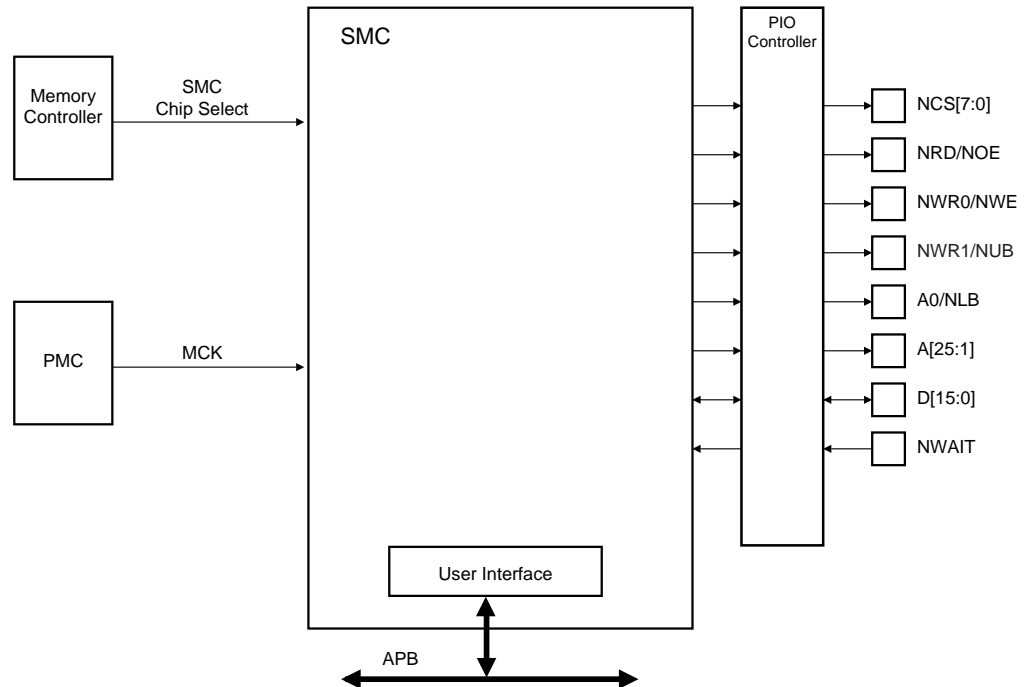
The Static Memory Controller (SMC) generates the signals that control the access to external static memory or peripheral devices. The SMC is fully programmable and can address up to 512M bytes. It has eight chip selects and a 26-bit address bus. The 16-bit data bus can be configured to interface with 8- or 16-bit external devices. Separate read and write control signals allow for direct memory and peripheral interfacing. The SMC supports different access protocols allowing single clock cycle memory accesses. It also provides an external wait request capability.

The main features of the SMC are:

- External memory mapping, 512-Mbyte address space
- Up to 8 Chip Select Lines
- 8- or 16-bit Data Bus
- Remap of Boot Memory
- Multiple Access Modes Supported
  - Byte Write or Byte Select Lines
  - Two different Read Protocols for each Memory Bank
- Multiple Device Adaptability
  - Compliant with LCD Module
  - Programmable Setup Time Read/Write
  - Programmable Hold Time Read/Write
- Multiple Wait State Management
  - Programmable Wait State Generation
  - External Wait Request
  - Programmable Data Float Time

## Block Diagram

**Figure 38.** Static Memory Controller Block Diagram



**Table 44.** I/O Lines Description

| Name     | Description                                | Type   | Active Level |
|----------|--|--------|--------------|
| NCS[7:0] | Static Memory Controller Chip Select Lines | Output | Low          |
| NRD/NOE  | Read/Output Enable Signal                  | Output | Low          |
| NWR0/NWE | Write 0/Write Enable Signal                | Output | Low          |
| NWR1/NUB | Write1/Upper Byte Select Signal            | Output | Low          |
| A0/NLB   | Address Bit 0/Lower Byte Select Signal     | Output | Low          |
| A[25:1]  | Address Bus                                | Output |              |
| D[15:0]  | Data Bus                                   | I/O    |              |
| NWAIT    | External Wait Signal                       | Input  | Low          |

Multiplexed signals are listed in Table 45 with their functions.

**Table 45.** Static Memory Controller Multiplexed Signals

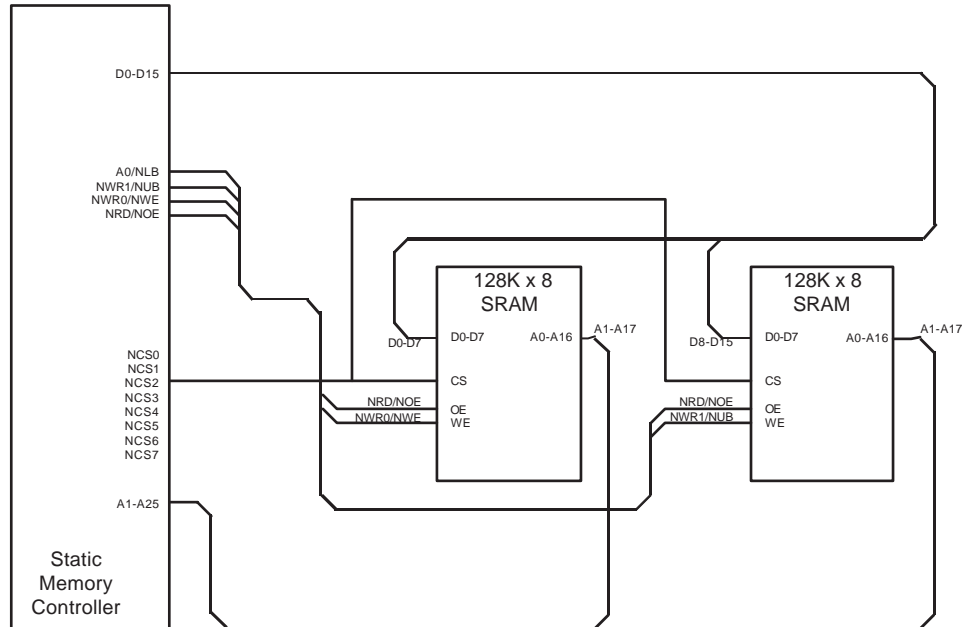
| Multiplexed Signals |     | Related Function   |
|---------------------|-----|--|
| A0                  | NLB | 8-bit or 16-bit data bus, see “Data Bus Width” on page 155.            |
| NRD                 | NOE | Byte-write or byte-select access, see “Write Access Type” on page 156. |
| NWR0                | NWE | Byte-write or byte-select access, see “Write Access Type” on page 156. |
| NWR1                | NUB | Byte-write or byte-select access, see “Write Access Type” on page 156. |



## Application Example

**Hardware Interface** Figure 39 shows an example of static memory device connection to the SMC.

**Figure 39.** SMC Connections to Static Memory Devices



## Product Dependencies

### I/O Lines

The pins used for interfacing the Static Memory Controller may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the Static Memory Controller pins to their peripheral function. If I/O lines of the Static Memory Controller are not used by the application, they can be used for other purposes by the PIO Controller.

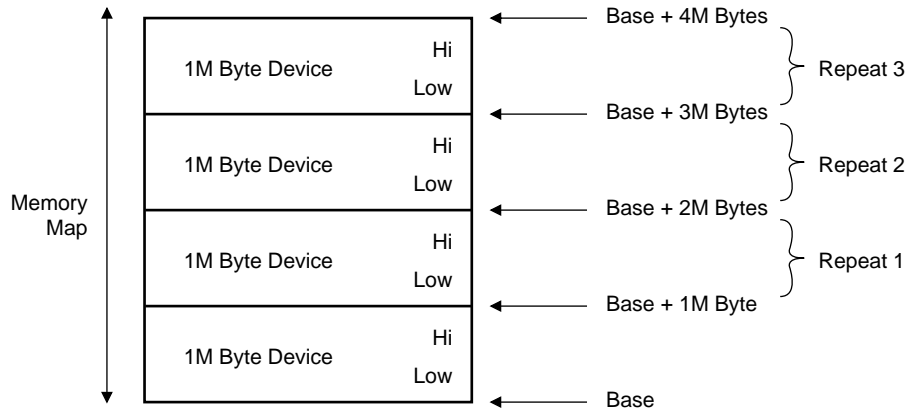
# Functional Description

## External Memory Interface

### External Memory Mapping

The memory map is defined by hardware and associates the internal 32-bit address space with the external 26-bit address bus. Note that A[25:0] is only significant for 8-bit memory. A[25:1] is used for 16-bit memory. If the physical memory device is smaller than the page size, it wraps around and appears to be repeated within the page. The SMC correctly handles any valid access to the memory device within the page. See Figure 40.

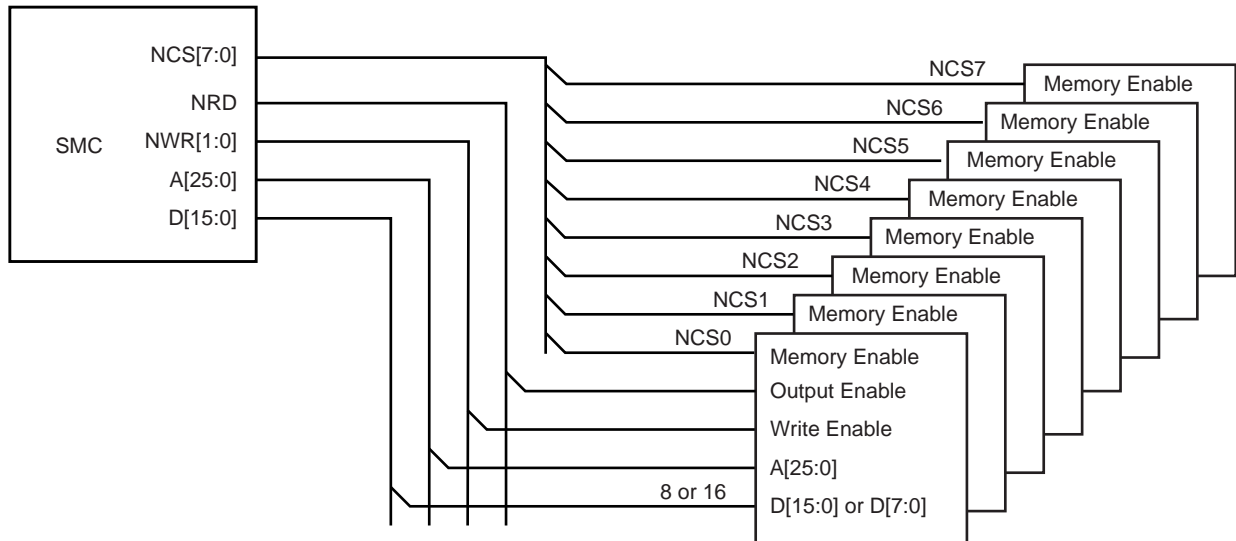
**Figure 40.** Case of an External Memory Smaller than Page Size



### Chip Select Lines

The Static Memory Controller provides up to eight chip select lines: NCS0 to NCS7.

**Figure 41.** Memory Connections for Eight External Devices<sup>(1)</sup>



Note: 1. The maximum address space per device is 512M bytes

## 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 SMC\_CSR for the corresponding chip select. See “SMC Chip Select Registers” on page 186.

Figure 42 shows how to connect a 512K x 8-bit memory on NCS2 (DBW = 10).

**Figure 42.** Memory Connection for an 8-bit Data Path Device

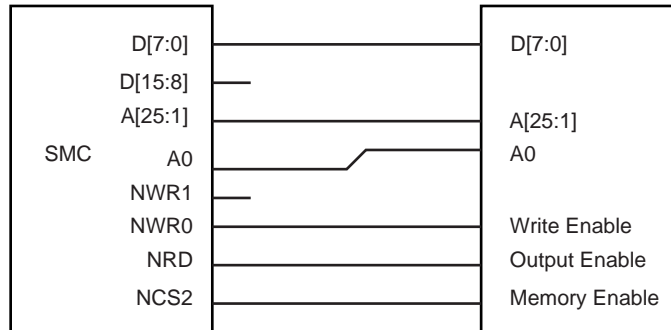
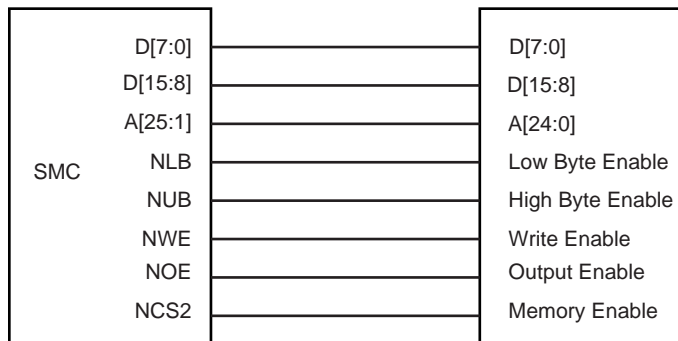


Figure 43 shows how to connect a 512K x 16-bit memory on NCS2 (DBW = 01).

**Figure 43.** Memory Connection for a 16-bit Data Path Device



## Write Access

### Write Access Type

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 SMC\_CSR for the corresponding chip select. See “SMC Chip Select Registers” on page 186.

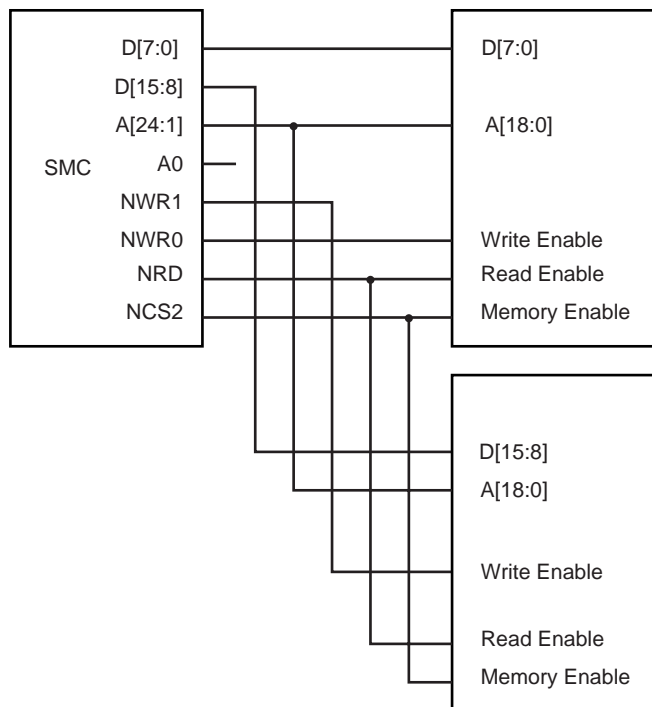
### Byte Write Access

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 44 shows how to connect two 512K x 8-bit devices in parallel on NCS2 (BAT = 0)

**Figure 44.** Memory Connection for 2 x 8-bit Data Path Devices



### Byte Select Access

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 45 shows how to connect a 16-bit device with byte and half-word access (e.g., SRAM device type) on NCS2 (BAT = 1).

**Figure 45.** Connection to a 16-bit Data Path Device with Byte and Half-word Access

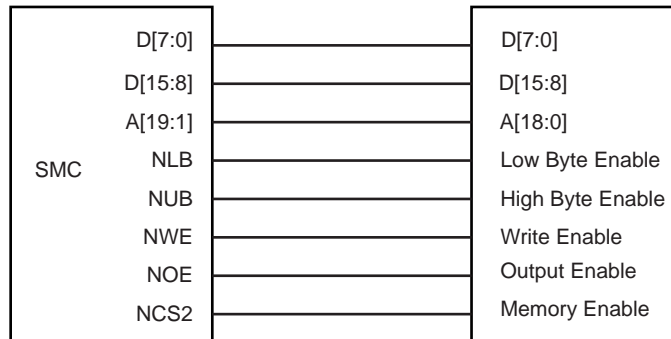
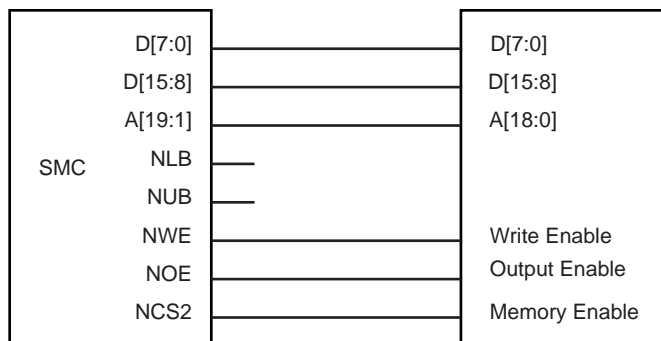


Figure 46 shows how to connect a 16-bit device without byte access (e.g., Flash device type) on NCS2 (BAT = 1).

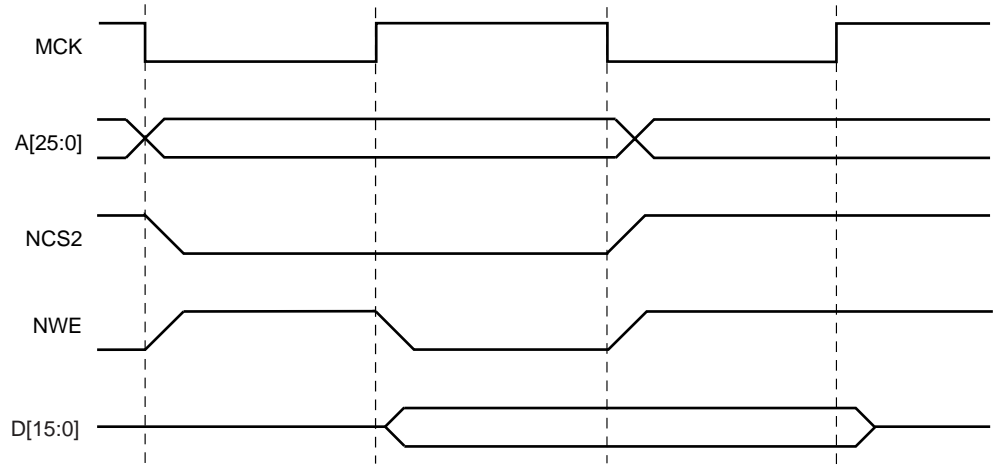
**Figure 46.** Connection to a 16-bit Data Path Device without Byte Write Capability



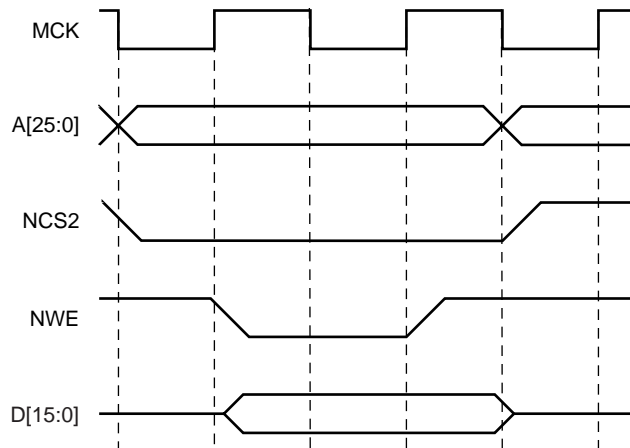
### Write Data Hold Time

During write cycles, data output becomes valid after the rising edge of MCK and remains valid after the rising edge of NWE. During a write access, the data remain on the bus 1/2 period of MCK after the rising edge of NWE. See Figure 47 and Figure 48.

**Figure 47.** Write Access with 0 Wait State



**Figure 48.** Write Access with 1 Wait State



## Read Access

### Read Protocols

The SMC provides two alternative protocols for external memory read accesses: standard and early read. The difference between the two protocols lies in the behavior of the NRD signal.

For write accesses, in both protocols, NWE has the same behavior. In the second half of the master clock cycle, NWE always goes low (see Figure 56 on page 164).

The protocol is selected by the DRP field in SMC\_CSR (See “SMC Chip Select Registers” on page 186.). Standard read protocol is the default protocol after reset.

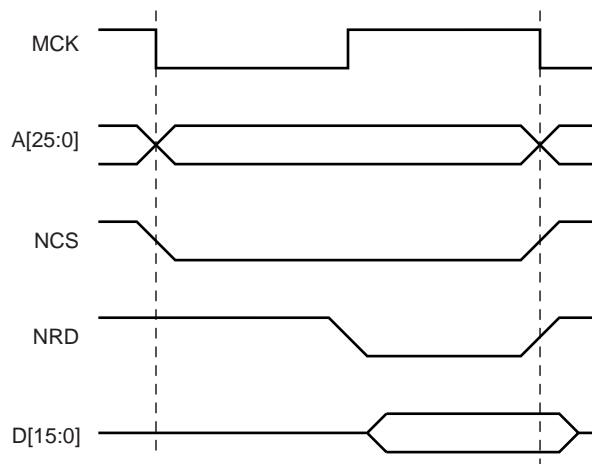
Note: In the following waveforms and descriptions, NRD represents NRD as well as NOE since the two signals have the same waveform. Likewise, NWE represents NWE, NWR0 and NWR1 unless NWR0 and NWR1 are otherwise represented. In addition, NCS represents NCS[7:0] (see “I/O Lines” on page 153, Table 44 and Table 45).

#### Standard Read Protocol

Standard read protocol implements a read cycle during 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 lines and NCS before the read cycle begins.

During a standard read protocol, NCS is set low and address lines are valid at the beginning of the external memory access, while NRD goes low only in the second half of the master clock cycle to avoid bus conflict. See Figure 49.

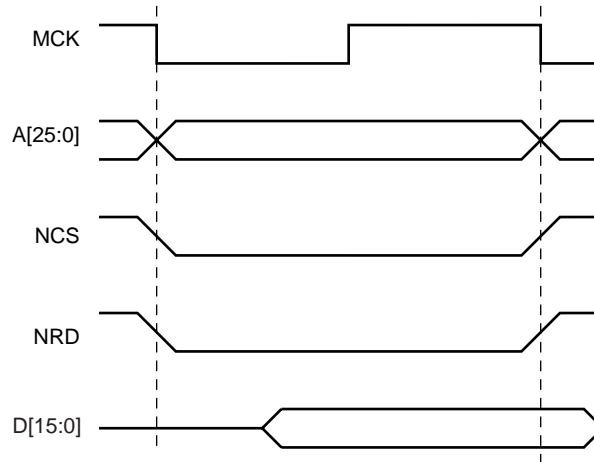
**Figure 49.** Standard Read Protocol



*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 50.** Early Read Protocol





## Wait State Management

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

- Standard wait states
- External wait states
- Data float wait states
- Chip select change wait states
- Early Read wait states

### Standard Wait States

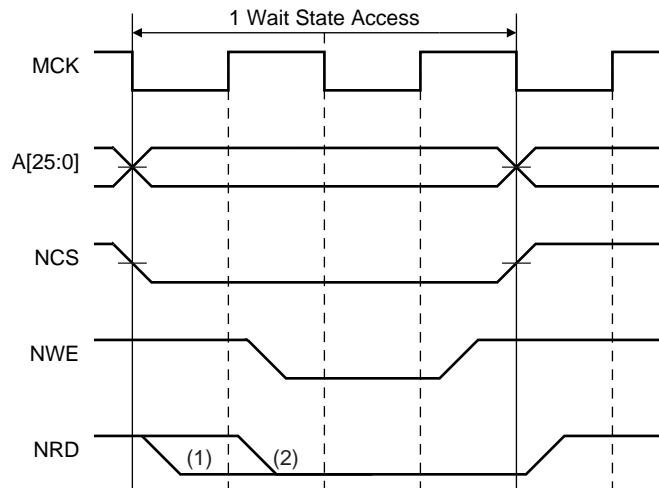
Each chip select can be programmed to insert one or more wait states during an access on the corresponding memory area. This is done by setting the WSEN field in the corresponding SMC\_CSR (See “SMC Chip Select Registers” on page 186.). The number of cycles to insert is programmed in the NWS field in the same register.

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

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

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

**Figure 51.** One Standard Wait State Access



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

### External Wait States

The NWAIT input pin is used to insert wait states beyond the maximum standard wait states programmable or in addition to. If NWAIT is asserted low, then the SMC adds a wait state and no changes are made to the output signals, the internal counters or the state. When NWAIT is de-asserted, the SMC completes the access sequence.

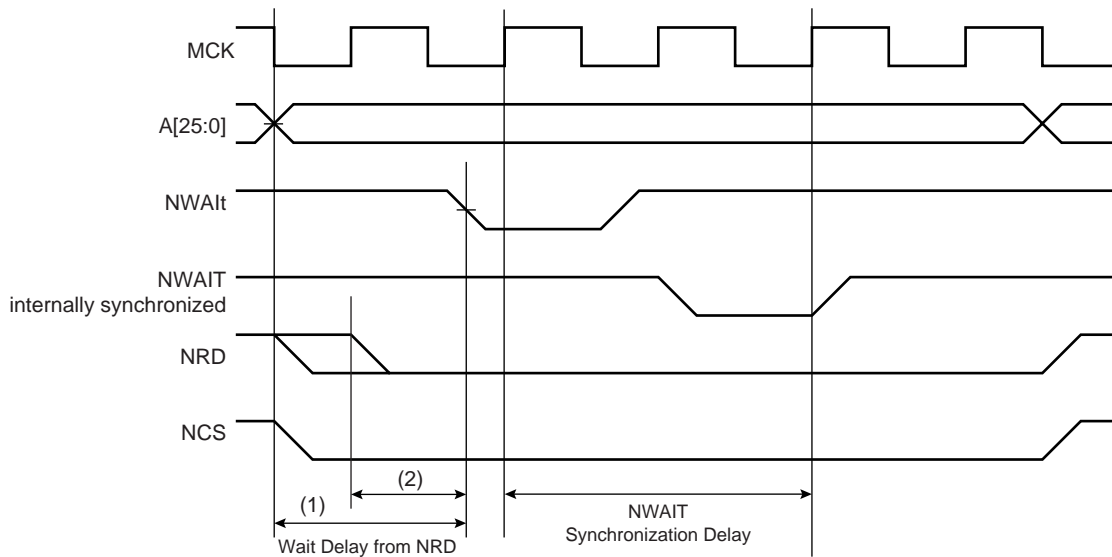
The input of the NWAIT signal is an asynchronous input. To avoid any metastability problems, NWAIT is synchronized before using it. This operation results in a two-cycle delay.

NWS must be programmed as a function of synchronization time and delay between NWAIT falling and control signals falling (NRD/NWE), otherwise SMC will not function correctly.

$$NWS > \text{Wait Delay from nrd/nwe} + \text{external\_nwait Synchronization Delay} + 1$$

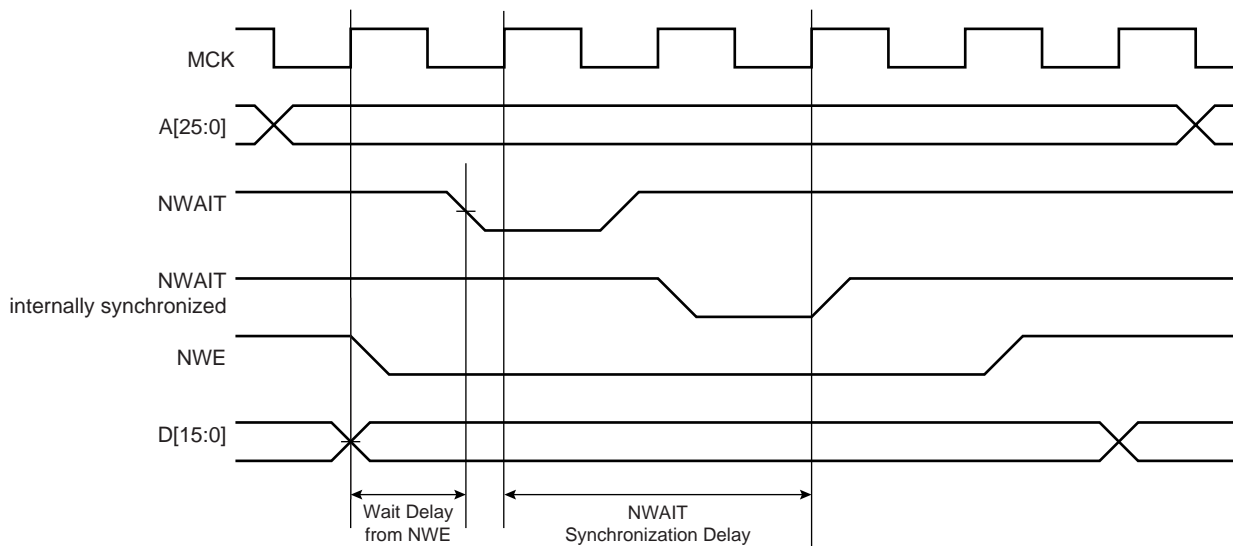
If NWAIT is asserted during a setup or hold timing, the SMC does not function correctly.

**Figure 52. NWAIT behaviour in Read Access**



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

**Figure 53. NWAIT behaviour in Write Access**



**Data Float Wait States**

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

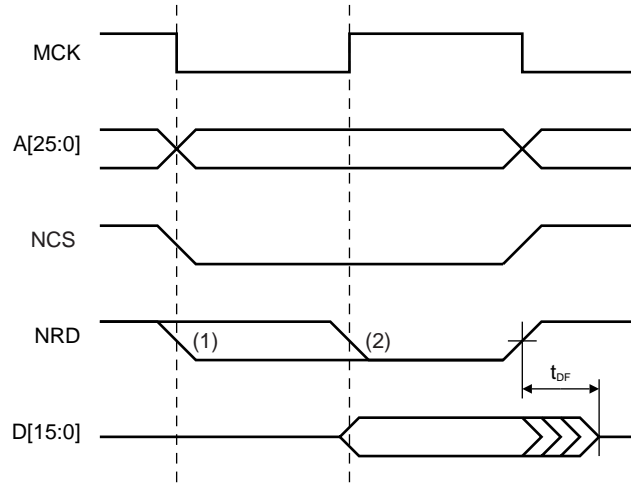
The Data Float Output Time ( $t_{DF}$ ) for each external memory device is programmed in the TDF field of the SMC\_CSR register for the corresponding chip select (See “SMC Chip Select Registers” on page 186.). The value of TDF indicates the number of data float wait cycles (between 0 and 15) to be inserted and represents the time allowed for the data output to go to 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.

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

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

**Figure 54.** Data Float Output Delay

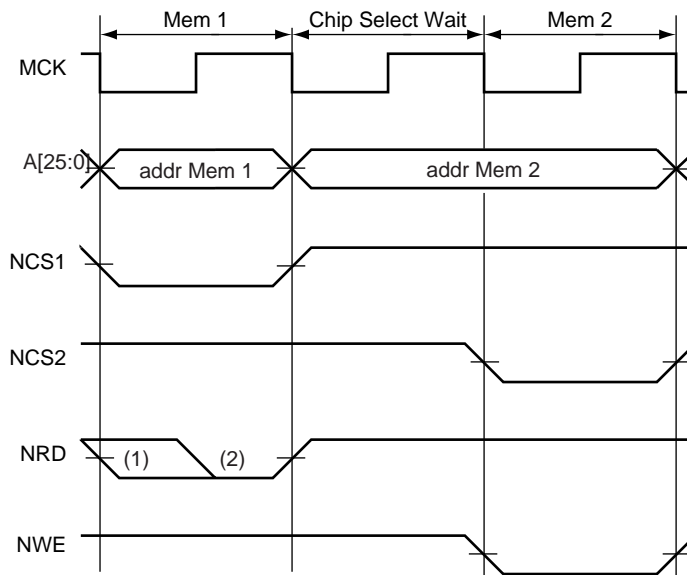


- Notes: 1. Early Read Protocol  
2. Standard Read Protocol

**Chip Select Change Wait State**

A chip select wait state is automatically inserted when consecutive accesses are made to two different external memories (if no other type of wait state has already been inserted). If a wait state has already been inserted (e.g., data float wait state), then no more wait states are added.

**Figure 55.** Chip Select Wait State



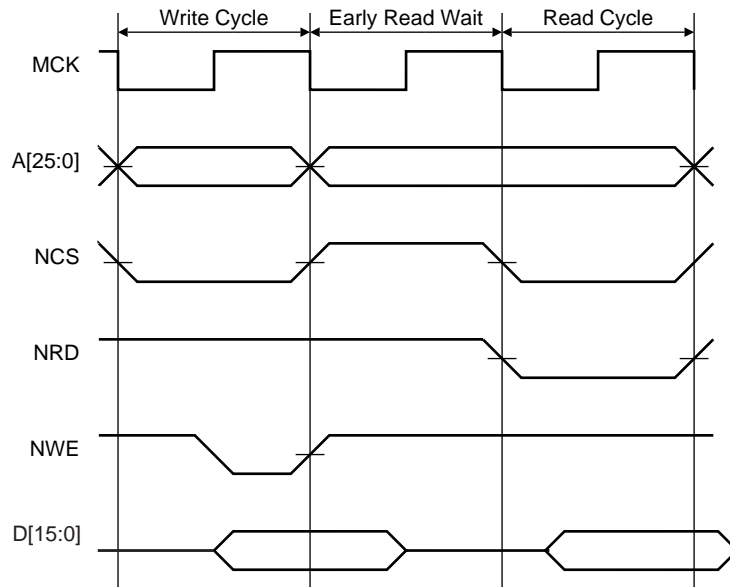
- Notes: 1. Early Read Protocol  
2. Standard Read Protocol

## 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 56). This wait state is generated in addition to any other programmed wait states (i.e., data float wait state).

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.

**Figure 56.** Early Read Wait States



## Setup and Hold Cycles

The SMC allows some memory devices to be interfaced with different setup, hold and pulse delays. These parameters are programmable and define the timing of each portion of the read and write cycles. However, it is not possible to use this feature in early read protocol.

If an attempt is made to program the setup parameter as not equal to zero and the hold parameter as equal to zero with  $WSEN = 0$  (0 standard wait state), the SMC does not operate correctly.

If consecutive accesses are made to two different external memories and the second memory is programmed with setup cycles, then no chip select change wait state is inserted (see Figure 61 on page 166).

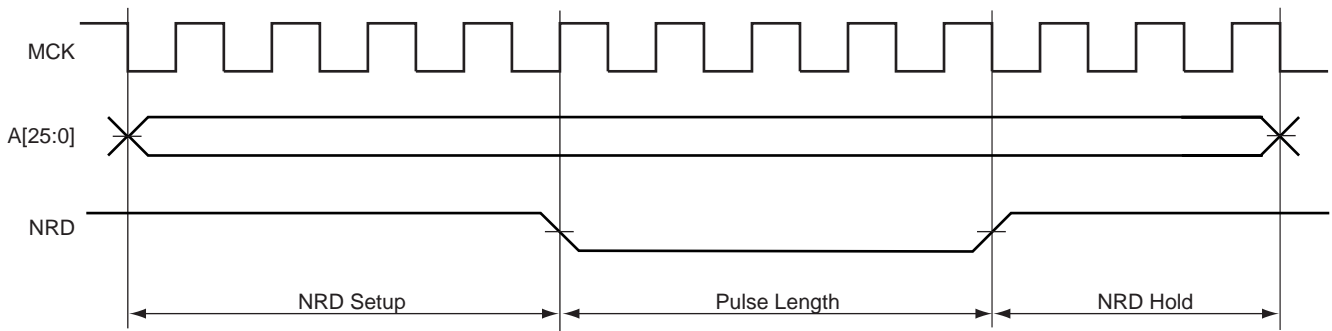
When a data float wait state ( $t_{DF}$ ) is programmed on the first memory bank and when the second memory bank is programmed with setup cycles, the SMC behaves as follows:

- If the number of  $t_{DF}$  is higher or equal to the number of setup cycles, the number of setup cycles inserted is equal to 0 (see Figure 62 on page 167).
- If the number of the setup cycle is higher than the number of  $t_{DF}$ , the number of  $t_{DF}$  inserted is 0 (see Figure 63 on page 167).

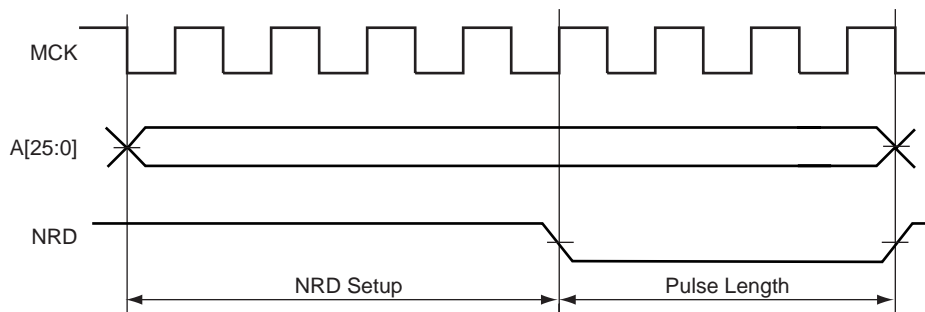
## Read Access

The read cycle can be divided into a setup, a pulse length and a hold. The setup parameter can have a value between 1.5 and 7.5 clock cycles, the hold parameter between 0 and 7 clock cycles and the pulse length between 1.5 and 128.5 clock cycles, by increments of one.

**Figure 57.** Read Access with Setup and Hold



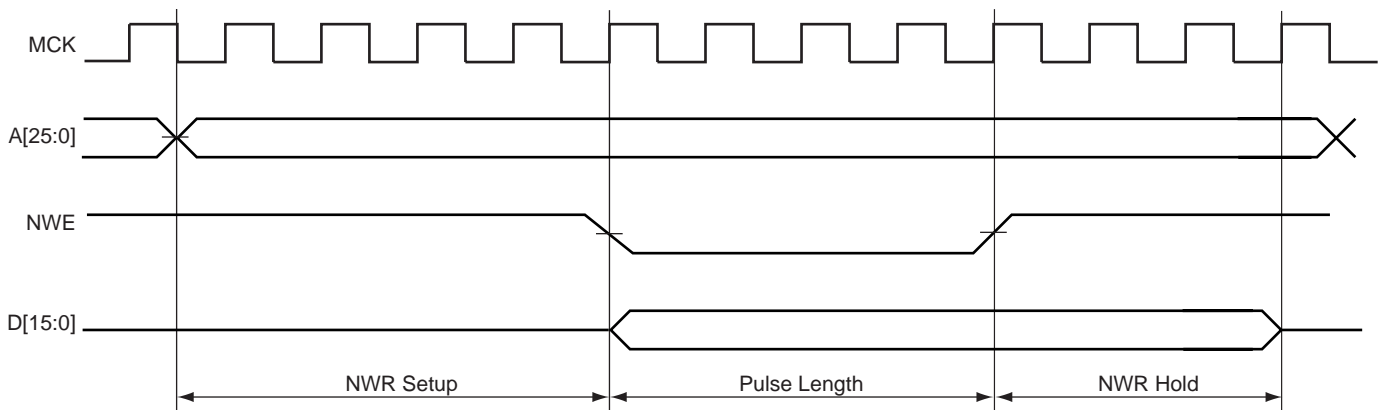
**Figure 58.** Read Access with Setup



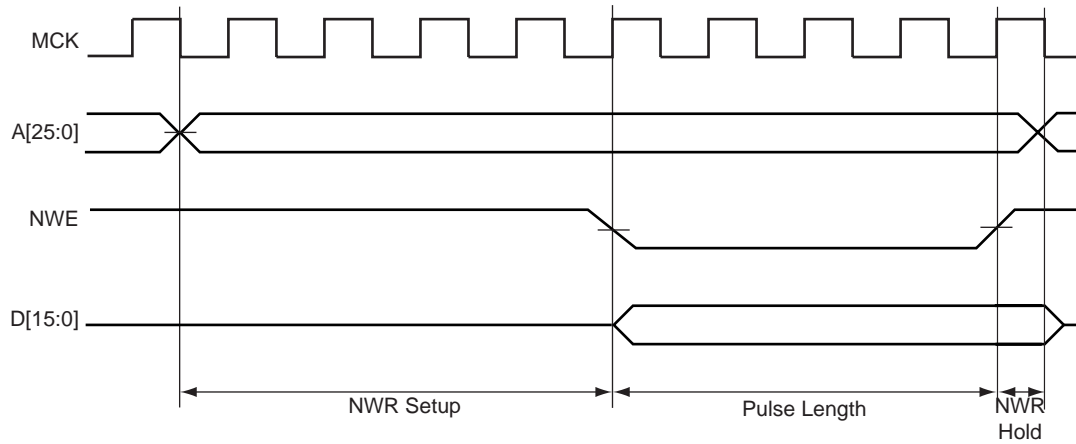
**Write Access**

The write cycle can be divided into a setup, a pulse length and a hold. The setup parameter can have a value between 1.5 and 7.5 clock cycles, the hold parameter between 0.5 and 7 clock cycles and the pulse length between 1 and 128 clock cycles by increments of one.

**Figure 59.** Write Access with Setup and Hold

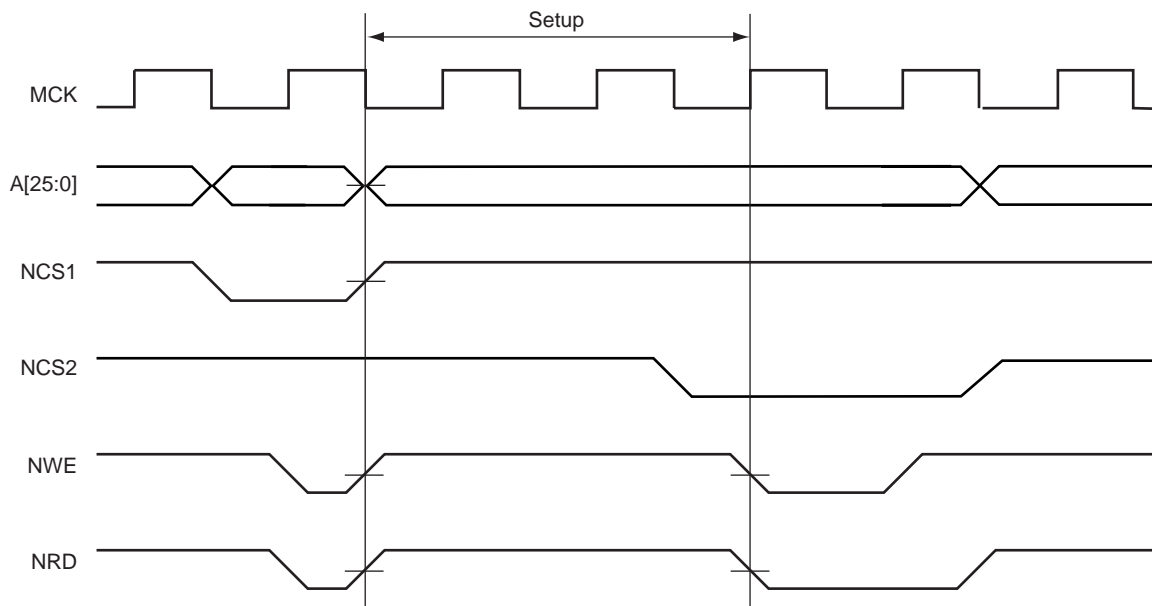


**Figure 60. Write Access with Setup**

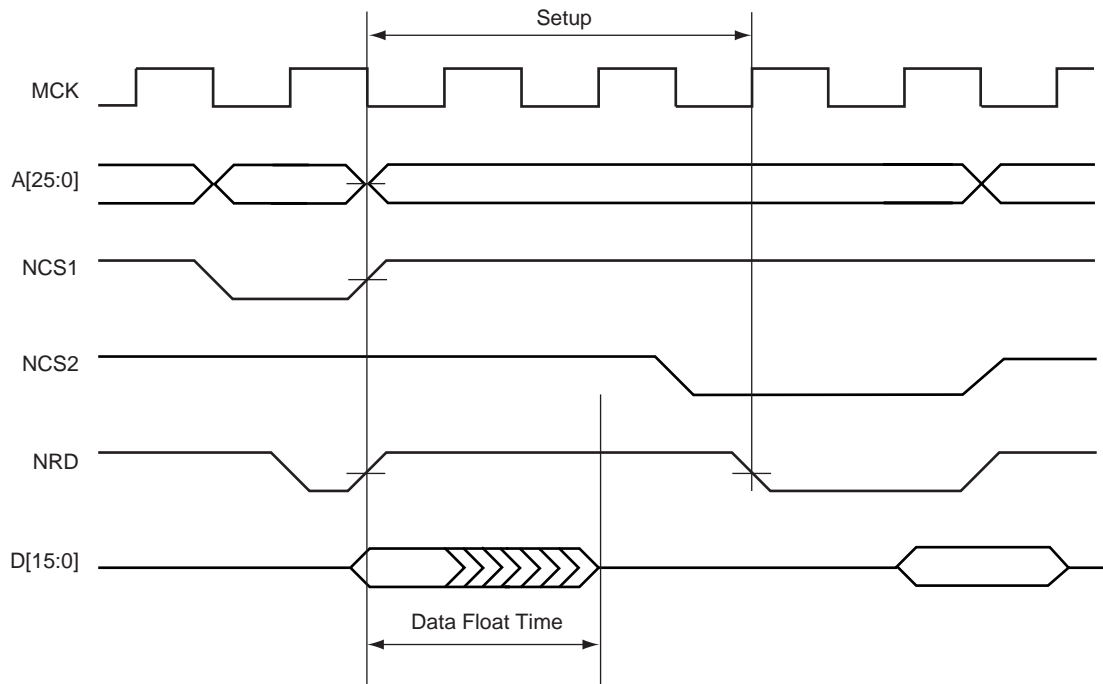


**Data Float Wait States with Setup Cycles**

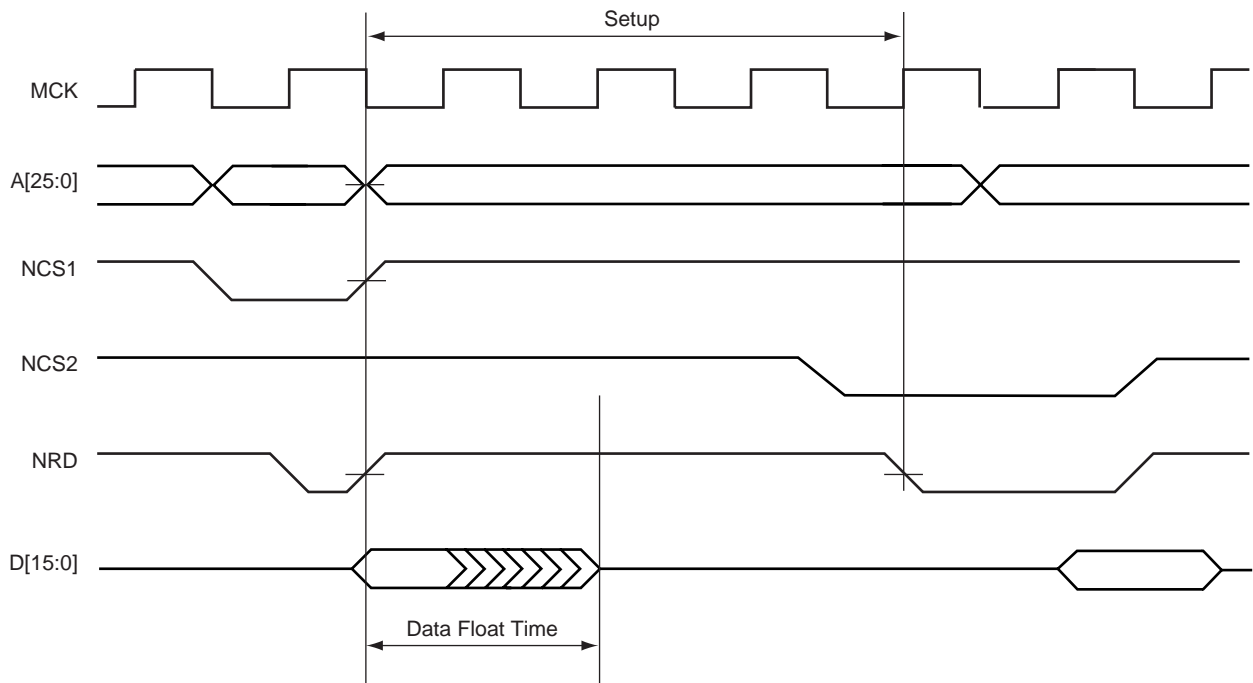
**Figure 61. Consecutive Accesses with Setup Programmed on the Second Access**



**Figure 62.** First Access with Data Float Wait States (TDF = 2) and Second Access with Setup (NRDSETUP = 1)



**Figure 63.** First Access with Data Float Wait States (TDF = 2) and Second Access with Setup (NRDSETUP = 3)



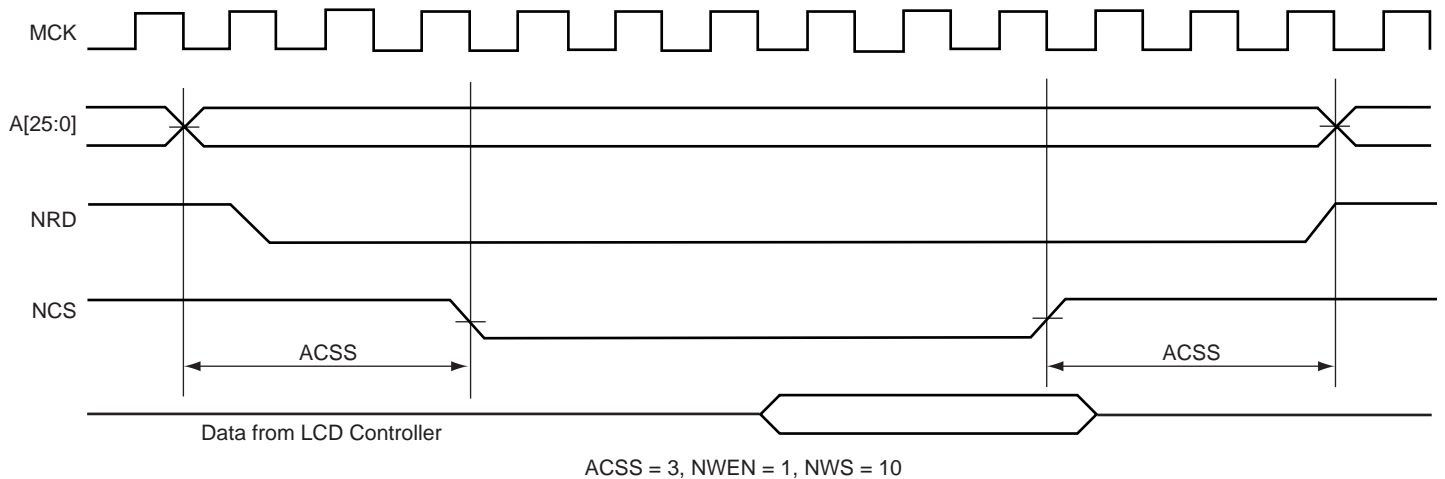
## LCD Interface Mode

The SMC can be configured to work with an external liquid crystal display (LCD) controller by setting the ACSS (Address to Chip Select Setup) bit in the SMC\_CSR registers (See “SMC Chip Select Registers” on page 186.).

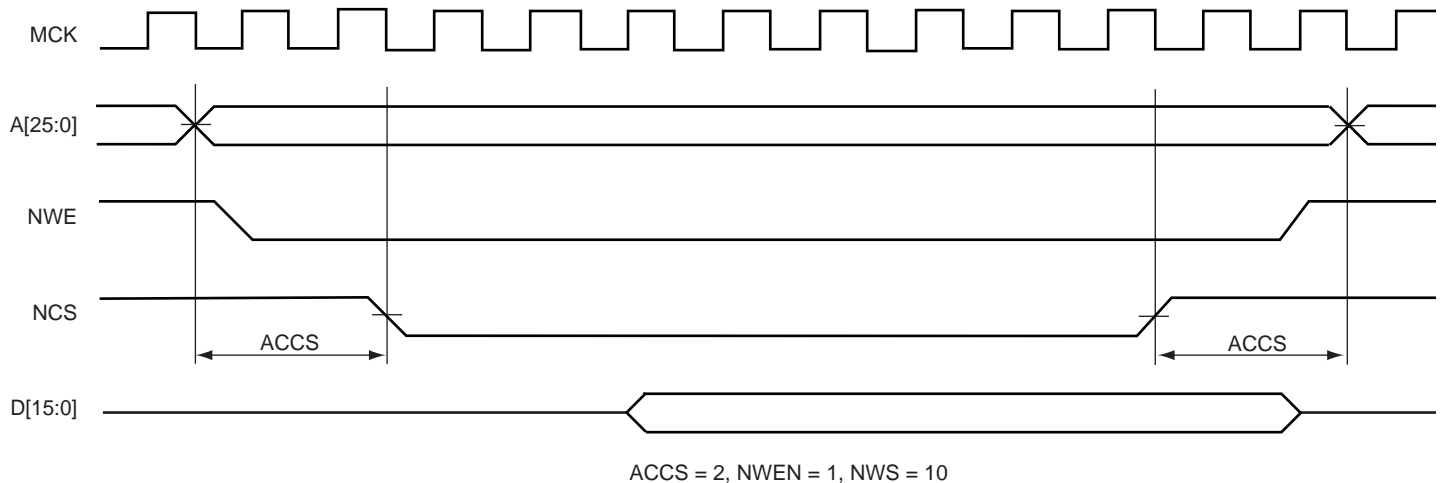
In LCD mode, NCS is shortened by one/two/three clock cycles at the leading and trailing edges, providing positive address setup and hold. For read accesses, the data is latched in the SMC when NCS is raised at the end of the access.

Additionally, WSEN must be set and NWS programmed with a value of two or more superior to ACSS. In LCD mode, it is not recommended to use RWHOLD or RWSETUP. If the above conditions are not satisfied, SMC does not operate correctly.

**Figure 64.** Read Access in LCD Interface Mode



**Figure 65.** Write Access in LCD Interface Mode



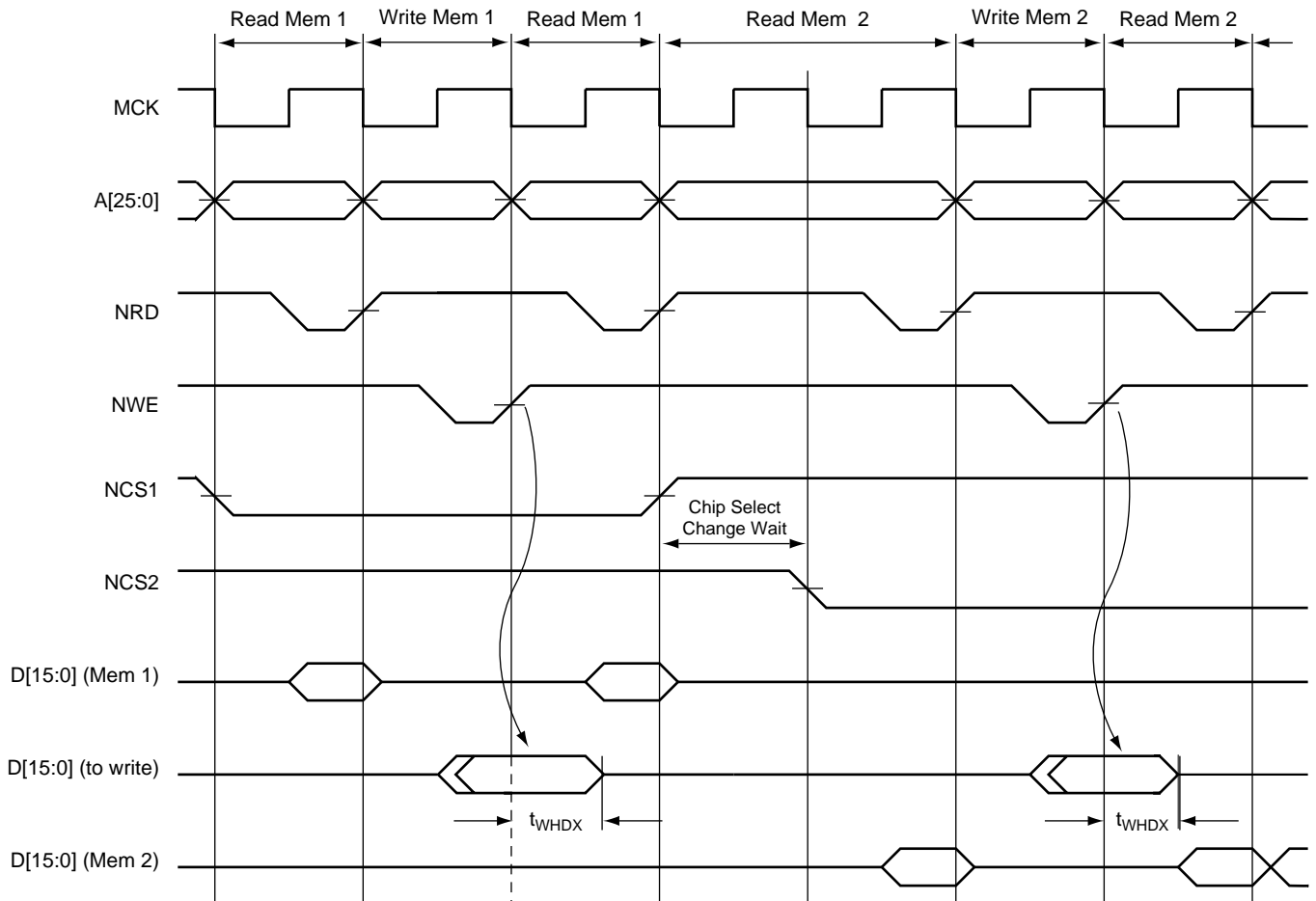


## Memory Access Waveforms

### Read Accesses in Standard and Early Protocols

Figure 66 on page 169 through Figure 69 on page 172 show examples of the alternatives for external memory read protocol.

**Figure 66.** Standard Read Protocol without  $t_{DF}$



**Figure 67. Early Read Protocol without  $t_{DF}$**

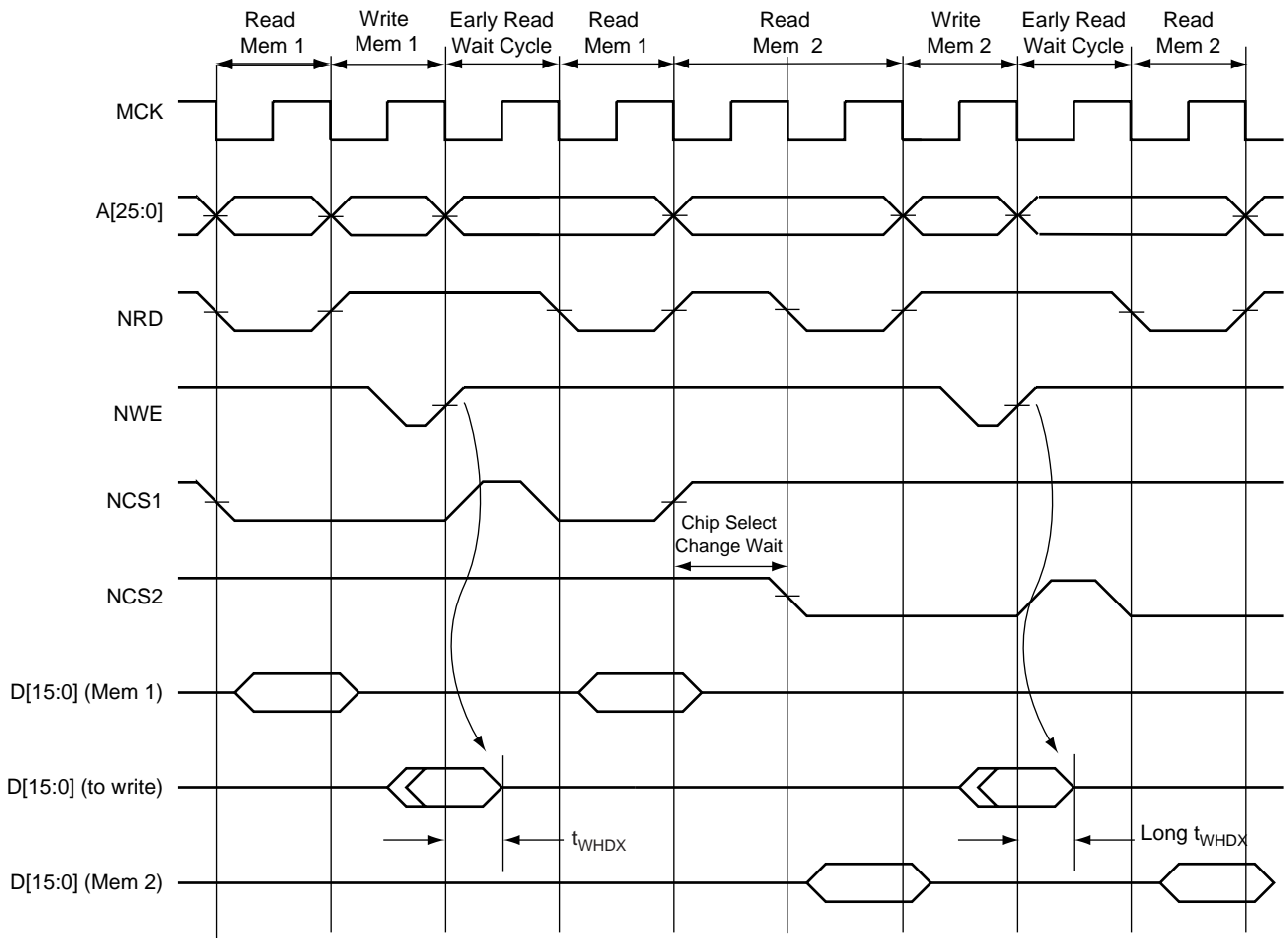
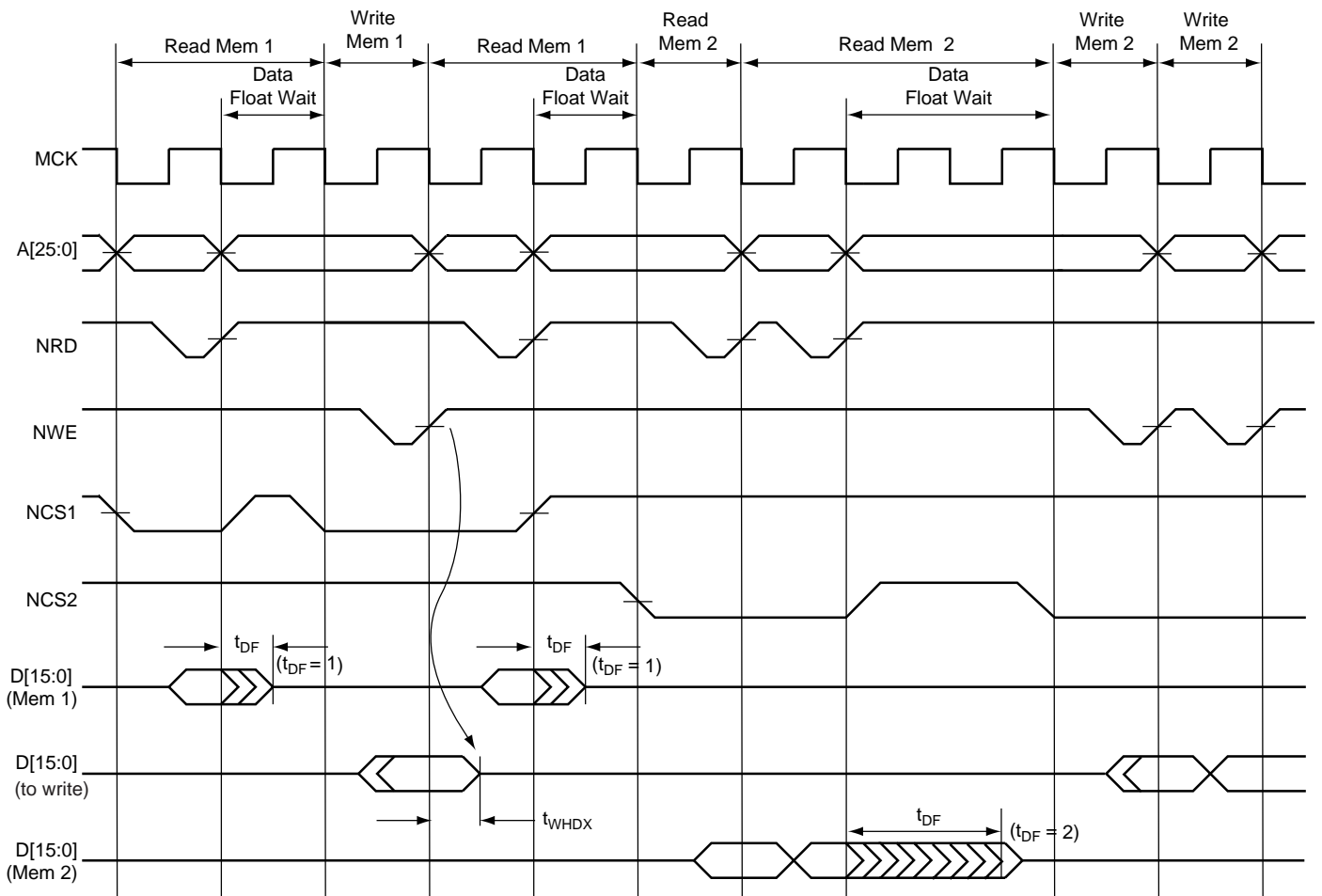
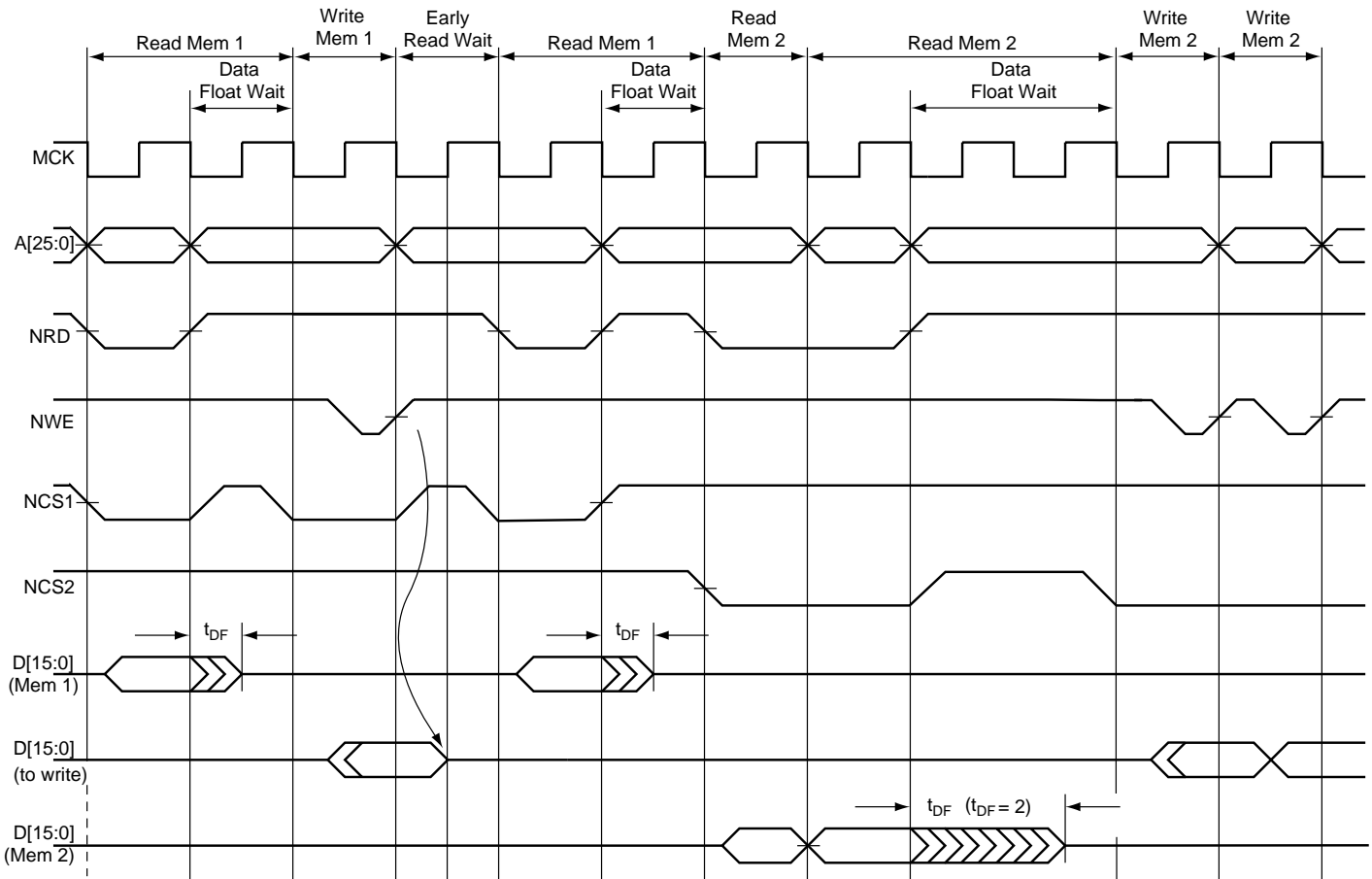


Figure 68. Standard Read Protocol with  $t_{DF}$



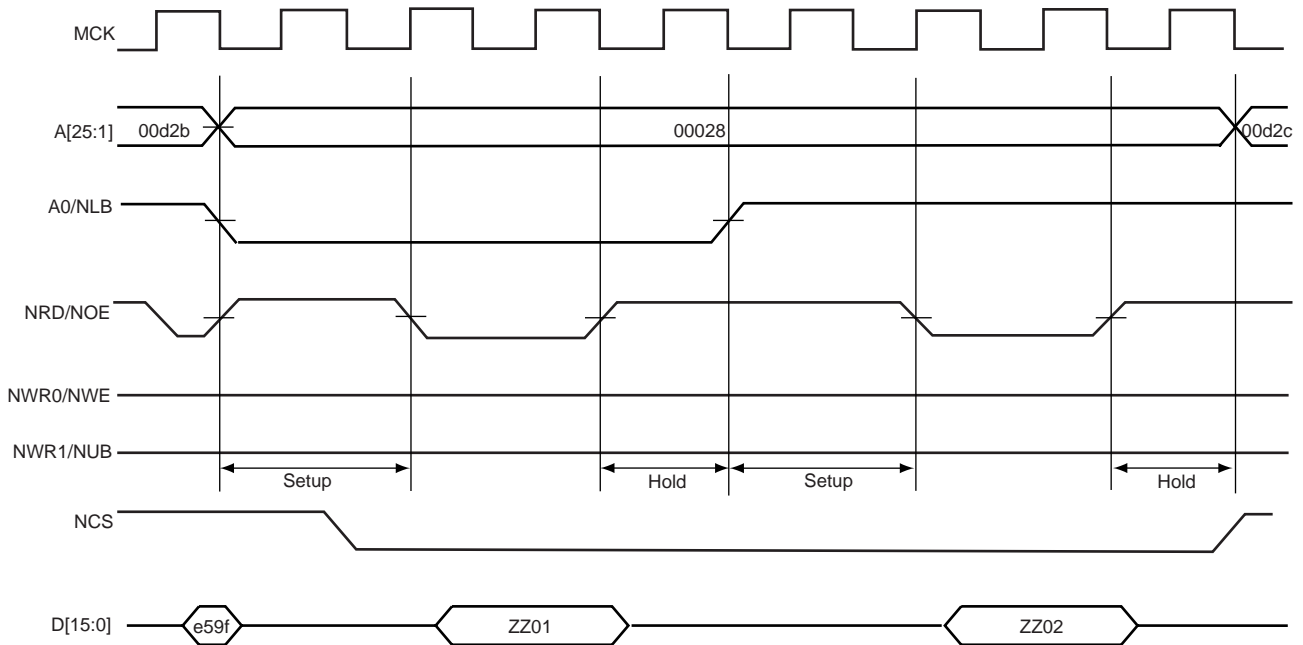
**Figure 69. Early Read Protocol with  $t_{DF}$**



## Accesses with Setup and Hold

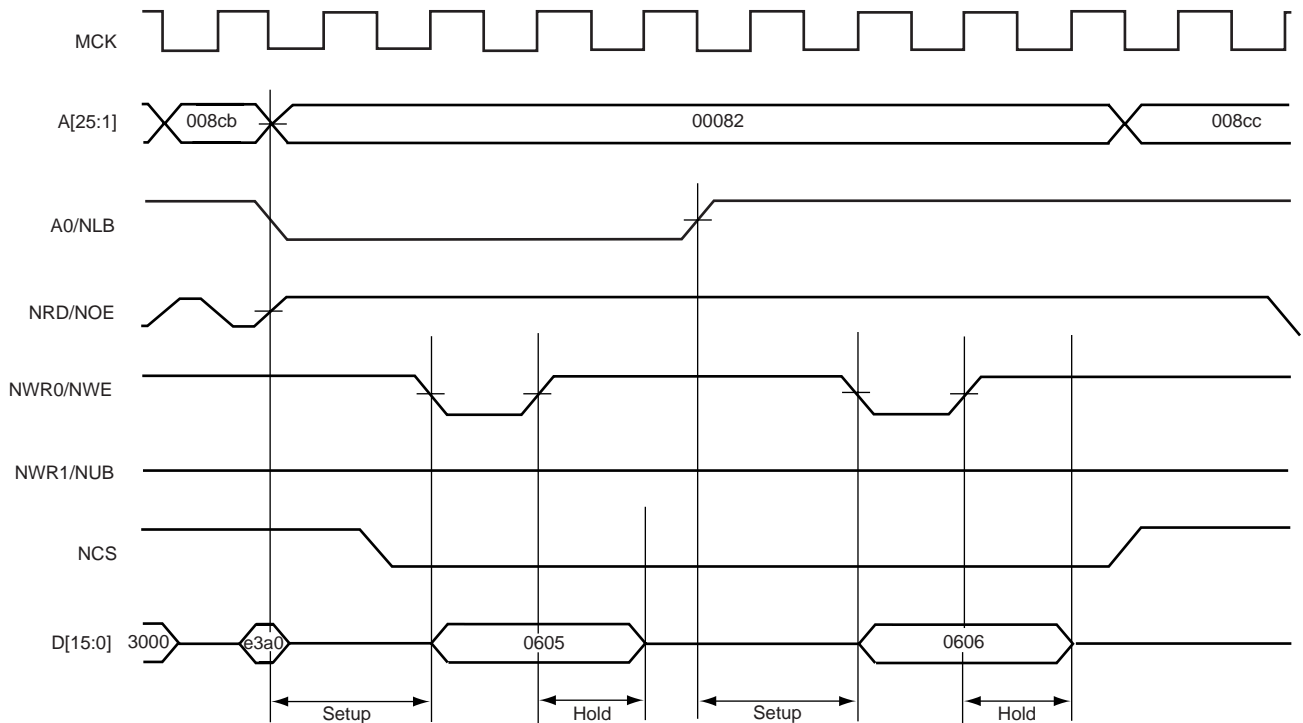
Figure 70 and Figure 71 show an example of read and write accesses with Setup and Hold Cycles.

**Figure 70.** Read Accesses in Standard Read Protocol with Setup and Hold<sup>(1)</sup>



Note: 1. Read access memory data bus width = 8, RWSETUP = 1, RWHOLD = 1, WSEN = 1, NWS = 0

**Figure 71.** Write Accesses with Setup and Hold<sup>(1)</sup>

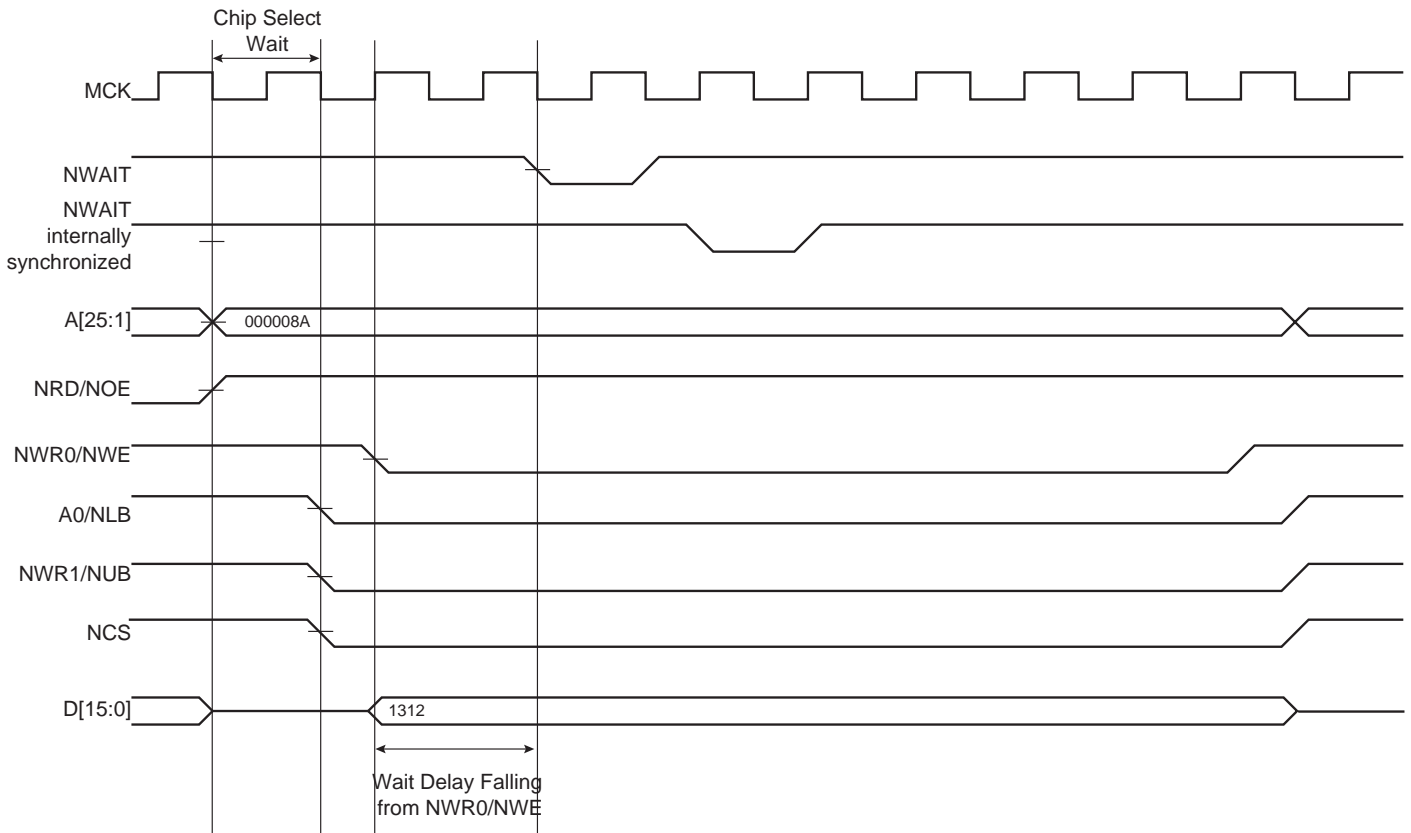


Note: 1. Write access, memory data bus width = 8, RWSETUP = 1, RWHOLD = 1, WSEN = 1, NWS = 0

### Accesses Using NWAIT Input Signal

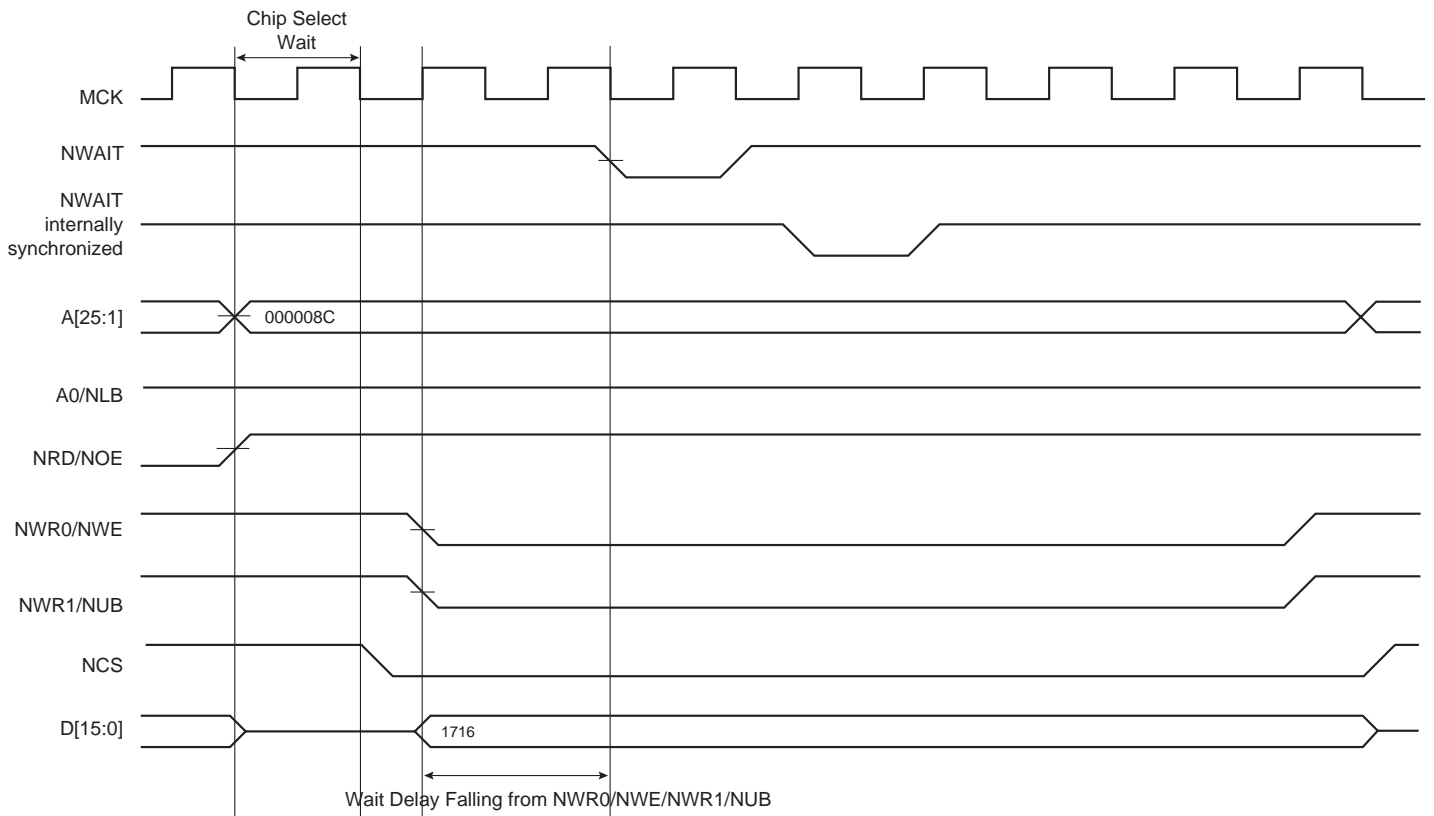
Figure 72 on page 174 through Figure 75 on page 177 show examples of accesses using NWAIT.

**Figure 72.** Write Access using NWAIT in Byte Select Type Access<sup>(1)</sup>



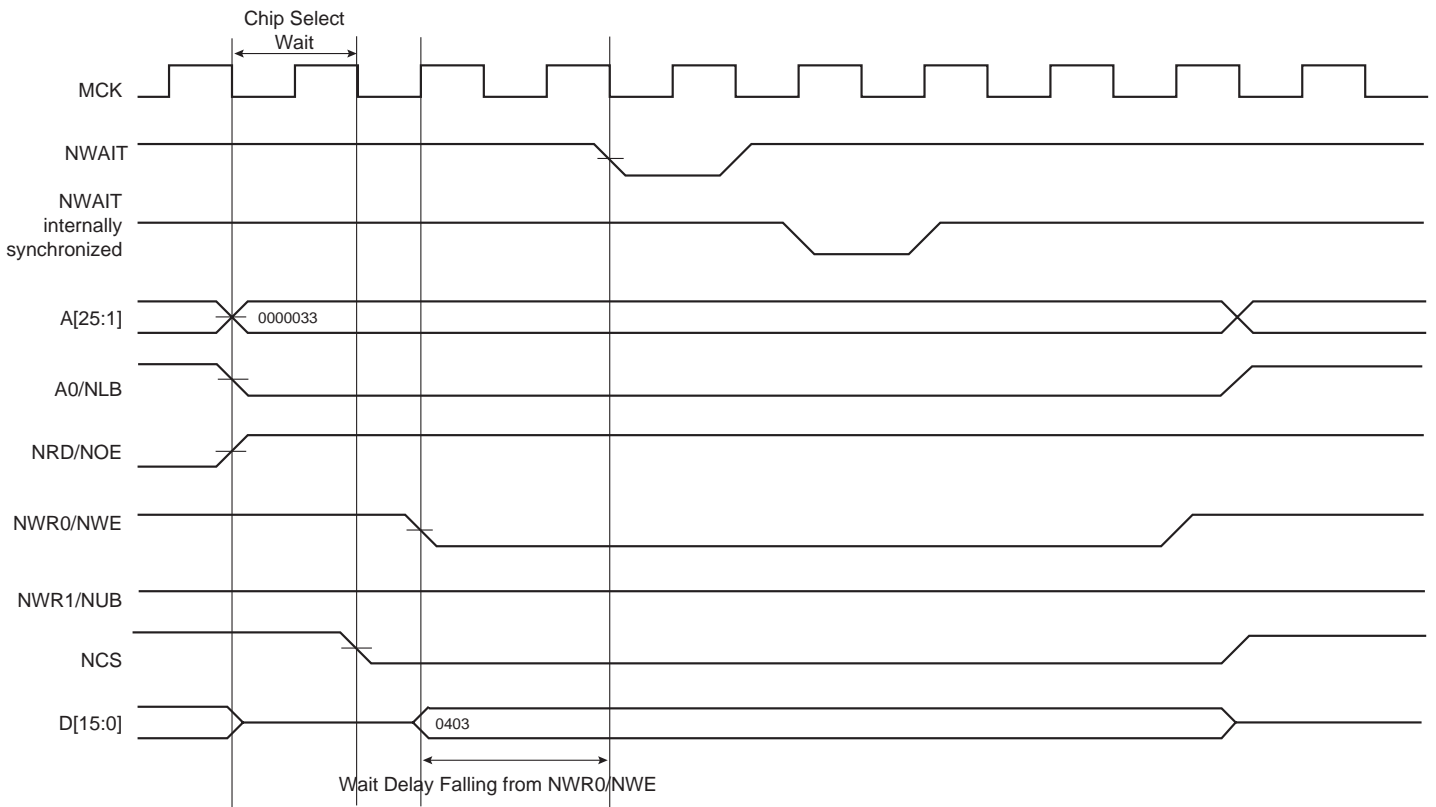
Note: 1. Write access memory, data bus width = 16 bits, WSEN = 1, NWS = 6

**Figure 73.** Write Access using NWAIT in Byte Write Type Access<sup>(1)</sup>



Note: 1. Write access memory, data bus width = 16 bits, WSEN = 1, NWS = 5

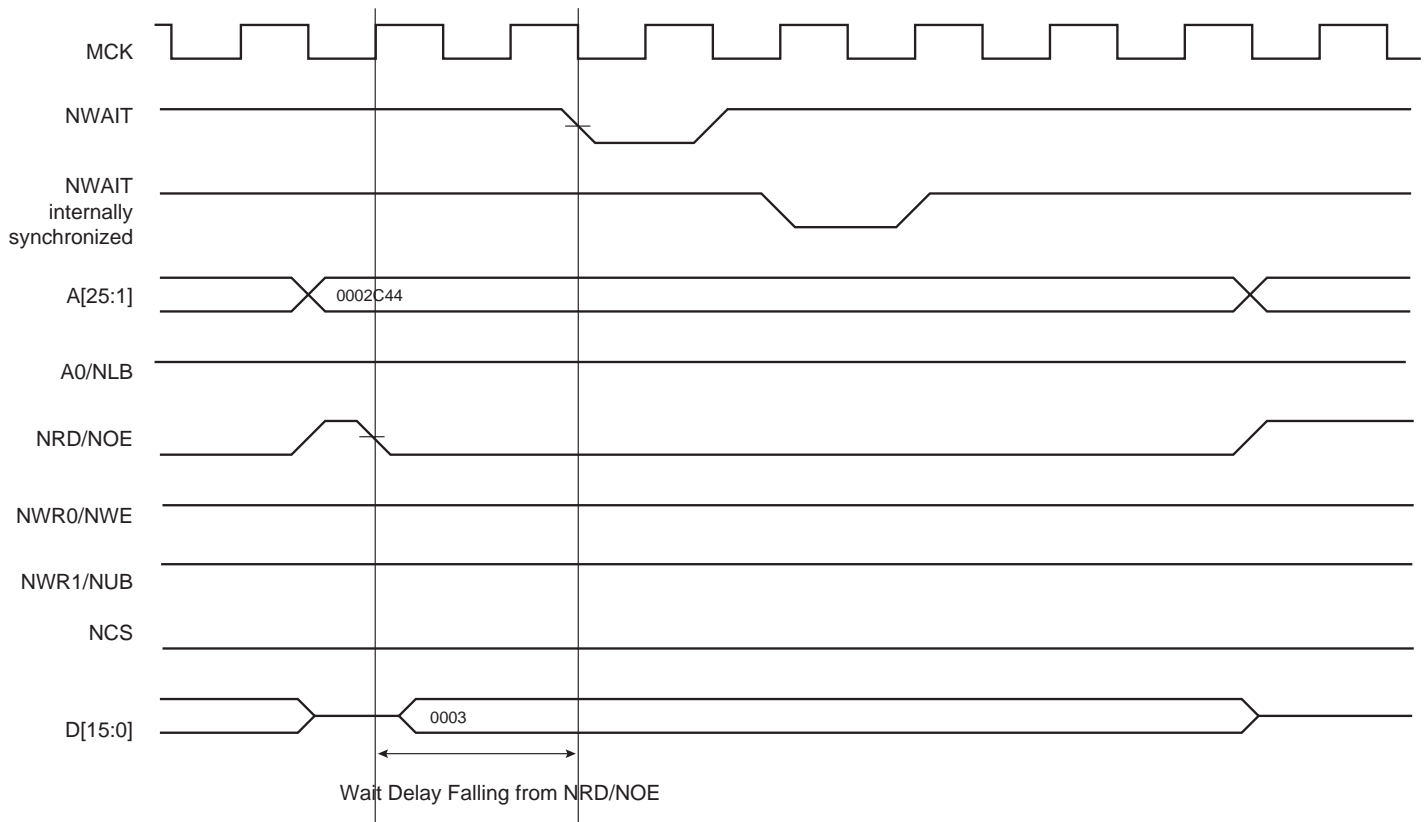
**Figure 74.** Write Access using NWAIT<sup>(1)</sup>



Note: 1. Write access memory, data bus width = 8 bits, WSEN = 1, NWS = 4



**Figure 75.** Read Access in Standard Protocol using NWAIT<sup>(1)</sup>



Note: 1. Read access, memory data bus width = 16, NWS = 5, WSEN = 1

## Memory Access Example Waveforms

Figure 76 on page 178 through Figure 82 on page 184 show the waveforms for read and write accesses to the various associated external memory devices. The configurations described are shown in Table 46.

**Table 46.** Memory Access Waveforms

| Figure Number | Number of Wait States | Bus Width | Size of Data Transfer |
|---------------|-----------------------|-----------|-----------------------|
| Figure 76     | 0                     | 16        | Word                  |
| Figure 77     | 1                     | 16        | Word                  |
| Figure 78     | 1                     | 16        | Half-word             |
| Figure 79     | 0                     | 8         | Word                  |
| Figure 80     | 1                     | 8         | Half-word             |
| Figure 81     | 1                     | 8         | Byte                  |
| Figure 82     | 0                     | 16        | Byte                  |

**Figure 76.** 0 Wait State, 16-bit Bus Width, Word Transfer

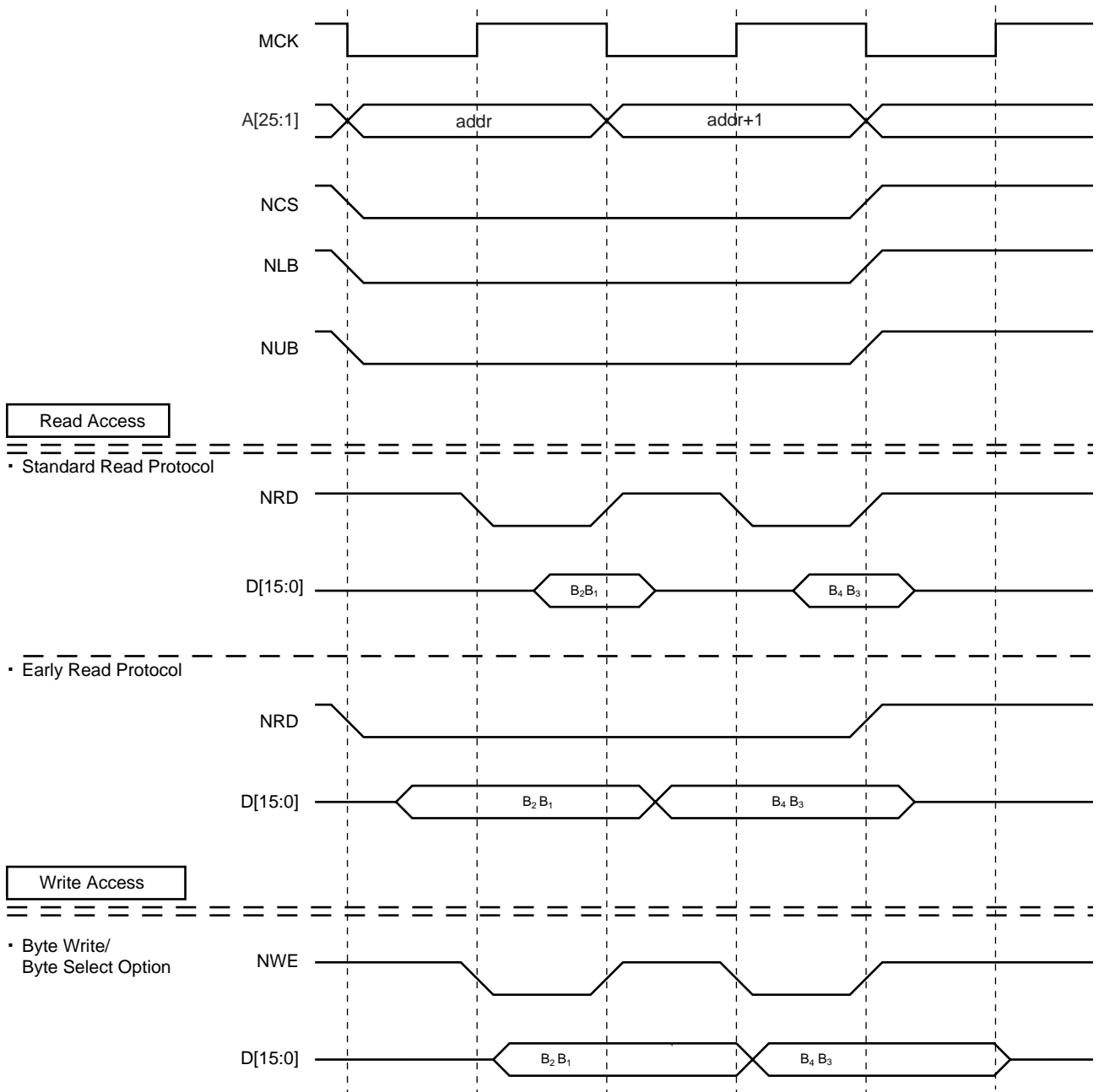
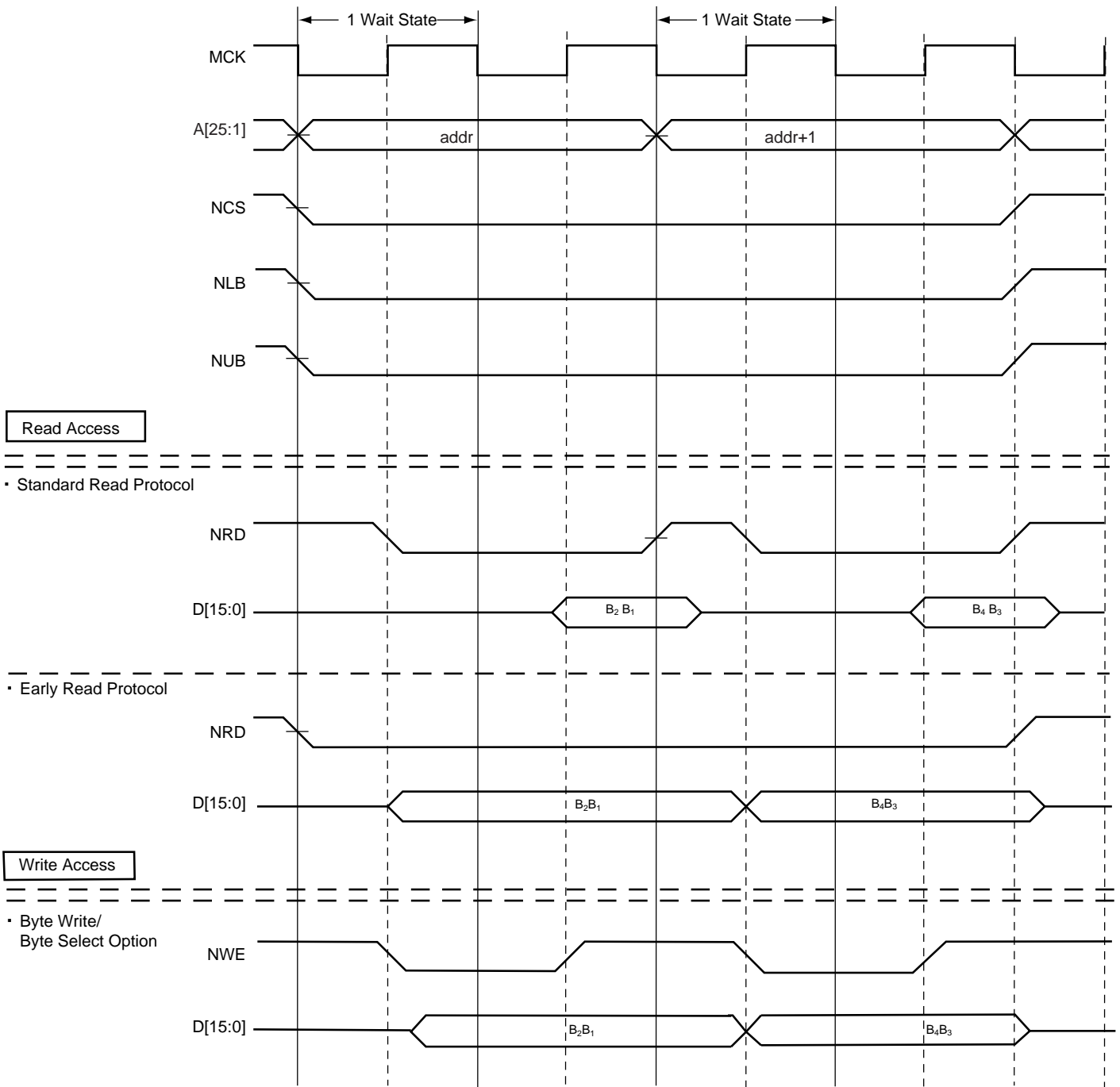


Figure 77. 1 Wait State, 16-bit Bus Width, Word Transfer



**Figure 78.** 1 Wait State, 16-bit Bus Width, Half-Word Transfer

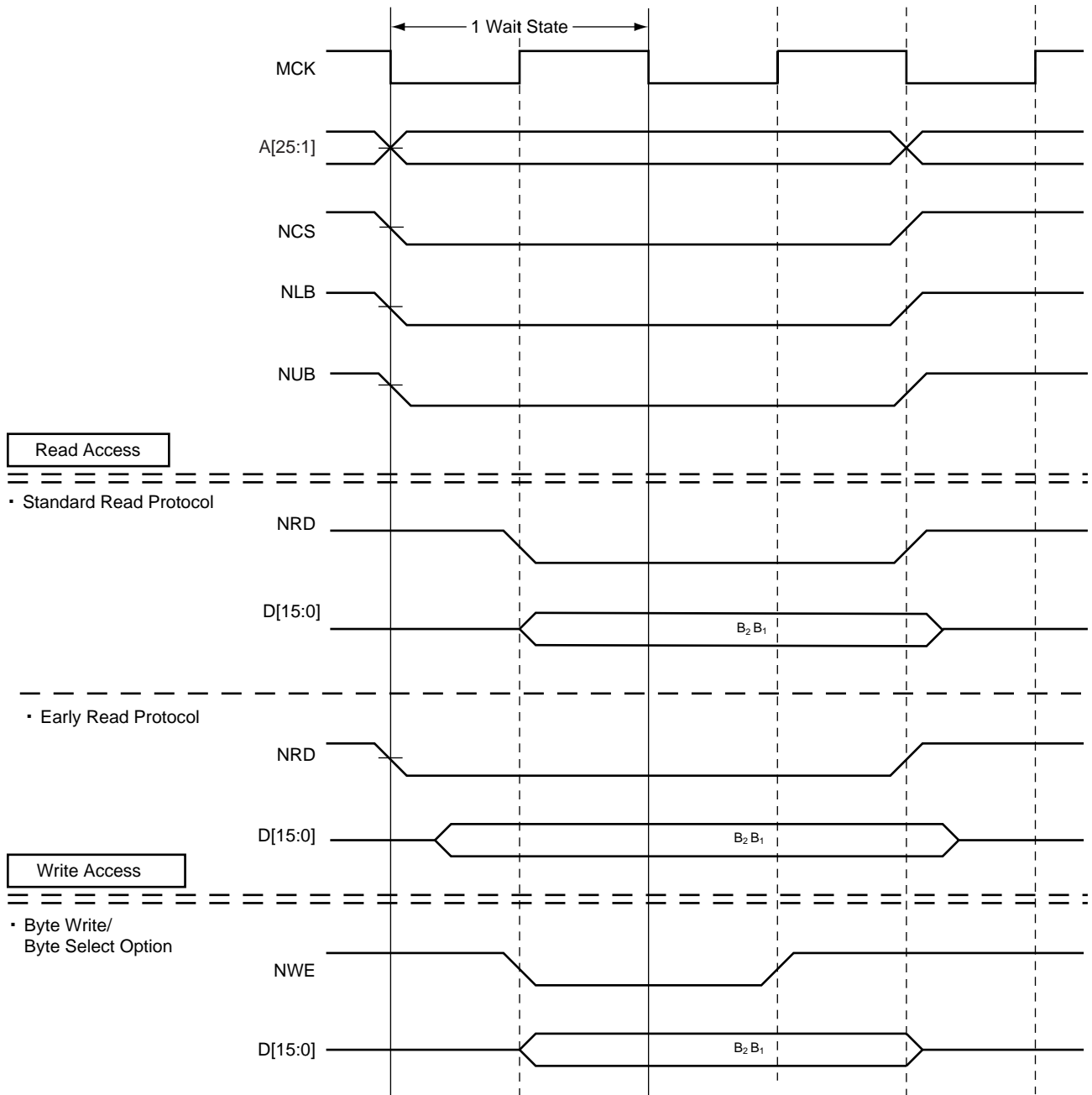
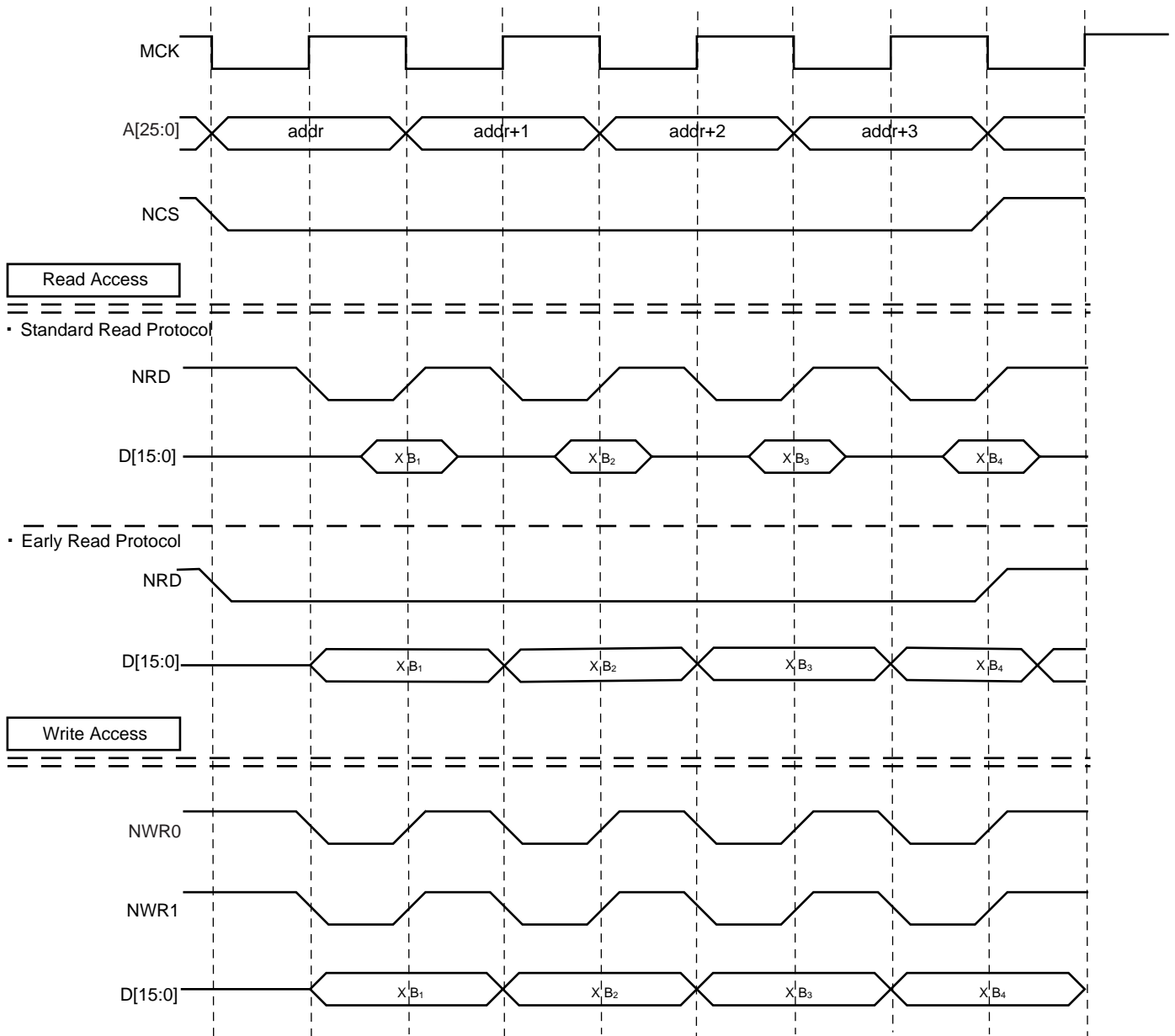


Figure 79. 0 Wait State, 8-bit Bus Width, Word Transfer



**Figure 80.** 1 Wait State, 8-bit Bus Width, Half-Word Transfer

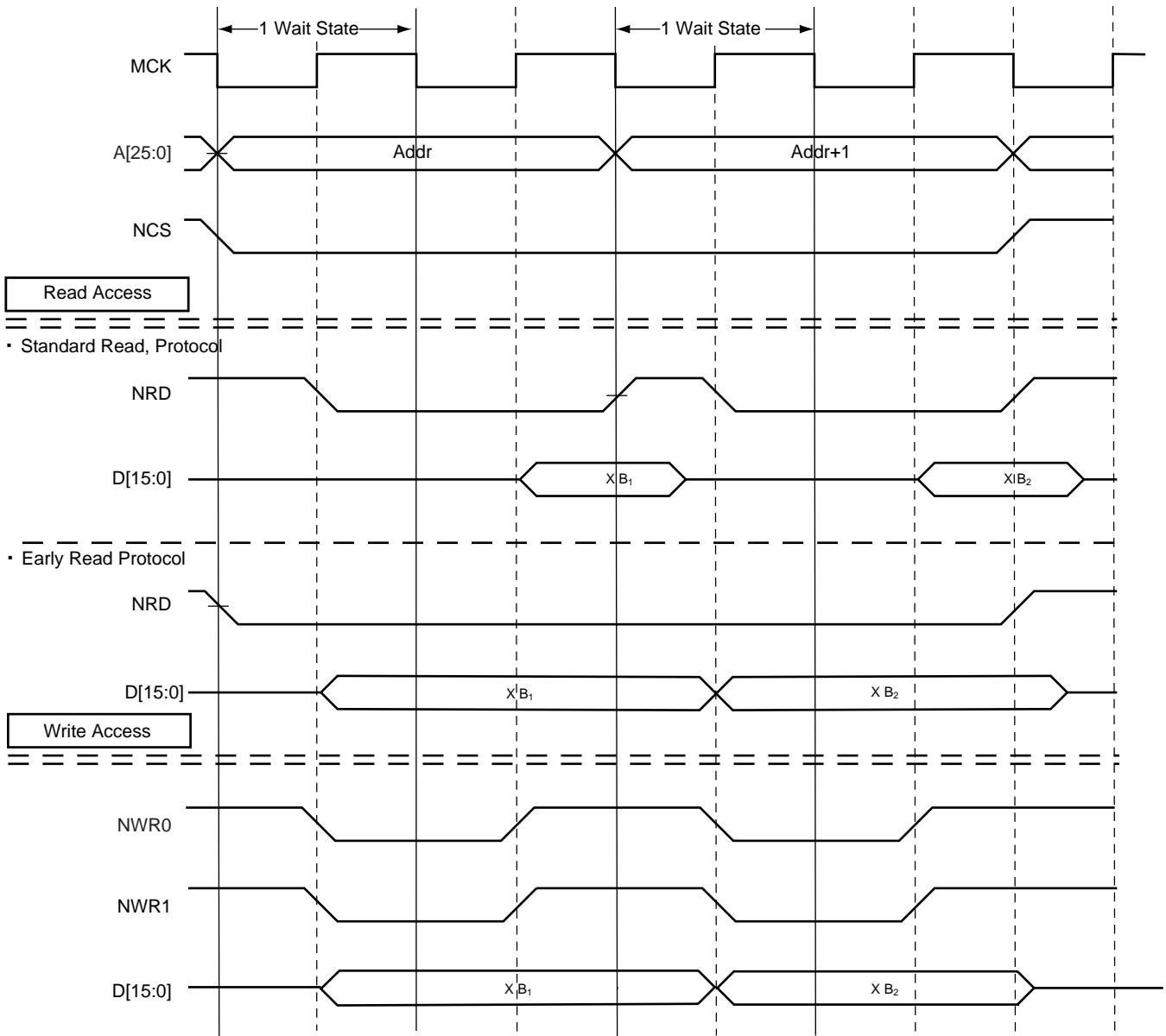
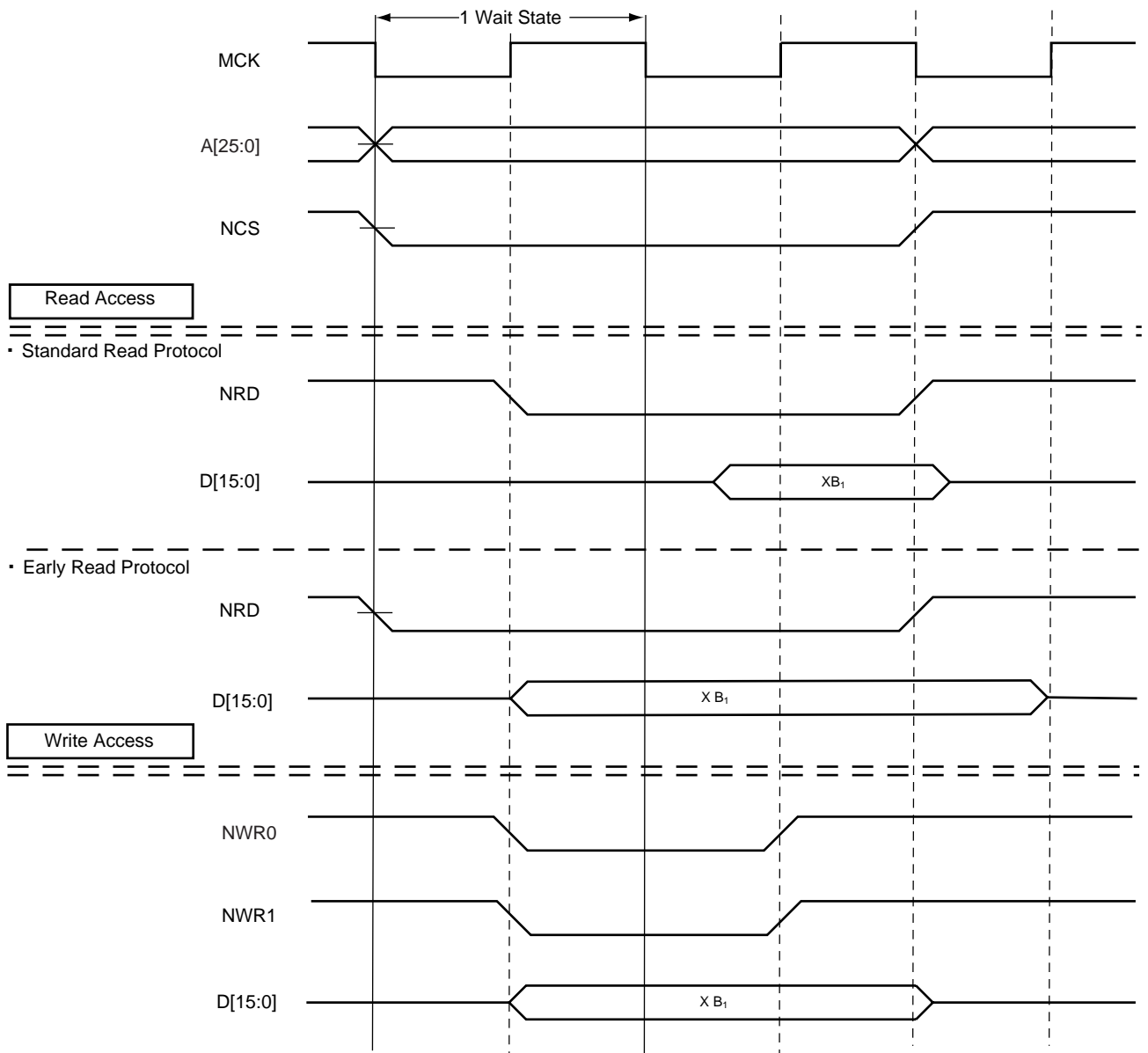
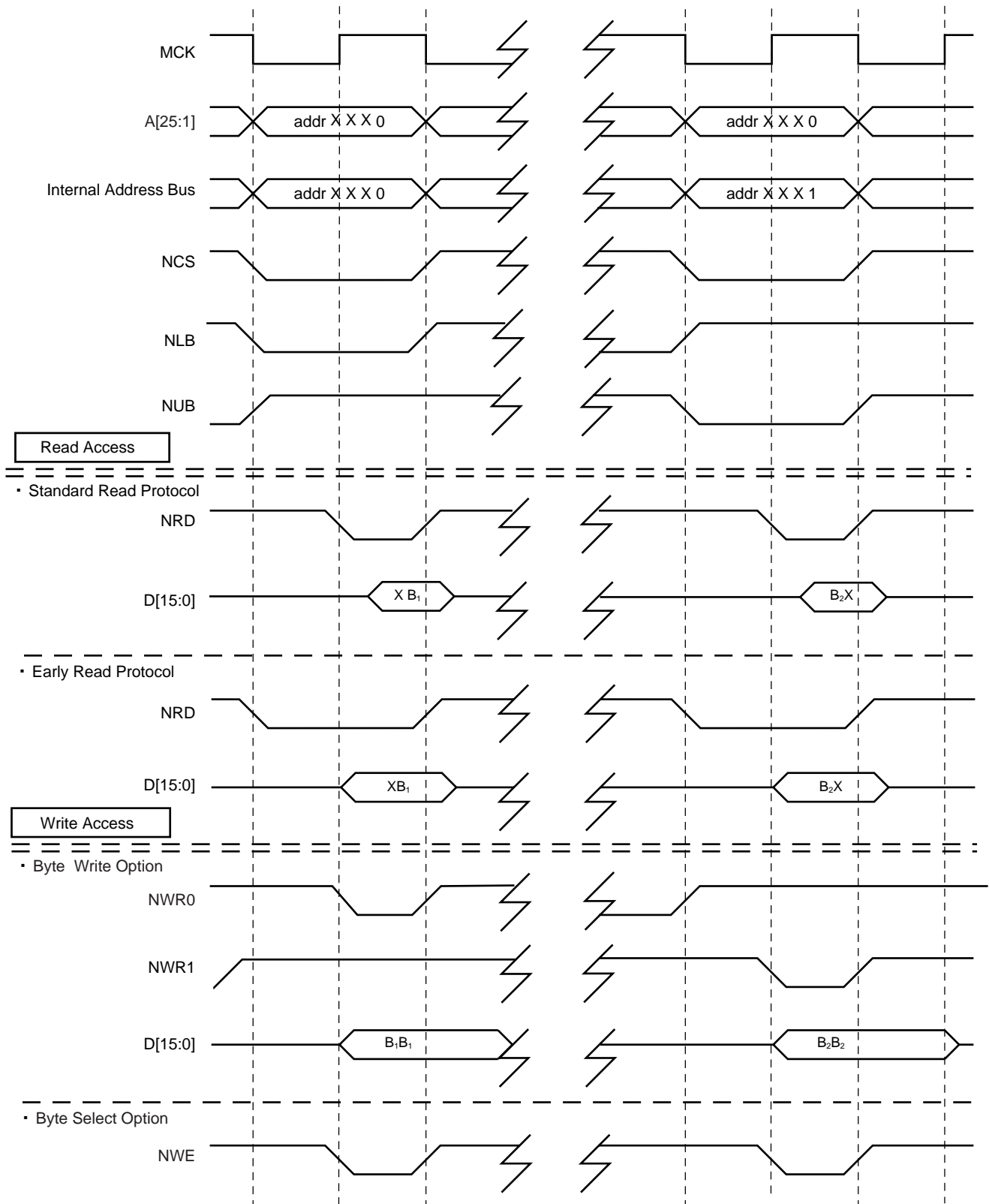


Figure 81. 1 Wait State, 8-bit Bus Width, Byte Transfer



**Figure 82.** 0 Wait State, 16-bit Bus Width, Byte Transfer





## Static Memory Controller (SMC) User Interface

The Static Memory Controller is programmed using the registers listed in Table 47. Eight Chip Select Registers (SMC\_CSR0 to SMC\_CSR7) are used to program the parameters for the individual external memories.

**Table 47.** Static Memory Controller Register Mapping

| Offset | Register                   | Name     | Access     | Reset State |
|--------|----------------------------|----------|------------|-------------|
| 0x00   | SMC Chip Select Register 0 | SMC_CSR0 | Read/Write | 0x00002000  |
| 0x04   | SMC Chip Select Register 1 | SMC_CSR1 | Read/Write | 0x00002000  |
| 0x08   | SMC Chip Select Register 2 | SMC_CSR2 | Read/Write | 0x00002000  |
| 0x0C   | SMC Chip Select Register 3 | SMC_CSR3 | Read/Write | 0x00002000  |
| 0x10   | SMC Chip Select Register 4 | SMC_CSR4 | Read/Write | 0x00002000  |
| 0x14   | SMC Chip Select Register 5 | SMC_CSR5 | Read/Write | 0x00002000  |
| 0x18   | SMC Chip Select Register 6 | SMC_CSR6 | Read/Write | 0x00002000  |
| 0x1C   | SMC Chip Select Register 7 | SMC_CSR7 | Read/Write | 0x00002000  |

## SMC Chip Select Registers

**Register Name:** SMC\_CSR0..SMC\_CSR7

**Access Type:** Read/write

**Reset Value:** See Table 47 on page 185

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

- **NWS: Number of Wait States**

This field defines the Read and Write signal pulse length from 1 cycle up to 128 cycles.

Note: When WSEN is 0, NWS will be read to 0 whichever the previous programmed value should be.

- **WSEN: Wait State Enable**

0: Wait states are disabled.

1: Wait states are enabled.

- **TDF: Data Float Time**

The external bus is marked occupied and cannot be used by another chip select during TDF cycles. Up to 15 cycles can be defined.

- **BAT: Byte Access Type**

This field is used only if DBW defines a 16- or 32-bit data bus.

0: Chip select line is connected to two 8-bit wide devices or four 8-bit wide devices.

1: Chip select line is connected to a 16-bit wide device.

- **DBW: Data Bus Width**

| DBW |   | Data Bus Width    |
|-----|---|-------------------|
| 0   | 0 | Reserved (32-bit) |
| 0   | 1 | 16-bit            |
| 1   | 0 | 8-bit             |
| 1   | 1 | Reserved          |

- **DRP: Data Read Protocol**

0: Standard Read Protocol is used.

1: Early Read Protocol is used.

• **ACSS: Address to Chip Select Setup**

| ACSS |   | Chip Select Waveform   |
|------|---|--|
| 0    | 0 | Standard, asserted at the beginning of the access and deasserted at the end. |
| 0    | 1 | One cycle less at the beginning and the end of the access.                   |
| 1    | 0 | Two cycles less at the beginning and the end of the access.                  |
| 1    | 1 | Three cycles less at the beginning and the end of the access.                |

• **RWSETUP: Read and Write Signal Setup Time**

See definition and description below.

• **RWHOLD: Read and Write Signal Hold Time**

See definition and description below

| RWSETUP |   |   | NRD Setup   | NWR Setup    |
|---------|---|---|---|--------------|
| 0       | 0 | 0 | ½ cycle <sup>(1)</sup> or 0 cycles <sup>(2)</sup> | ½ cycle      |
| 0       | 0 | 1 | 1 + ½ cycles                                      | 1 + ½ cycles |
| 0       | 1 | 0 | 2 + ½ cycles                                      | 2 + ½ cycles |
| 0       | 1 | 1 | 3 + ½ cycles                                      | 3 + ½ cycles |
| 1       | 0 | 0 | 4 + ½ cycles                                      | 4 + ½ cycles |
| 1       | 0 | 1 | 5 + ½ cycles                                      | 5 + ½ cycles |
| 1       | 1 | 0 | 6 + ½ cycles                                      | 6 + ½ cycles |
| 1       | 1 | 1 | 7 + ½ cycles                                      | 7 + ½ cycles |

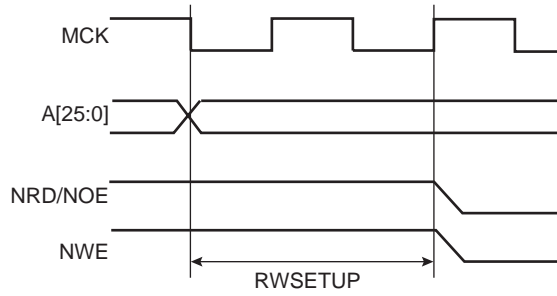
| RWHOLD |   |   | NRD Hold | NWR Hold |
|--------|---|---|----------|----------|
| 0      | 0 | 0 | 0        | ½ cycle  |
| 0      | 0 | 1 | 1 cycles | 1 cycle  |
| 0      | 1 | 0 | 2 cycles | 2 cycles |
| 0      | 1 | 1 | 3 cycles | 3 cycles |
| 1      | 0 | 0 | 4 cycles | 4 cycles |
| 1      | 0 | 1 | 5 cycles | 5 cycles |
| 1      | 1 | 0 | 6 cycles | 6 cycles |
| 1      | 1 | 1 | 7 cycles | 7 cycles |

- Notes: 1. In Standard Read Protocol.  
 2. In Early Read Protocol. (It is not possible to use the parameters RWSETUP or RWHOLD in this mode)

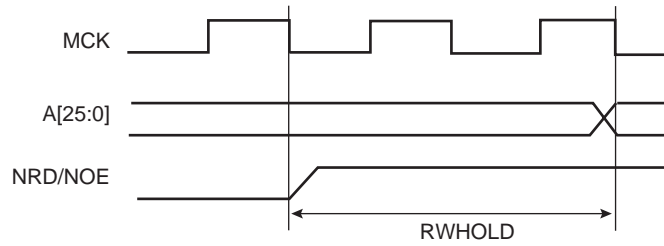
| NWS <sup>(2)</sup> | NRD Pulse Length | NWR Pulse Length |
|--------------------|------------------|------------------|
| 0                  | 1 + ½ cycles     | 1 cycles         |
| 1                  | 2 + ½ cycles     | 2 cycles         |
| Up to X = 127      | X + 1 + ½ cycles | X + 1 cycle      |

- Notes: 1. For a visual description, please refer to "Setup and Hold Cycles" on page 164 and the diagrams in Figure xx and Figure yy and Figure zz.  
 2. WSEN is considered to be 1.

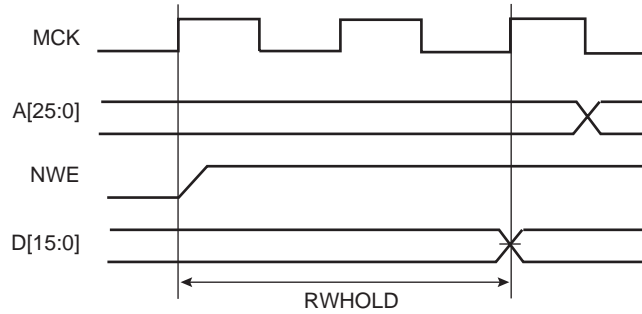
**Figure 83.** Read/write Setup



**Figure 84.** Read Hold



**Figure 85.** Write Hold



## SDRAM Controller (SDRAMC)

### Overview

The SDRAM Controller (SDRAMC) extends the memory capabilities of a chip by providing the interface to an external 16-bit or 32-bit SDRAM device. The page size supports ranges from 2048 to 8192 and the number of columns from 256 to 2048. It supports byte (8-bit), half-word (16-bit) and word (32-bit) accesses.

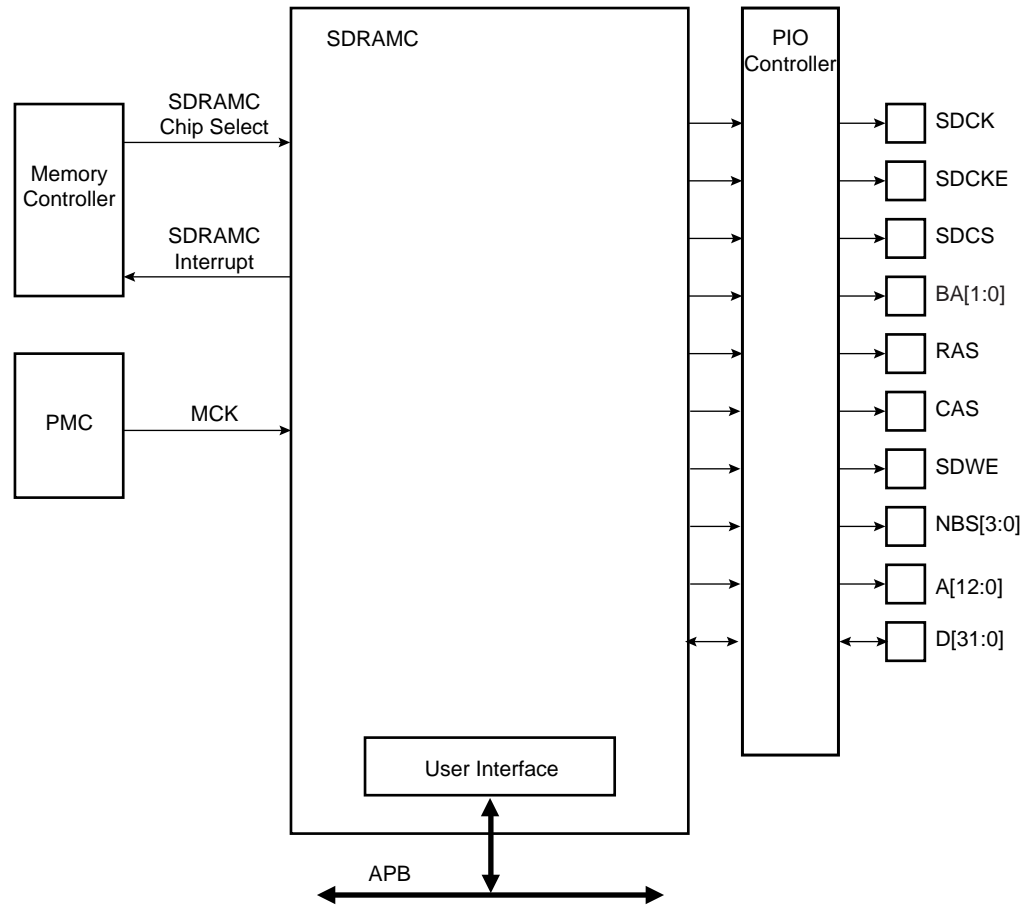
The SDRAM Controller supports a read or write burst length of one location. It does not support byte read/write bursts or half-word write bursts. It keeps track of the active row in each bank, thus maximizing SDRAM performance, e.g., the application may be placed in one bank and data in the other banks. So as to optimize performance, it is advisable to avoid accessing different rows in the same bank.

Features of the SDRAMC are:

- Numerous Configurations Supported
  - 2K, 4K, 8K Row Address Memory Parts
  - SDRAM with Two or Four Internal Banks
  - SDRAM with 16- or 32-bit Data Path
- Programming Facilities
  - Word, Half-word, Byte Access
  - Automatic Page Break When Memory Boundary Has Been Reached
  - Multibank Ping-pong Access
  - Timing Parameters Specified by Software
  - Automatic Refresh Operation, Refresh Rate is Programmable
- Energy-saving Capabilities
  - Self-refresh and Low-power Modes Supported
- Error Detection
  - Refresh Error Interrupt
- SDRAM Power-up Initialization by Software
- Latency is Set to Two Clocks (CAS Latency of 1, 3 Not Supported)
- Auto Precharge Command Not Used

## Block Diagram

Figure 86. SDRAM Controller Block Diagram



## I/O Lines Description

Table 48. I/O Line Description

| Name     | Description                  | Type   | Active Level |
|----------|------------------------------|--------|--------------|
| SDCK     | SDRAM Clock                  | Output |              |
| SDCKE    | SDRAM Clock Enable           | Output | High         |
| SDCS     | SDRAM Controller Chip Select | Output | Low          |
| BA[1:0]  | Bank Select Signals          | Output |              |
| RAS      | Row Signal                   | Output | Low          |
| CAS      | Column Signal                | Output | Low          |
| SDWE     | SDRAM Write Enable           | Output | Low          |
| NBS[3:0] | Data Mask Enable Signals     | Output | Low          |
| A[12:0]  | Address Bus                  | Output |              |
| D[31:0]  | Data Bus                     | I/O    |              |





## Software Interface

The SDRAM Controller's function is to make the SDRAM device access protocol transparent to the user. Table 49 to Table 54 illustrate the SDRAM device memory mapping therefore seen by the user in correlation with the device structure. Various configurations are illustrated.

### 32-bit Memory Data Bus Width

**Table 49.** SDRAM Configuration Mapping: 2K Rows, 256/512/1024/2048 Columns

| CPU Address Line |         |    |         |         |           |    |           |           |           |    |    |    |    |    |              |    |             |             |             |   |   |        |   |        |        |        |   |
|------------------|---------|----|---------|---------|-----------|----|-----------|-----------|-----------|----|----|----|----|----|--------------|----|-------------|-------------|-------------|---|---|--------|---|--------|--------|--------|---|
| 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 |
|                  |         |    |         |         | Bk[1:0]   |    |           |           | Row[10:0] |    |    |    |    |    |              |    |             |             | Column[7:0] |   |   |        |   |        |        | M[1:0] |   |
|                  |         |    |         | Bk[1:0] |           |    |           | Row[10:0] |           |    |    |    |    |    |              |    |             | Column[8:0] |             |   |   |        |   |        | M[1:0] |        |   |
|                  |         |    | Bk[1:0] |         |           |    | Row[10:0] |           |           |    |    |    |    |    |              |    | Column[9:0] |             |             |   |   |        |   | M[1:0] |        |        |   |
|                  | Bk[1:0] |    |         |         | Row[10:0] |    |           |           |           |    |    |    |    |    | Column[10:0] |    |             |             |             |   |   | M[1:0] |   |        |        |        |   |

**Table 50.** SDRAM Configuration Mapping: 4K Rows, 256/512/1024/2048 Columns

| CPU Address Line |         |         |         |         |           |           |           |           |    |    |    |    |    |    |              |             |             |             |   |   |   |        |        |        |        |   |   |
|------------------|---------|---------|---------|---------|-----------|-----------|-----------|-----------|----|----|----|----|----|----|--------------|-------------|-------------|-------------|---|---|---|--------|--------|--------|--------|---|---|
| 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 |
|                  |         |         |         | Bk[1:0] |           |           |           | Row[11:0] |    |    |    |    |    |    |              |             |             | Column[7:0] |   |   |   |        |        |        | M[1:0] |   |   |
|                  |         |         | Bk[1:0] |         |           |           | Row[11:0] |           |    |    |    |    |    |    |              |             | Column[8:0] |             |   |   |   |        |        | M[1:0] |        |   |   |
|                  |         | Bk[1:0] |         |         |           | Row[11:0] |           |           |    |    |    |    |    |    |              | Column[9:0] |             |             |   |   |   |        | M[1:0] |        |        |   |   |
|                  | Bk[1:0] |         |         |         | Row[11:0] |           |           |           |    |    |    |    |    |    | Column[10:0] |             |             |             |   |   |   | M[1:0] |        |        |        |   |   |

**Table 51.** SDRAM Configuration Mapping: 8K Rows, 256/512/1024/2048 Columns

| CPU Address Line |         |         |         |    |           |           |           |    |    |    |    |    |    |    |              |             |             |   |   |   |   |        |        |        |   |   |   |
|------------------|---------|---------|---------|----|-----------|-----------|-----------|----|----|----|----|----|----|----|--------------|-------------|-------------|---|---|---|---|--------|--------|--------|---|---|---|
| 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 |
|                  |         |         | Bk[1:0] |    |           |           | Row[12:0] |    |    |    |    |    |    |    |              |             | Column[7:0] |   |   |   |   |        |        | M[1:0] |   |   |   |
|                  |         | Bk[1:0] |         |    |           | Row[12:0] |           |    |    |    |    |    |    |    |              | Column[8:0] |             |   |   |   |   |        | M[1:0] |        |   |   |   |
|                  | Bk[1:0] |         |         |    | Row[12:0] |           |           |    |    |    |    |    |    |    | Column[9:0]  |             |             |   |   |   |   | M[1:0] |        |        |   |   |   |
|                  | Bk[1:0] |         |         |    | Row[12:0] |           |           |    |    |    |    |    |    |    | Column[10:0] |             |             |   |   |   |   | M[1:0] |        |        |   |   |   |

- Notes:
1. M[1:0] is the byte address inside a 32-bit word.
  2. Bk[1] = BA1, Bk[0] = BA0.



## 16-bit Memory Data Bus Width

**Table 52.** SDRAM Configuration Mapping: 2K Rows, 256/512/1024/2048 Columns

| CPU Address Line |    |    |         |         |         |         |           |           |           |           |    |    |    |    |    |    |              |             |             |             |   |   |   |    |    |    |    |
|------------------|----|----|---------|---------|---------|---------|-----------|-----------|-----------|-----------|----|----|----|----|----|----|--------------|-------------|-------------|-------------|---|---|---|----|----|----|----|
| 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  |
|                  |    |    |         |         |         | Bk[1:0] |           |           |           | Row[10:0] |    |    |    |    |    |    |              |             |             | Column[7:0] |   |   |   |    |    |    | M0 |
|                  |    |    |         |         | Bk[1:0] |         |           |           | Row[10:0] |           |    |    |    |    |    |    |              |             | Column[8:0] |             |   |   |   |    |    | M0 |    |
|                  |    |    |         | Bk[1:0] |         |         |           | Row[10:0] |           |           |    |    |    |    |    |    |              | Column[9:0] |             |             |   |   |   |    | M0 |    |    |
|                  |    |    | Bk[1:0] |         |         |         | Row[10:0] |           |           |           |    |    |    |    |    |    | Column[10:0] |             |             |             |   |   |   | M0 |    |    |    |

**Table 53.** SDRAM Configuration Mapping: 4K Rows, 256/512/1024/2048 Columns

| CPU Address Line |    |         |         |         |         |           |           |           |           |    |    |    |    |    |    |              |             |             |             |   |   |   |    |    |    |    |   |
|------------------|----|---------|---------|---------|---------|-----------|-----------|-----------|-----------|----|----|----|----|----|----|--------------|-------------|-------------|-------------|---|---|---|----|----|----|----|---|
| 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 |
|                  |    |         |         |         | Bk[1:0] |           |           |           | Row[11:0] |    |    |    |    |    |    |              |             |             | Column[7:0] |   |   |   |    |    |    | M0 |   |
|                  |    |         |         | Bk[1:0] |         |           |           | Row[11:0] |           |    |    |    |    |    |    |              |             | Column[8:0] |             |   |   |   |    |    | M0 |    |   |
|                  |    |         | Bk[1:0] |         |         |           | Row[11:0] |           |           |    |    |    |    |    |    |              | Column[9:0] |             |             |   |   |   |    | M0 |    |    |   |
|                  |    | Bk[1:0] |         |         |         | Row[11:0] |           |           |           |    |    |    |    |    |    | Column[10:0] |             |             |             |   |   |   | M0 |    |    |    |   |

**Table 54.** SDRAM Configuration Mapping: 8K Rows, 256/512/1024/2048 Columns

| CPU Address Line |         |         |         |         |           |           |           |           |    |    |    |    |    |    |              |             |             |             |   |   |   |    |    |    |    |   |   |
|------------------|---------|---------|---------|---------|-----------|-----------|-----------|-----------|----|----|----|----|----|----|--------------|-------------|-------------|-------------|---|---|---|----|----|----|----|---|---|
| 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 |
|                  |         |         |         | Bk[1:0] |           |           |           | Row[12:0] |    |    |    |    |    |    |              |             |             | Column[7:0] |   |   |   |    |    |    | M0 |   |   |
|                  |         |         | Bk[1:0] |         |           |           | Row[12:0] |           |    |    |    |    |    |    |              |             | Column[8:0] |             |   |   |   |    |    | M0 |    |   |   |
|                  |         | Bk[1:0] |         |         |           | Row[12:0] |           |           |    |    |    |    |    |    |              | Column[9:0] |             |             |   |   |   |    | M0 |    |    |   |   |
|                  | Bk[1:0] |         |         |         | Row[12:0] |           |           |           |    |    |    |    |    |    | Column[10:0] |             |             |             |   |   |   | M0 |    |    |    |   |   |

- Notes:
1. M0 is the byte address inside a 16-bit half-word.
  2. Bk[1] = BA1, Bk[0] = BA0.

## Product Dependencies

### SDRAM Devices Initialization

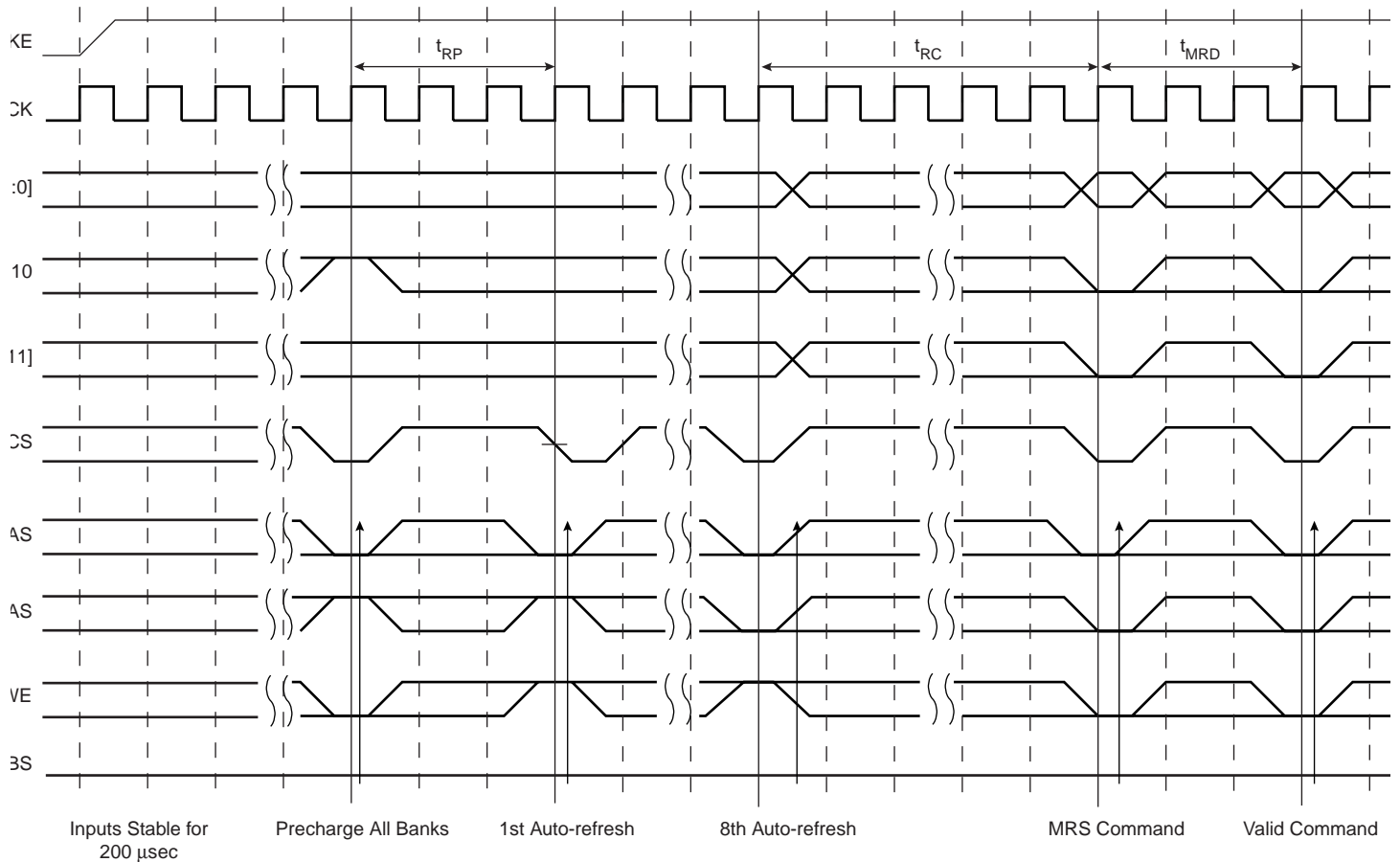
The initialization sequence is generated by software. The SDRAM devices are initialized by the following sequence:

1. A minimum pause of 200  $\mu$ s is provided to precede any signal toggle.
2. An All Banks Precharge command is issued to the SDRAM devices.
3. Eight auto-refresh (CBR) cycles are provided.
4. A mode register set (MRS) cycle is issued to program the parameters of the SDRAM devices, in particular CAS latency and burst length.
5. A Normal Mode command is provided, 3 clocks after  $t_{MRD}$  is met.
6. Write refresh rate into the count field in the SDRAMC Refresh Timer register. (Refresh rate = delay between refresh cycles).

After these six steps, the SDRAM devices are fully functional.

The commands (NOP, MRS, CBR, normal mode) are generated by programming the command field in the SDRAMC Mode register

**Figure 89.** SDRAM Devices Initialization Sequence



**I/O Lines**

The pins used for interfacing the SDRAM Controller may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the SDRAM Controller pins to their peripheral function. If I/O lines of the SDRAM Controller are not used by the application, they can be used for other purposes by the PIO Controller.

**Interrupt**

The SDRAM Controller interrupt (Refresh Error notification) is connected to the Memory Controller. This interrupt may be ORed with other System Peripheral interrupt lines and is finally provided as the System Interrupt Source (Source 1) to the AIC (Advanced Interrupt Controller).

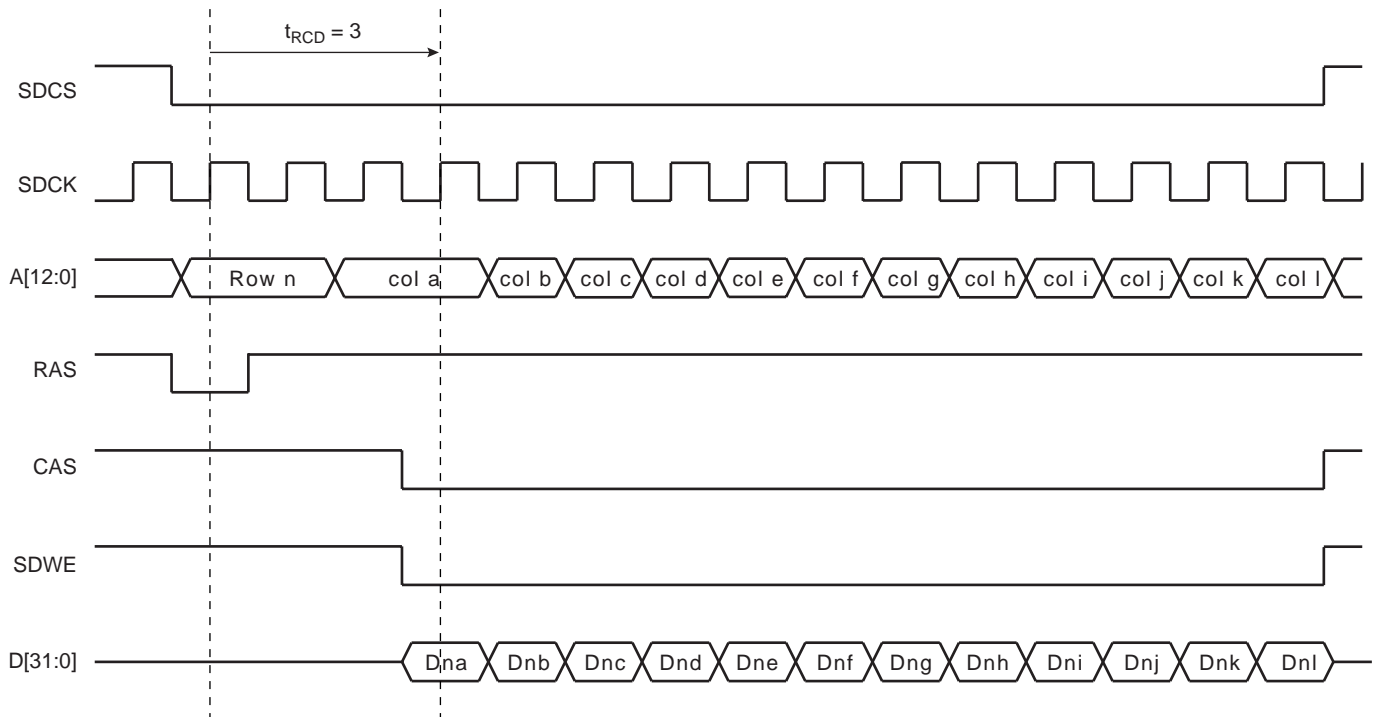
Using the SDRAM Controller interrupt requires the AIC to be programmed first.

**Functional Description**

**SDRAM Controller Write Cycle**

The SDRAM Controller allows burst access or single access. To initiate a burst access, the SDRAM Controller uses the transfer type signal provided by the master requesting the access. If the next access is a sequential write access, writing to the SDRAM device is carried out. If the next access is a write-sequential access, but the current access is to a boundary page, or if the next access is in another row, then the SDRAM Controller generates a precharge command, activates the new row and initiates a write command. To comply with SDRAM timing parameters, additional clock cycles are inserted between precharge/active ( $t_{RP}$ ) commands and active/write ( $t_{RCD}$ ) commands. For definition of these timing parameters, refer to the “SDRAMC Configuration Register” on page 204. This is described in Figure 90 below.

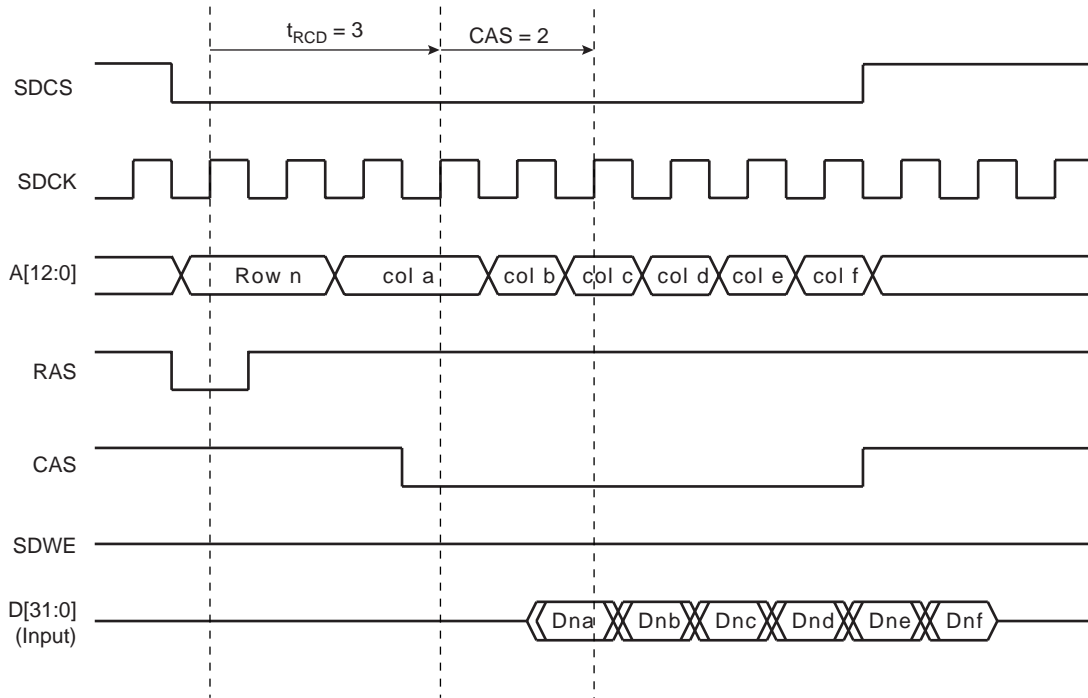
**Figure 90.** Write Burst, 32-bit SDRAM Access



## SDRAM Controller Read Cycle

The SDRAM Controller allows burst access or single access. To initiate a burst access, the SDRAM Controller uses the transfer type signal provided by the master requesting the access. If the next access is a sequential read access, reading to the SDRAM device is carried out. If the next access is a sequential read access, but the current access is to a boundary page, or if the next access is in another row, then the SDRAM Controller generates a precharge command, activates the new row and initiates a read command. To comply with SDRAM timing parameters, an additional clock cycle is inserted between the precharge/active ( $t_{RP}$ ) command and the active/read ( $t_{RCD}$ ) command. After a read command, additional wait states are generated to comply with cas latency. The SDRAM Controller supports a cas latency of two. For definition of these timing parameters, refer to “SDRAMC Configuration Register” on page 204. This is described in Figure 91 below.

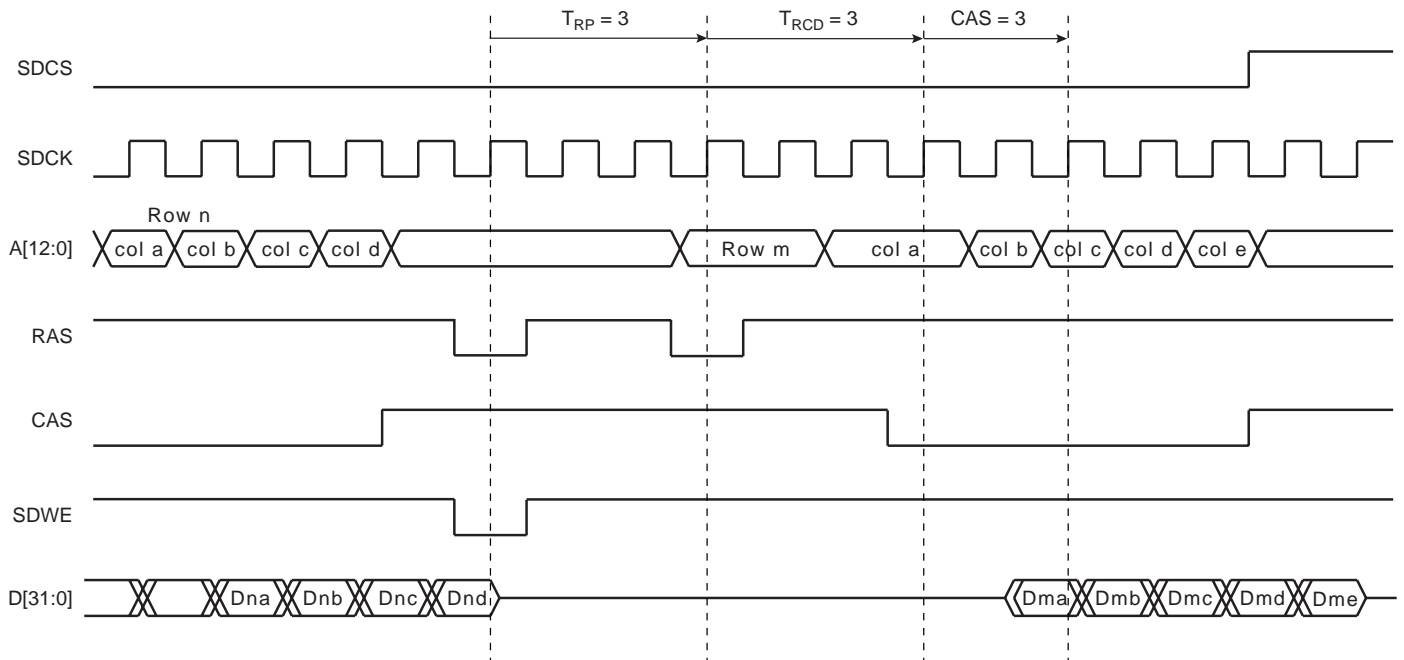
**Figure 91.** Read Burst, 32-bit SDRAM access



**Border Management**

When the memory row boundary has been reached, an automatic page break is inserted. In this case, the SDRAM controller generates a precharge command, activates the new row and initiates a read or write command. To comply with SDRAM timing parameters, an additional clock cycle is inserted between the precharge/active ( $t_{RP}$ ) command and the active/read ( $t_{RCD}$ ) command. This is described in Figure 92 below.

**Figure 92.** Read Burst with Boundary Row Access



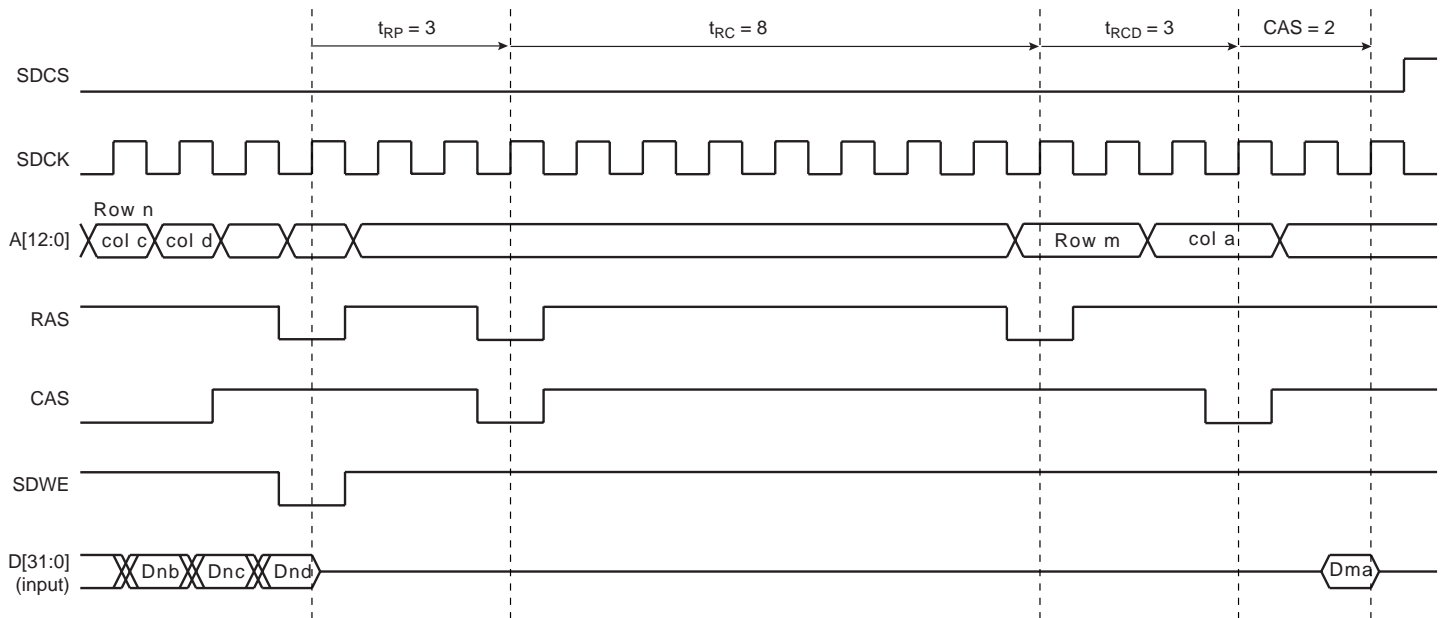
## SDRAM Controller Refresh Cycles

An auto-refresh command is used to refresh the SDRAM device. Refresh addresses are generated internally by the SDRAM device and incremented after each auto-refresh automatically. The SDRAM Controller generates these auto-refresh commands periodically. A timer is loaded with the value in the register SDRAMC\_TR that indicates the number of clock cycles between refresh cycles.

A refresh error interrupt is generated when the previous auto-refresh command did not perform. It will be acknowledged by reading the Interrupt Status Register (SDRAMC\_ISR).

When the SDRAM Controller initiates a refresh of the SDRAM device, internal memory accesses are not delayed. However, if the CPU tries to access the SDRAM, the slave will indicate that the device is busy and the ARM BWAIT signal will be asserted. See Figure 93 below.

**Figure 93.** Refresh Cycle Followed by a Read Access



## Power Management

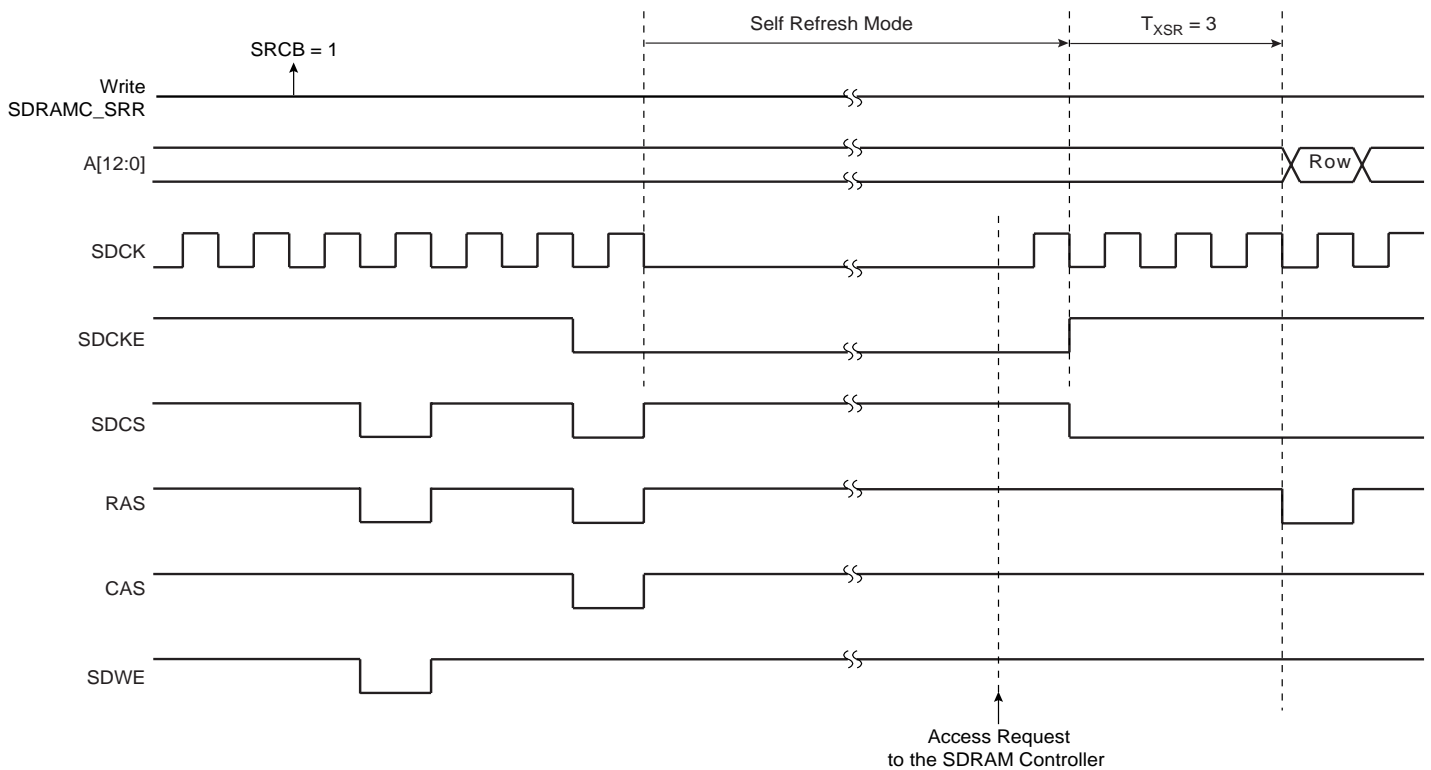
### Self-refresh Mode

Self-refresh mode is used in power-down mode, i.e., when no access to the SDRAM device is possible. In this case, power consumption is very low. The mode is activated by programming the self-refresh command bit (SRCB) in SDRAMC\_SRR. In self-refresh mode, the SDRAM device retains data without external clocking and provides its own internal clocking, thus performing its own auto-refresh cycles. All the inputs to the SDRAM device become “don’t care” except SDCKE, which remains low. As soon as the SDRAM device is selected, the SDRAM Controller provides a sequence of commands and exits self-refresh mode, so the self-refresh command bit is disabled.

To re-activate this mode, the self-refresh command bit must be re-programmed.

The SDRAM device must remain in self-refresh mode for a minimum period of  $t_{RAS}$  and may remain in self-refresh mode for an indefinite period. This is described in Figure 94 below.

Figure 94. Self-refresh Mode Behavior



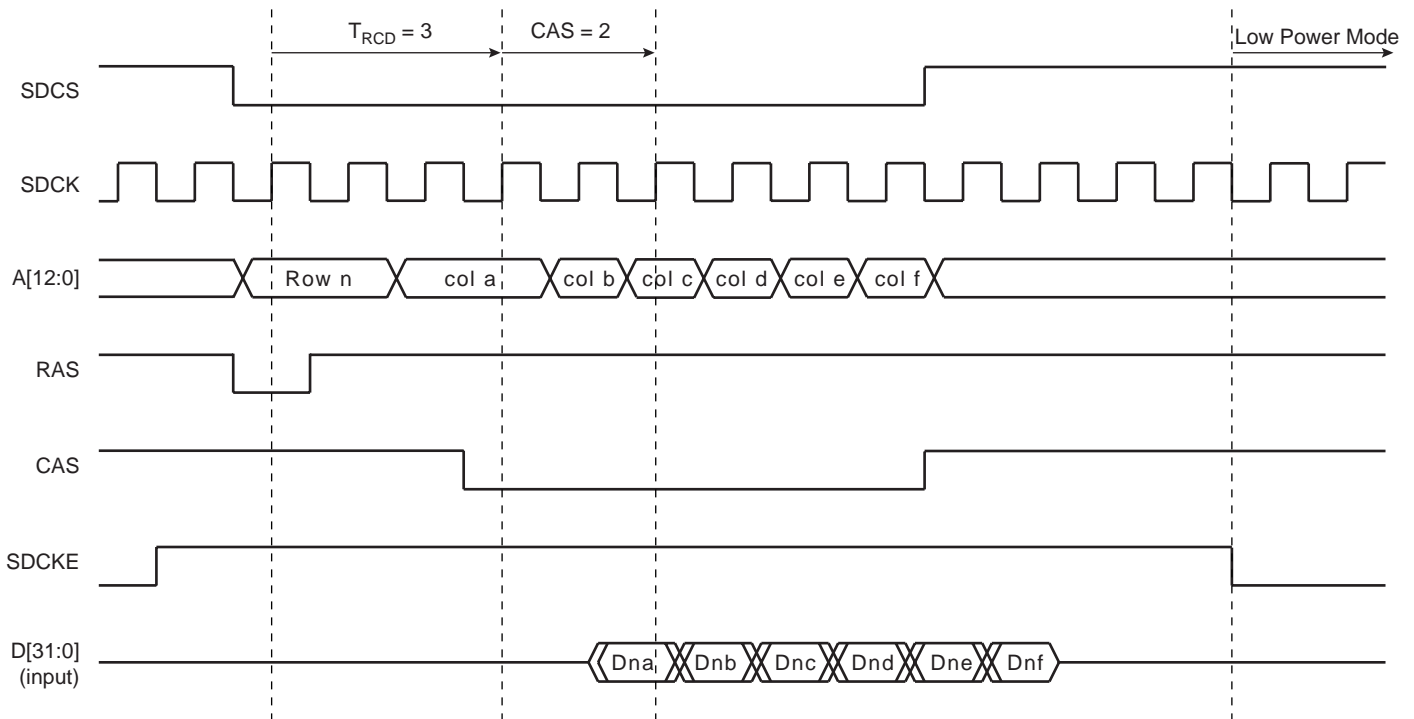
### Low-power Mode

Low-power mode is used in power-down mode, i.e., when no access to the SDRAM device is possible. In this mode, power consumption is greater than in self-refresh mode. This state is similar to normal mode (No low-power mode/No self-refresh mode), but the SDCKE pin is low and the input and output buffers are deactivated as soon as the SDRAM device is no longer accessible. In contrast to self-refresh mode, the SDRAM device cannot remain in low-power mode longer than the refresh period (64 ms for a whole device refresh operation). As no auto-refresh operations are performed in this mode, the SDRAM Controller carries out the refresh operation. In order to exit low-power mode, a NOP command is required. The exit procedure is faster than in self-refresh mode.

When self-refresh mode is enabled, it is recommended to avoid enabling low-power mode. When low-power mode is enabled, it is recommended to avoid enabling self-refresh mode.

This is described in Figure 95 below.

**Figure 95.** Low-power Mode Behavior





## SDRAM Controller (SDRAMC) User Interface

**Table 55.** SDRAM Controller Memory Map

| Offset | Register                          | Name       | Access     | Reset State |
|--------|-----------------------------------|------------|------------|-------------|
| 0x00   | SDRAMC Mode Register              | SDRAMC_MR  | Read/Write | 0x00000010  |
| 0x04   | SDRAMC Refresh Timer Register     | SDRAMC_TR  | Read/Write | 0x00000800  |
| 0x08   | SDRAMC Configuration Register     | SDRAMC_CR  | Read/Write | 0x2A99C140  |
| 0x0C   | SDRAMC Self Refresh Register      | SDRAMC_SRR | Write-only | –           |
| 0x10   | SDRAMC Low Power Register         | SDRAMC_LPR | Read/Write | 0x0         |
| 0x14   | SDRAMC Interrupt Enable Register  | SDRAMC_IER | Write-only | –           |
| 0x18   | SDRAMC Interrupt Disable Register | SDRAMC_IDR | Write-only | –           |
| 0x1C   | SDRAMC Interrupt Mask Register    | SDRAMC_IMR | Read-only  | 0x0         |
| 0x20   | SDRAMC Interrupt Status Register  | SDRAMC_ISR | Read-only  | 0x0         |



## SDRAMC Mode Register

Register Name: SDRAMC\_MR

Access Type: Read/Write

Reset Value: 0x00000010

|    |    |    |     |      |    |    |    |
|----|----|----|-----|------|----|----|----|
| 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  |
| –  | –  | –  | DBW | MODE |    |    |    |

### • MODE: SDRAMC Command Mode

This field defines the command issued by the SDRAM Controller when the SDRAM device is accessed.

| MODE |   |   |   | Description  |
|------|---|---|---|--|
| 0    | 0 | 0 | 0 | Normal mode. Any access to the SDRAM is decoded normally.  |
| 0    | 0 | 0 | 1 | The SDRAM Controller issues a NOP command when the SDRAM device is accessed regardless of the cycle.   |
| 0    | 0 | 1 | 0 | The SDRAM Controller issues an “All Banks Precharge” command when the SDRAM device is accessed regardless of the cycle.  |
| 0    | 0 | 1 | 1 | The SDRAM Controller issues a “Load Mode Register” command when the SDRAM device is accessed regardless of the cycle. The address offset with respect to the SDRAM device base address is used to program the Mode Register. For instance, when this mode is activated, an access to the “SDRAM_Base + offset” address generates a “Load Mode Register” command with the value “offset” written to the SDRAM device Mode Register. |
| 0    | 1 | 0 | 0 | The SDRAM Controller issues a “Refresh” Command when the SDRAM device is accessed regardless of the cycle. Previously, an “All Banks Precharge” command must be issued.  |

### • DBW: Data Bus Width

0: Data bus width is 32 bits.

1: Data bus width is 16 bits.

**SDRAMC Refresh Timer Register**

**Register Name:** SDRAMC\_TR

**Access Type:** Read/Write

**Reset Value:** 0x00000800

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

• **COUNT: SDRAMC Refresh Timer Count**

This 12-bit field is loaded into a timer that generates the refresh pulse. Each time the refresh pulse is generated, a refresh burst is initiated. The value to be loaded depends on the SDRAMC clock frequency (MCK: Master Clock), the refresh rate of the SDRAM device and the refresh burst length where 15.6 μs per row is a typical value for a burst of length one.

To refresh the SDRAM device even if the reset value is not equal to 0, this 12-bit field must be written. If this condition is not satisfied, no refresh command is issued and no refresh of the SDRAM device is carried out.



## SDRAMC Configuration Register

Register Name: SDRAMC\_CR

Access Type: Read/Write

Reset Value: 0x2A99C140

|      |    |      |    |    |    |      |    |
|------|----|------|----|----|----|------|----|
| 31   | 30 | 29   | 28 | 27 | 26 | 25   | 24 |
| -    |    | TXSR |    |    |    | TRAS |    |
| 23   | 22 | 21   | 20 | 19 | 18 | 17   | 16 |
| TRAS |    | TRCD |    |    |    | TRP  |    |
| 15   | 14 | 13   | 12 | 11 | 10 | 9    | 8  |
| TRP  |    | TRC  |    |    |    | TWR  |    |
| 7    | 6  | 5    | 4  | 3  | 2  | 1    | 0  |
| TWR  |    | CAS  |    | NB | NR |      | NC |

### • NC: Number of Column Bits

Reset value is 8 column bits.

| NC |   | Column Bits |
|----|---|-------------|
| 0  | 0 | 8           |
| 0  | 1 | 9           |
| 1  | 0 | 10          |
| 1  | 1 | 11          |

### • NR: Number of Row Bits

Reset value is 11 row bits.

| NR |   | Row Bits |
|----|---|----------|
| 0  | 0 | 11       |
| 0  | 1 | 12       |
| 1  | 0 | 13       |
| 1  | 1 | Reserved |

### • NB: Number of Banks

Reset value is two banks.

| NB | Number of Banks |
|----|-----------------|
| 0  | 2               |
| 1  | 4               |

- **CAS: CAS Latency**

Reset value is two cycles.

In the SDRAMC, only a CAS latency of two cycles is managed. In any case, another value must be programmed.

| CAS |   | CAS Latency (Cycles) |
|-----|---|----------------------|
| 0   | 0 | Reserved             |
| 0   | 1 | Reserved             |
| 1   | 0 | 2                    |
| 1   | 1 | Reserved             |

- **TWR: Write Recovery Delay**

Reset value is two cycles.

This field defines the Write Recovery Time in number of cycles. Number of cycles is between 2 and 15.

If TWR is less than or equal to 2, two clock periods are inserted by default.

- **TRC: Row Cycle Delay**

Reset value is eight cycles.

This field defines the delay between a Refresh and an Activate Command in number of cycles. Number of cycles is between 2 and 15.

If TRC is less than or equal to 2, two clock periods are inserted by default.

- **TRP: Row Precharge Delay**

Reset value is three cycles.

This field defines the delay between a Precharge Command and another Command in number of cycles. Number of cycles is between 2 and 15.

If TRP is less than or equal to 2, two clock periods are inserted by default.

- **TRCD: Row to Column Delay**

Reset value is three cycles.

This field defines the delay between an Activate Command and a Read/Write Command in number of cycles. Number of cycles is between 2 and 15.

If TRCD is less than or equal to 2, two clock periods are inserted by default.

- **TRAS: Active to Precharge Delay**

Reset value is five cycles.

This field defines the delay between an Activate Command and a Precharge Command in number of cycles. Number of cycles is between 2 and 15.

If TRAS is less than or equal to 2, two clock periods are inserted by default.

- **TXSR: Exit Self Refresh to Active Delay**

Reset value is five cycles.

This field defines the delay between SCKE set high and an Activate Command in number of cycles. Number of cycles is between 1/2 and 15.5.

If TXSR is equal to 0, 1/2 clock period is inserted by default.

## SDRAMC Self-refresh Register

**Register Name:** SDRAMC\_SRR

**Access Type:** Write-only

|    |    |    |    |    |    |    |      |
|----|----|----|----|----|----|----|------|
| 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    |
| –  | –  | –  | –  | –  | –  | –  | SRCB |

- **SRCB: Self-refresh Command Bit**

0: No effect.

1: The SDRAM Controller issues a self-refresh command to the SDRAM device, the SDCK clock is inactivated and the SDCKE signal is set low. The SDRAM device leaves self-refresh mode when accessed again.

## SDRAMC Low-power Register

**Register Name:** SDRAMC\_LPR

**Access Type:** Read/Write

**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    |
| –  | –  | –  | –  | –  | –  | –  | LPCB |

- **LPCB: Low-power Command Bit**

0: The SDRAM Controller low-power feature is inhibited: no low-power command is issued to the SDRAM device.

1: The SDRAM Controller issues a low-power command to the SDRAM device after each burst access, the SDCKE signal is set low. The SDRAM device will leave low-power mode when accessed and enter it after the access.

### SDRAMC Interrupt Enable Register

Register Name: SDRAMC\_IER

Access Type: Write-only

|    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|-----|
| 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   |
| –  | –  | –  | –  | –  | –  | –  | RES |

- **RES: Refresh Error Status**

0: No effect.

1: Enables the refresh error interrupt.

### SDRAMC Interrupt Disable Register

Register Name: SDRAMC\_IDR

Access Type: Write-only

|    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|-----|
| 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   |
| –  | –  | –  | –  | –  | –  | –  | RES |

- **RES: Refresh Error Status**

0: No effect.

1: Disables the refresh error interrupt.

## SDRAMC Interrupt Mask Register

Register Name: SDRAMC\_IMR

Access Type: Read-only

|    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|-----|
| 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   |
| –  | –  | –  | –  | –  | –  | –  | RES |

- **RES: Refresh Error Status**

0: The refresh error interrupt is disabled.

1: The refresh error interrupt is enabled.

## SDRAMC Interrupt Status Register

Register Name: SDRAMC\_ISR

Access Type: Read-only

|    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|-----|
| 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   |
| –  | –  | –  | –  | –  | –  | –  | RES |

- **RES: Refresh Error Status**

0: No refresh error has been detected since the register was last read.

1: A refresh error has been detected since the register was last read.



## Burst Flash Controller (BFC)

### Overview

The Burst Flash Controller (BFC) provides an interface for external 16-bit Burst Flash devices and handles an address space of 256M bytes. It supports byte, half-word and word aligned accesses and can access up to 32M bytes of Burst Flash devices. The BFC also supports data bus and address bus multiplexing. The Burst Flash interface supports only continuous burst reads. Programmable burst lengths of four or eight words are not possible. The BFC never generates an abort signal, regardless of the requested address within the 256M bytes of address space.

The BFC can operate with two burst read protocols depending on whether or not the address increment of the Burst Flash device is signal controlled. The Burst Flash Controller Mode Register (BFC\_MR) located in the BFC user interface is used in programming Asynchronous or Burst Operating Modes. In Burst Mode, the read protocol, Clock Controlled Address Advance, automatically increments the address at each clock cycle. Whereas in Signal Controlled Address Advance protocol the address is incremented only when the Burst Address Advance signal is active. When Address and Data Bus Multiplexing Mode is chosen, the sixteen lowest address bits are multiplexed with the data bus.

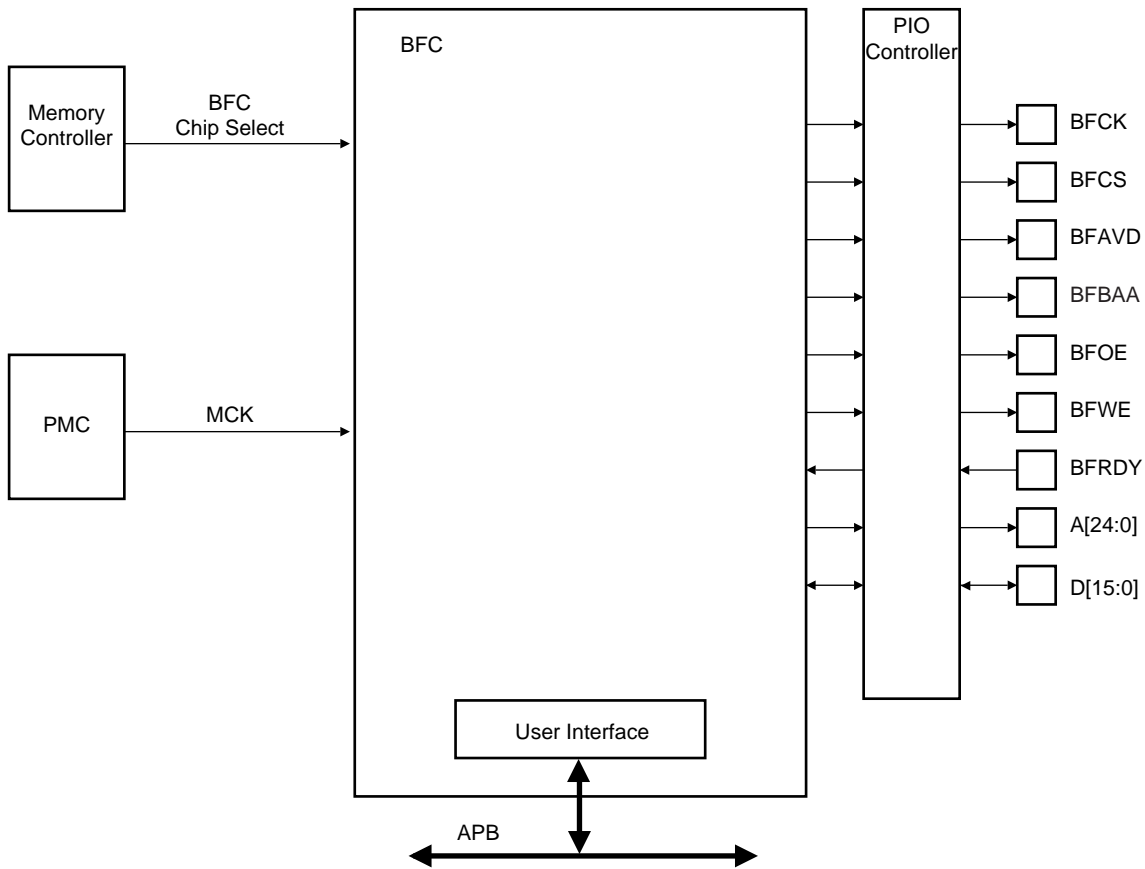
The BFC clock speed is programmable to be either master clock or master clock divided by 2 or 4. Page size handling (16 bytes to 1024 bytes) is required by some Burst Flash devices unable to handle continuous burst read. The number of latency cycles after address valid goes up to sixteen cycles. The number of latency cycles after output enable runs between one and three cycles. The Burst Flash Controller can also be programmed to suspend and maintain the current burst. This attribute gives other devices the possibility to share the BFC busses without any loss of efficiency. In Burst Mode, the BFC can restart a sequential access without any additional latency.

Features of the Burst Flash Controller are:

- Multiple Access Modes Supported
  - Asynchronous or Burst Mode Byte, Half-word or Word Read Accesses
  - Asynchronous Mode Half-word Write Accesses
- Adaptability to Different Device Speed Grades
  - Programmable Burst Flash Clock Rate
  - Programmable Data Access Time
  - Programmable Latency after Output Enable
- Adaptability to Different Device Access Protocols and Bus Interfaces
  - Two Burst Read Protocols: Clock Control Address Advance or Signal Controlled Address Advance
  - Multiplexed or Separate Address and Data Busses
  - Continuous Burst and Page Mode Accesses Supported

## Block Diagram

Figure 96. Burst Flash Controller Block Diagram



## I/O Lines Description

Table 56. I/O Lines Description

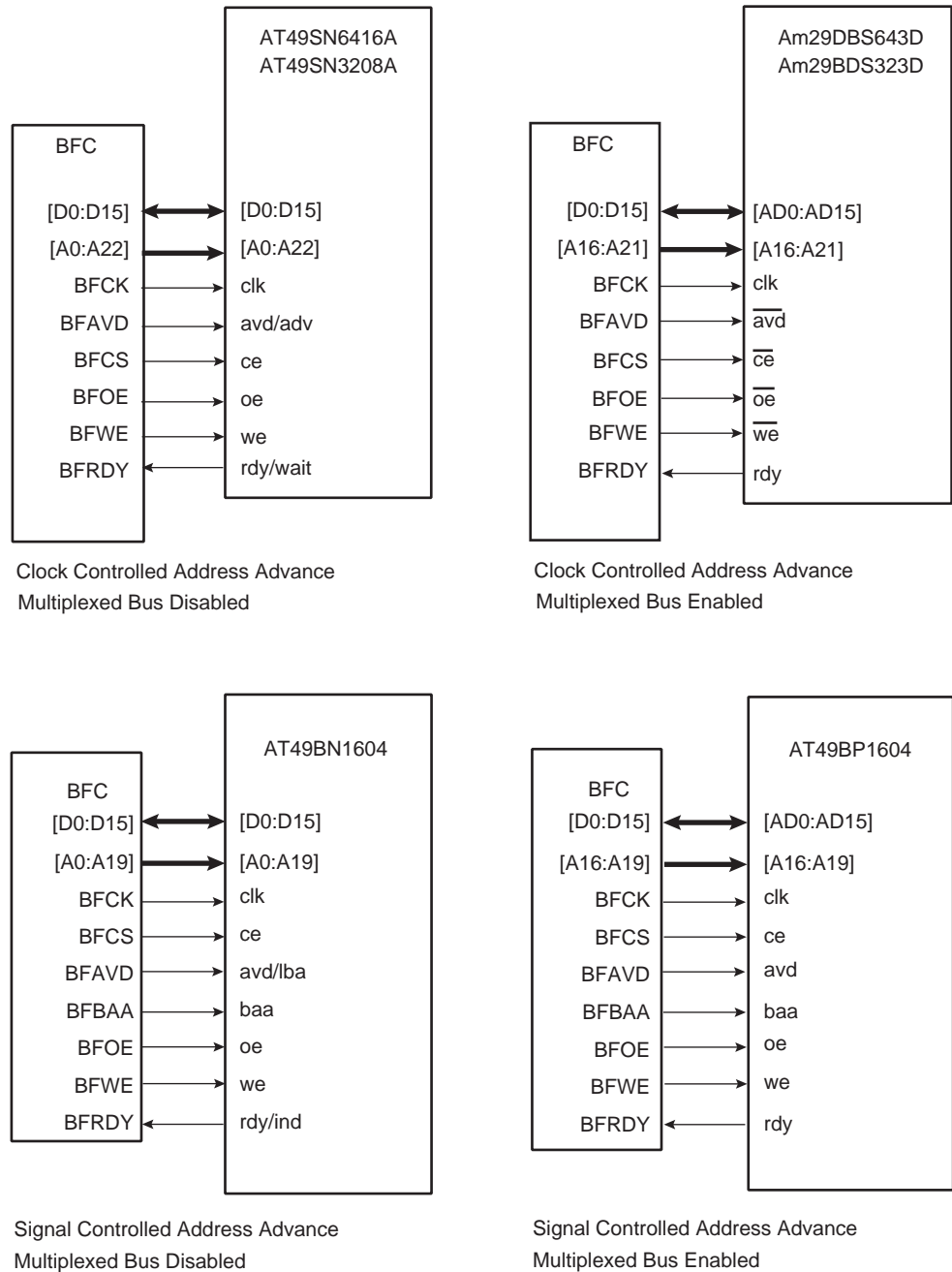
| Name    | Description                 | Type   | Active Level |
|---------|-----------------------------|--------|--------------|
| BFCK    | Burst Flash Clock           | Output |              |
| BFCS    | Burst Flash Chip Select     | Output | Low          |
| BFAVD   | Burst Flash Address Valid   | Output | Low          |
| BFBA    | Burst Flash Address Advance | Output | Low          |
| BFOE    | Burst Flash Output Enable   | Output | Low          |
| BFWE    | Burst Flash Write Enable    | Output | Low          |
| BFRDY   | Burst Flash Ready           | Input  | High         |
| A[24:0] | Address Bus                 | Output |              |
| D[15:0] | Data Bus                    | I/O    |              |

## Application Example

### Burst Flash Interface

The Burst Flash Interface provides control, address and data signals to the Burst Flash Memory. These signals are detailed in the “Functional Description” on page 212 which describes the BFC functionality and operating modes. Figure 97 below presents an illustration of the possible connections of the BFC to some popular Burst Flash Memories.

**Figure 97.** Burst Flash Controller Connection Example



## Product Dependencies

### Supported Burst Flash Devices

The Burst Flash Controller is designed to preferentially support the following ATMEL Burst Flash devices:

- AT49SN6416A and AT49SN6416AT (64 Mbits x 16)
- AT49SN3208A and AT49SN3208AT (32 Mbits x 16)
- AT49BN3208 and AT49BN3208T (32 Mbits x 16)

### I/O Lines

The pins used for interfacing the Burst Flash Controller may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the Burst Flash Controller pins to their peripheral function. If I/O lines of the Burst Flash Controller are not used by the application, they can be used for other purposes by the PIO Controller.

### Functional Description

The Burst Flash Controller drives the following signals:

- Address Valid (BFAVD), to latch the addresses
- Clock (BFCK), to supply the burst clock
- Burst Advance Address (BFBA), to control the Burst Flash memory address advance when programmed to operate in signal controlled burst advance
- Write Enable (BFWE), to write to the Burst Flash device
- Output Enable (BFOE), to enable the external device data drive on the data bus

When enabled, the BFC also drives the address bus, the data bus and the Chip Select (BFCS) line. The Ready Signal (BFRDY) is taken as an input and used as an indicator for the next data availability.

### Burst Flash Controller Reset State

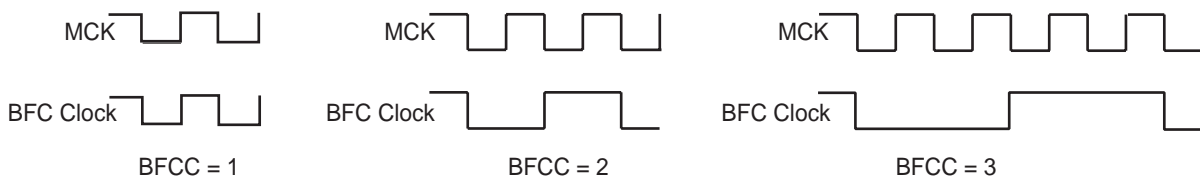
After reset, the BFC is disabled and, therefore, must be enabled by programming the field BFCOM. See “Burst Flash Controller Mode Register” on page 221. At this time, the Burst Flash Controller operates in Asynchronous Mode. The Burst Flash memory can be programmed by writing and reading in Asynchronous Mode.

### Burst Flash Controller Clock Selection

The BFC clock rate is programmable to be either Master Clock, Master Clock divided by 2 or Master Clock divided by 4. The clock selection is necessary in Burst Mode as well as in Asynchronous Mode. The latency fields in the mode register and all burst Flash control signal waveforms are related to the Burst Flash Clock (BFCK) period.

The BFC clock rate is selected by the BFCC field. See “Burst Flash Controller Mode Register” on page 221.

**Figure 98.** Burst Flash Clock Rates



## Burst Flash Controller Asynchronous Mode

In Asynchronous Mode, the Burst Flash Controller clock is off. The BFCK signal is driven low.

The BFC performs read access to bytes (8-bits), half-words (16-bits), and words (32-bits). In the last case, the BFC autonomously transforms the word read request into two separate half-word reads. This is fully transparent to the user.

The BFC performs only half-word write requests. Write requests for bytes or words are ignored by the BFC.

For any access in the address space, the address is driven on the address bus while a pulse is driven on the BFAVD signal (see Figure 99 on page 214, and Figure 100 on page 215). The Burst Flash address is also driven on the data bus if the multiplexed data and address bus options are enabled. (Figure 99 on page 214).

- For write access, the signal BFWE is asserted in the following BFCK clock cycle.
- For read access, the signal BFOE is asserted one cycle later. This additional cycle in read accesses has been inserted to switch the I/O pad direction so as to avoid conflict on the Burst Flash data bus when address and data busses are multiplexed.

The Address Valid Latency (AVL) determines the length of the pulses as a number of Master Clock cycles. The AVL field (See “Burst Flash Controller Mode Register” on page 221.) is coded as the Address Valid Latency minus 1. Waveforms in Figure 99 on page 214 and Figure 100 on page 215 show the AVL field definition in read and write accesses.

- In read access, the access finishes with the rising edge of BFOE.
- In write access, data and address lines are released one half cycle after the rising edge of BFWE.

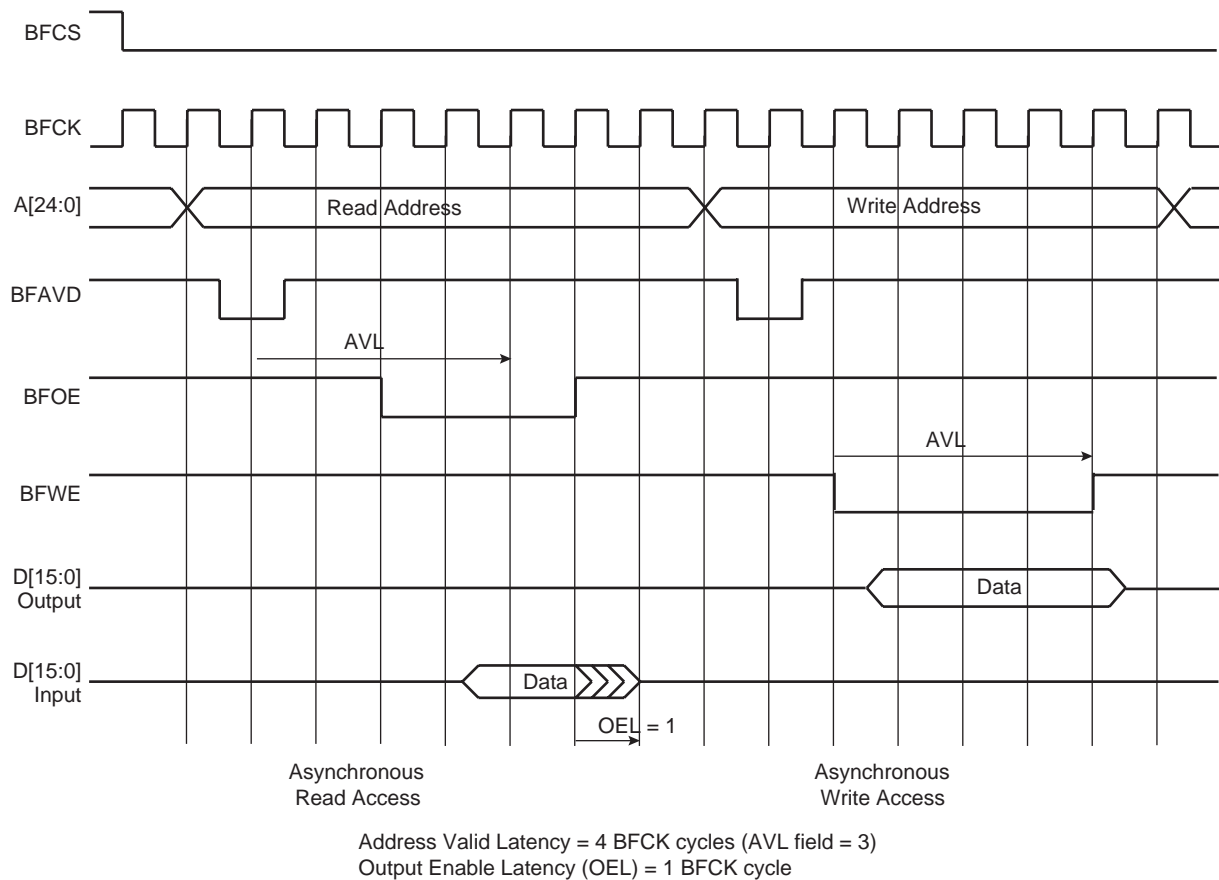
After a read access to the Burst Flash, it takes Output Enable Latency (OEL) cycles for the Burst Flash device to release the data bus. The OEL field (See “Burst Flash Controller Mode Register” on page 221.) gives the OEL expressed in BFCK Clock cycles. This prevents other memory controllers from using the Data Bus until it is released by the Burst Flash device.

In Figure 99 on page 214 (multiplexed address and data busses), one idle cycle (OEL = 1) is inserted between the read and write accesses. The Burst Flash device must release the data bus before the BFC can drive the address. As shown in Figure 100 on page 215, where busses are not multiplexed, the write access can start as soon as the read access ends. In the same way, the OEL has no impact when a read follows a write access.

Waveforms in Figure 99 on page 214 below and Figure 100 on page 215 are related to the Burst Flash Controller Clock even though the BFCK pin is driven low in Asynchronous Mode. The BFCC field (See “Burst Flash Controller Mode Register” on page 221.) is used as a measure of the burst Flash speed and must also be programmed in Asynchronous Mode.



**Figure 100.** Asynchronous Read and Write Accesses with Non-multiplexed Address and Data



**Burst Flash Controller Synchronous Mode**

Writing the Burst Flash Controller Operating Mode field (BFCOM) to 2 (see “Burst Flash Controller Mode Register” on page 221) puts the BFC in Burst Mode. The BFC Clock is driven on the BFCK pin. Only read accesses are treated and write accesses are ignored. The BFC supports read access of bytes, half-words or words.

**Burst Read Protocols**

The BFC supports two burst read protocols:

- Clock Controlled Address Advance, the internal address of the burst Flash is automatically incremented at each BFCK cycle.
- Signal Controlled Address Advance, the internal address of the burst Flash is incremented only when the BFBA signal is active.

**Read Access in Burst Mode**

When a read access is requested in Burst Mode, the requested address is registered in the BFC. For subsequent read accesses, the address is compared to the previous one. Then the two following cases are considered:

1. In case of a non-sequential access, the current burst is broken and the BFC launches a new burst by performing an address latch cycle. The address is presented on the address bus in any case and on the data bus if the multiplexed bus option is enabled.

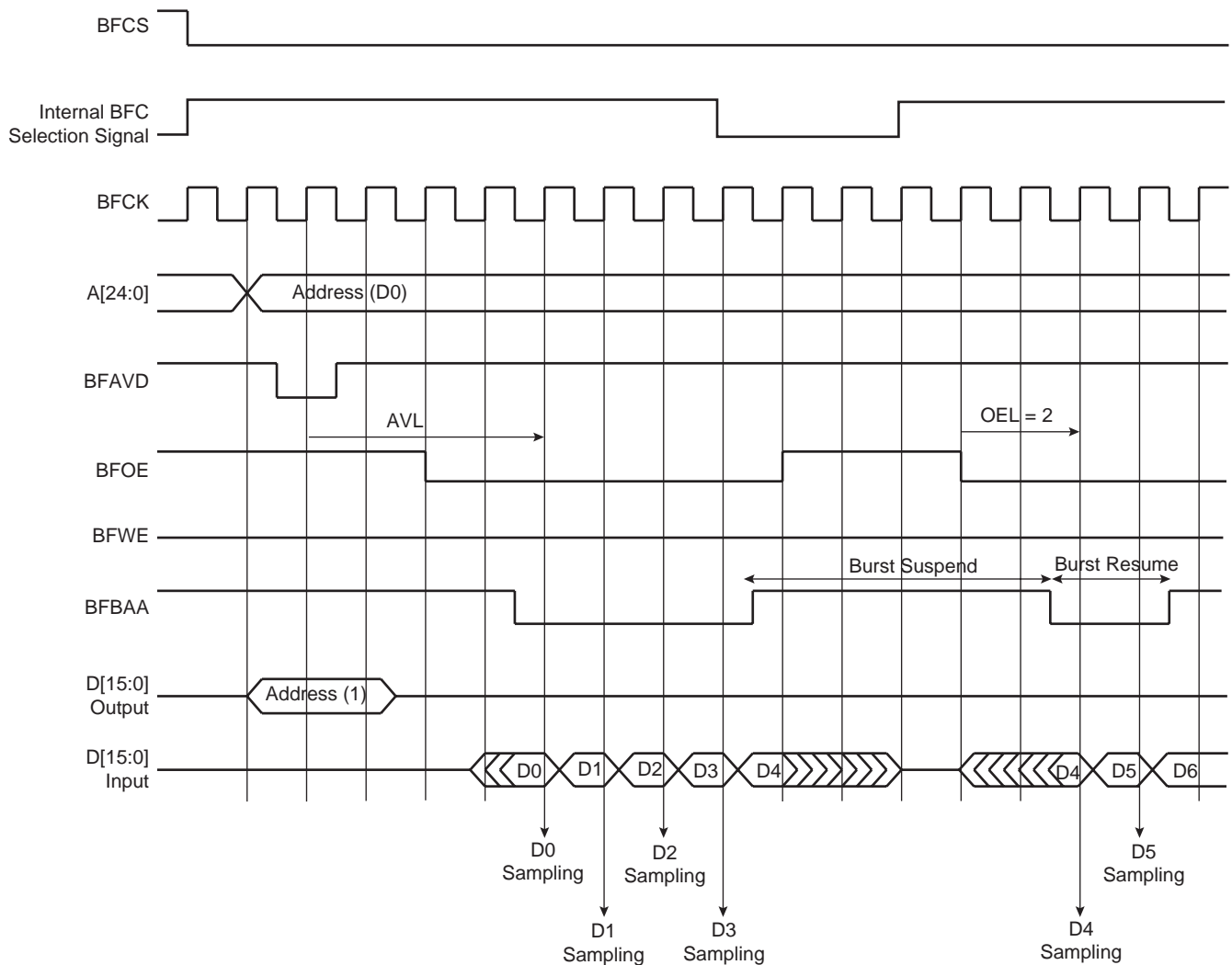
This new address is registered in the BFC and is then used as reference for further accesses.

2. In case of sequential access, and provided that the BFOEH mode is selected in the mode register (See “Burst Flash Controller Mode Register” on page 221.), the internal burst address is incremented:
  - Through the BFBA pin, if the Signal Controlled Address Advance is enabled.
  - By enabling the clock during one clock cycle in Clock Controlled Address Advance Mode.

These protocols are illustrated in Figure 101 below and Figure 102 on page 217. The Address Valid Latency (AVL+1, see “Burst Flash Controller Mode Register” on page 221) gives the number of cycles from the first rising clock edge when BFAVD is asserted to the rising edge that causes the read of data D1.

Note: This rising edge is also used to latch D0 in the BFC.

**Figure 101.** Burst Suspend and Resume with Signal Control Address Advance

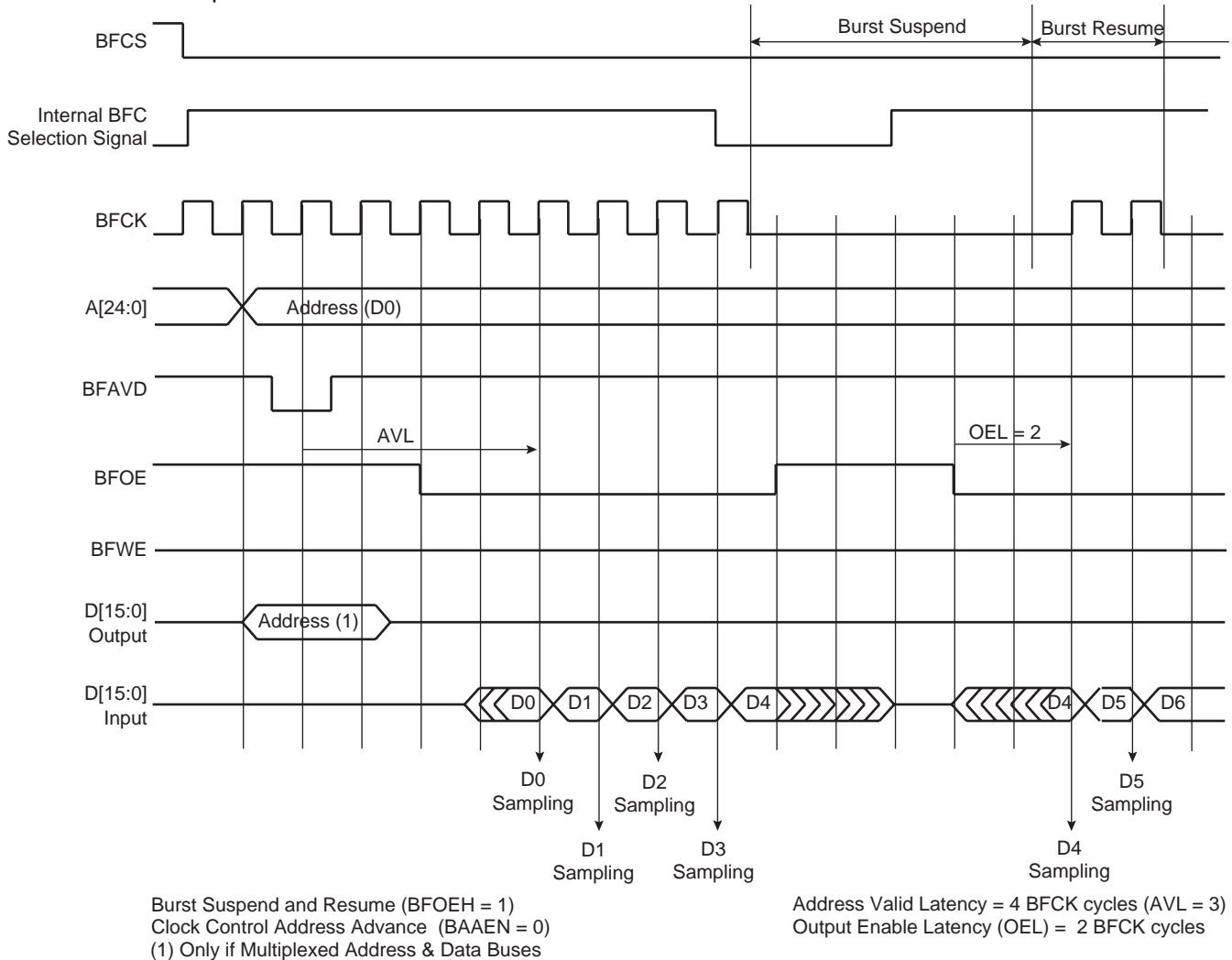


Burst Suspend and Resume (BFOEH = 1)  
 Signal Control Address Advance (BAAEN = 1)  
 (1) Only if Multiplexed Address & Data Buses

Address Valid Latency = 4 BFCK cycles (AVL field = 3)  
 Output Enable Latency (OEL) = 2 BFCK cycles



**Figure 102.** Burst Suspend and Resume with Clock Control Address Advance

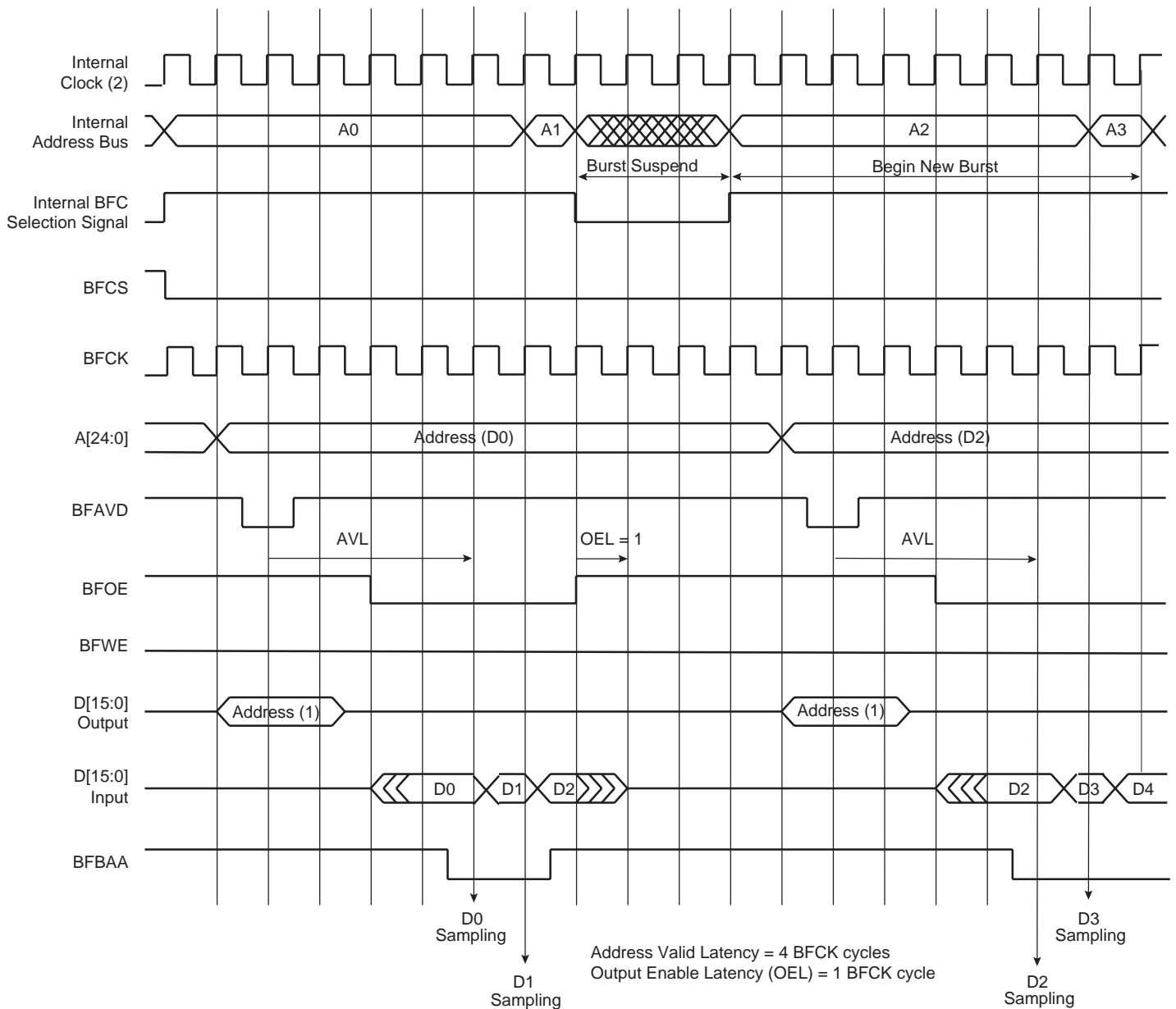


## Burst Suspension for Transfer Enabling

The BFC can suspend a burst to enable other internal transfers, or other memory controllers to use the memory address and data busses if they are shared. Two modes are provided on the BFOEH bit (Burst Flash Output Enable Handling, see “Burst Flash Controller Mode Register” on page 221):

- BFOEH = 1: the BFC suspends the burst when it is no longer selected and the BFOE pin is deasserted. When a new sequential access on the Burst Flash device is requested, the burst is resumed and the BFOE pin is asserted again. The data is available on the data bus after OEL cycles. This mode provides a minimal access latency. (Refer to Figure 101 on page 216 and Figure 102 above).
- BFOEH = 0: the BFC suspends the burst when it is no longer selected and the BFOE pin is deasserted. When a new access to the Burst Flash device is requested, either sequential or not, a new burst is initialized and the next data is available as defined by the AVL latency field in the Mode Register. This mode is provided for Burst Flash devices for which the deassertion of the BFOE signal causes an irreversible break of the burst. Figure 103 on page 218 shows the access request to the BFC and the deassertion of the BFOE signal due to a deselection of the BFC (Suspend). When the BFC is requested again, a new burst is started even though the requested address is sequential to the previously requested address.

**Figure 103.** Burst Flash Controller with No Burst Enable Handling



(1) Only if Multiplexed Address & Data Busses  
 (2) Master Clock Mode (BFCC =1)

No Burst Output Enable Handling (BFOEH = 0)  
 Signal Control Advance Address (BAAEN = 1)

**Continuous Burst Reads**

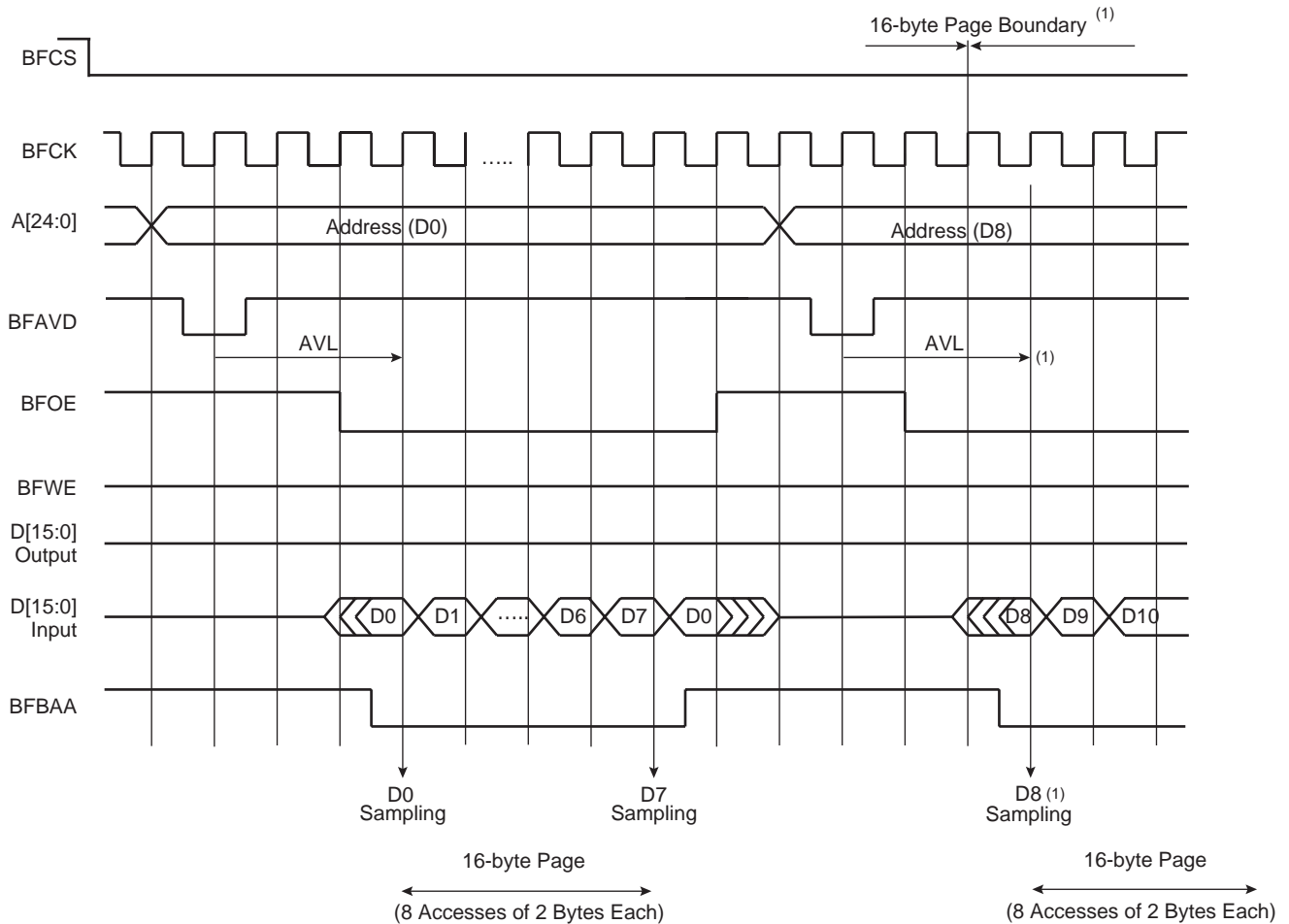
The BFC performs continuous burst reads. It is also possible to program page sizes from 16 bytes up to 1024 bytes. This is done by setting the appropriate value in the PAGES field of the “Burst Flash Controller Mode Register” on page 221.

*Page Mode*

In Page Mode, the BFC stops the current burst and starts a new burst each time the requested address matches a page boundary. Figure 104 on page 219 illustrates a 16-byte page size. Data D0 to D10 belong to two separate pages and are accessed through two burst accesses. This mode is provided for Burst Flash devices that cannot handle continuous burst read (in which case, a continuous burst access to address D0 would cause the Burst Flash internal

address to wrap around address D0). Page Mode can be disabled by programming a null value in the PAGES field of the “Burst Flash Controller Mode Register” on page 221.

**Figure 104.** Burst Read in Page Mode



Burst Read in Page Mode (16 Bytes)  
 Signal Control Advance Address (BAAEN = 1)  
 (1) A New Page Begins at D8

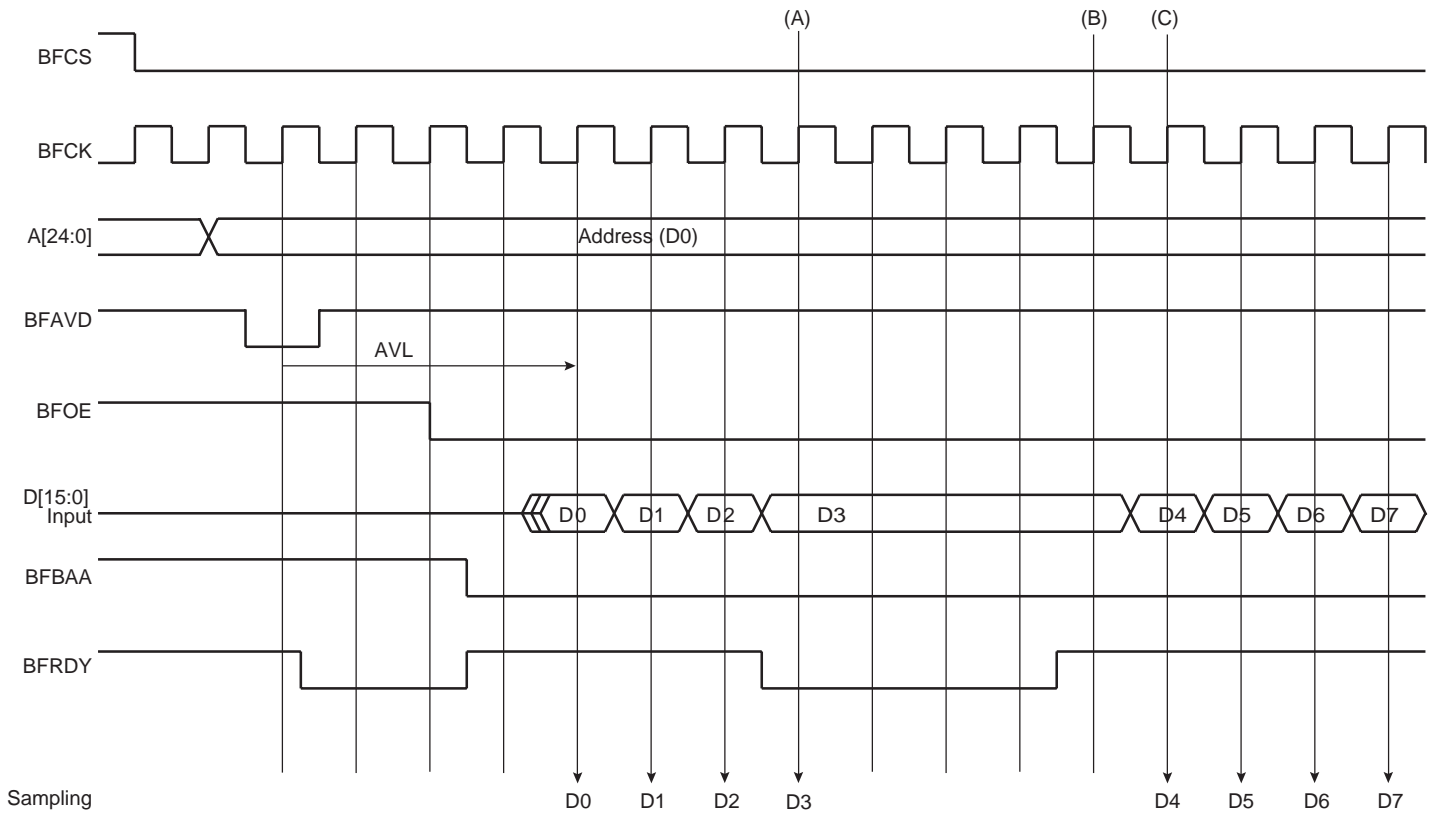
Address Valid Latency = 3 BFCK cycles (AVL field = 2)  
 Output Enable Latency (OEL) = 1 BFCK cycle  
 Page Size = 16 Bytes

### Ready Enable Mode

In Ready Enable Mode (bit RDYEN in the “Burst Flash Controller Mode Register” on page 221), the BFC uses the Ready Signal (BFRDY) from the burst Flash device as an indicator of the next data availability. The BFRDY signal must be asserted one BFCK cycle before data is valid. In Figure 105 on page 220 below, the BFRDY signal indicates on edge (A) that the expected D4 data will not be available on the next rising BFCK edge. The BFRDY signal remains low until rising at edge (B). D4 is then sampled on edge (C).

When the RDYEN mode is disabled (RDYEN = 0), the BFRDY signal at the BFC input interface is ignored. This mode is provided for Burst Flash devices that do not handle the BFRDY signal.

**Figure 105.** Burst Read Using BFRDY Signal



Burst Read  
Signal Control Advance Address (BAAEN = 1)

Address Valid Latency = 4 BFCK cycles (AVL field = 3)  
Output Enable Latency (OEL) = 1 BFCK cycle

## Burst Flash Controller (BFC) User Interface

### Burst Flash Controller Mode Register

Register Name: BFC\_MR  
 Access Type: Read/Write  
 Reset Value: 0x0

|     |    |     |    |       |       |       |       |
|-----|----|-----|----|-------|-------|-------|-------|
| 31  | 30 | 29  | 28 | 27    | 26    | 25    | 24    |
| –   | –  | –   | –  | –     | –     | –     | –     |
| 23  | 22 | 21  | 20 | 19    | 18    | 17    | 16    |
| –   | –  | –   | –  | RDYEN | MUXEN | BFOEH | BAAEN |
| 15  | 14 | 13  | 12 | 11    | 10    | 9     | 8     |
| –   | –  | OEL |    |       | PAGES |       |       |
| 7   | 6  | 5   | 4  | 3     | 2     | 1     | 0     |
| AVL |    |     |    | BFCC  |       | BFCOM |       |

- **BFCOM: Burst Flash Controller Operating Mode**

| BFCOM |   | BFC Operating Mode |
|-------|---|--------------------|
| 0     | 0 | Disabled.          |
| 0     | 1 | Asynchronous       |
| 1     | 0 | Burst Read         |
| 1     | 1 | Reserved           |

- **BFCC: Burst Flash Controller Clock**

| BFCC |   | BFC Clock                 |
|------|---|---------------------------|
| 0    | 0 | Reserved                  |
| 0    | 1 | Master Clock              |
| 1    | 0 | Master Clock divided by 2 |
| 1    | 1 | Master Clock divided by 4 |

- **AVL: Address Valid Latency**

The Address Valid Latency is defined as the number of BFC Clock Cycles from the first BFCK rising edge when BFAVD is asserted to the BFCK rising edge that samples read data. The Latency is equal to AVL + 1.

- **PAGES: Page Size**

This field defines the page size handling and the page size.

| Pages |   |   | Page Size   |
|-------|---|---|---|
| 0     | 0 | 0 | No page handling. The Ready Signal (BFRDY) is sampled to check if the next data is available. |
| 0     | 0 | 1 | 16 bytes page size  |
| 0     | 1 | 0 | 32 bytes page size  |
| 0     | 1 | 1 | 64 bytes page size  |
| 1     | 0 | 0 | 128 bytes page size   |
| 1     | 0 | 1 | 256 bytes page size   |
| 1     | 1 | 0 | 512 bytes page size   |
| 1     | 1 | 1 | 1024 bytes page size  |

- **OEL: Output Enable Latency**

This field defines the number of idle cycles inserted after each level change on the BFOE output enable signal. OEL range is 1 to 3.

- **BAAEN: Burst Address Advance Enable**

0: The burst clock is enabled to increment the burst address or, disabled to remain at the same address.

1: The burst clock is continuous and the burst address advance is controlled with the BFBA pin.

- **BFOEH: Burst Flash Output Enable Handling**

0: No burst resume in Burst Mode. When the BFC is deselected, this causes an irreversible break of the burst. A new burst will be initiated for the next access.

1: Burst resume. When the BFC is deselected, the burst is suspended. It will be resumed if the next access is sequential to the last one.

- **MUXEN: Multiplexed Bus Enable**

0: The address and data busses operate independently.

1: The address and data busses are multiplexed. Actually, the address is presented on both the data bus and the address bus when the BFAVD signal is asserted.

- **RDYEN: Ready Enable Mode**

0: The BFRDY input signal at the BFC input interface is ignored.

1: The BFRDY input signal is used as an indicator of data availability in the next cycle.

## Peripheral Data Controller (PDC)

### Overview

The Peripheral Data Controller (PDC) transfers data between on-chip serial peripherals such as the UART, USART, SSC, SPI, MCI and the on- and off-chip memories. Using the Peripheral Data Controller avoids processor intervention and removes the processor interrupt-handling overhead. This significantly reduces the number of clock cycles required for a data transfer and, as a result, improves the performance of the microcontroller and makes it more power efficient.

The PDC channels are implemented in pairs, each pair being dedicated to a particular peripheral. One channel in the pair is dedicated to the receiving channel and one to the transmitting channel of each UART, USART, SSC and SPI.

The user interface of a PDC channel is integrated in the memory space of each peripheral. It contains:

- A 32-bit memory pointer register
- A 16-bit transfer count register
- A 32-bit register for next memory pointer
- A 16-bit register for next transfer count

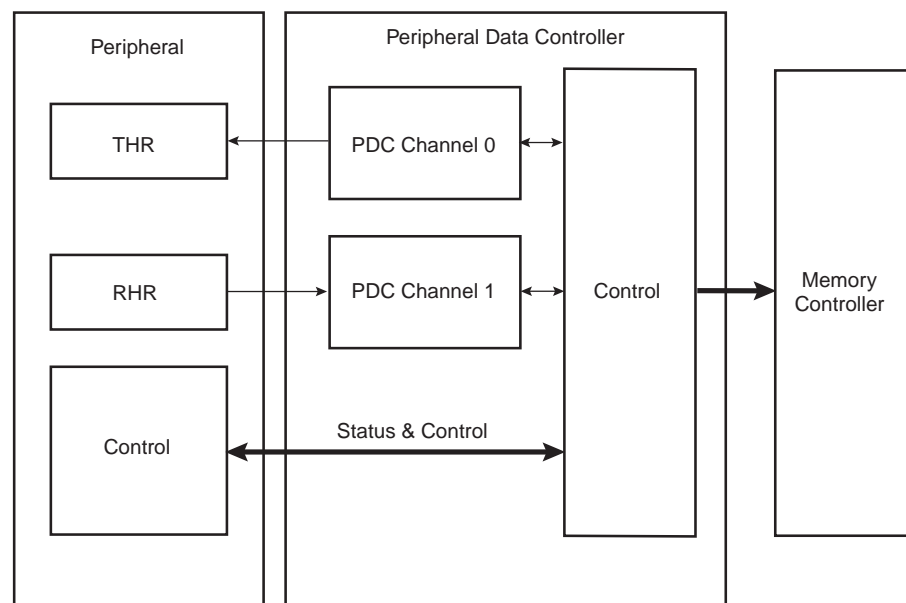
The peripheral triggers PDC transfers using transmit and receive signals. When the programmed data is transferred, an end of transfer interrupt is generated by the corresponding peripheral.

Important features of the PDC are:

- Generates Transfers to/from Peripherals Such as DBGU, USART, SSC, SPI and MCI
- Supports Up to Twenty Channels (Product Dependent)
- One Master Clock Cycle Needed for a Transfer from Memory to Peripheral
- Two Master Clock Cycles Needed for a Transfer from Peripheral to Memory

### Block Diagram

Figure 106. Block Diagram



## Functional Description

### Configuration

The PDC channels user interface enables the user to configure and control the data transfers for each channel. The user interface of a PDC channel is integrated into the user interface of the peripheral (offset 0x100), which it is related to.

Per peripheral, it contains four 32-bit Pointer Registers (RPR, RNPR, TPR, and TNPR) and four 16-bit Counter Registers (RCR, RNCR, TCR, and TNCR).

The size of the buffer (number of transfers) is configured in an internal 16-bit transfer counter register, and it is possible, at any moment, to read the number of transfers left for each channel.

The memory base address is configured in a 32-bit memory pointer by defining the location of the first address to access in the memory. It is possible, at any moment, to read the location in memory of the next transfer and the number of remaining transfers. The PDC has dedicated status registers which indicate if the transfer is enabled or disabled for each channel. The status for each channel is located in the peripheral status register. Transfers can be enabled and/or disabled by setting TXTEN/TXTDIS and RXTEN/RXTDIS in PDC Transfer Control Register. These control bits enable reading the pointer and counter registers safely without any risk of their changing between both reads.

The PDC sends status flags to the peripheral visible in its status-register (ENDRX, ENDTX, RXBUFF, and TXBUFE).

ENDRX flag is set when the PERIPH\_RCR register reaches zero.

RXBUFF flag is set when both PERIPH\_RCR and PERIPH\_RNCR reach zero.

ENDTX flag is set when the PERIPH\_TCR register reaches zero.

TXBUFE flag is set when both PERIPH\_TCR and PERIPH\_TNCR reach zero.

These status flags are described in the peripheral status register.

### Memory Pointers

Each peripheral is connected to the PDC by a receiver data channel and a transmitter data channel. Each channel has an internal 32-bit memory pointer. Each memory pointer points to a location anywhere in the memory space (on-chip memory or external bus interface memory).

Depending on the type of transfer (byte, half-word or word), the memory pointer is incremented by 1, 2 or 4, respectively for peripheral transfers.

If a memory pointer is reprogrammed while the PDC is in operation, the transfer address is changed, and the PDC performs transfers using the new address.

### Transfer Counters

There is one internal 16-bit transfer counter for each channel used to count the size of the block already transferred by its associated channel. These counters are decremented after each data transfer. When the counter reaches zero, the transfer is complete and the PDC stops transferring data.

If the Next Counter Register is equal to zero, the PDC disables the trigger while activating the related peripheral end flag.

If the counter is reprogrammed while the PDC is operating, the number of transfers is updated and the PDC counts transfers from the new value.

Programming the Next Counter/Pointer registers chains the buffers. The counters are decremented after each data transfer as stated above, but when the transfer counter reaches zero,



the values of the Next Counter/Pointer are loaded into the Counter/Pointer registers in order to re-enable the triggers.

For each channel, two status bits indicate the end of the current buffer (ENDRX, ENTX) and the end of both current and next buffer (RXBUFF, TXBUFE). These bits are directly mapped to the peripheral status register and can trigger an interrupt request to the AIC.

The peripheral end flag is automatically cleared when one of the counter-registers (Counter or Next Counter Register) is written.

Note: When the Next Counter Register is loaded into the Counter Register, it is set to zero.

## Data Transfers

The peripheral triggers PDC transfers using transmit (TXRDY) and receive (RXRDY) signals.

When the peripheral receives an external character, it sends a Receive Ready signal to the PDC which then requests access to the system bus. When access is granted, the PDC starts a read of the peripheral Receive Holding Register (RHR) and then triggers a write in the memory.

After each transfer, the relevant PDC memory pointer is incremented and the number of transfers left is decremented. When the memory block size is reached, a signal is sent to the peripheral and the transfer stops.

The same procedure is followed, in reverse, for transmit transfers.

## Priority of PDC Transfer Requests

The Peripheral Data Controller handles transfer requests from the channel according to priorities fixed for each product. These priorities are defined in the product datasheet.

If simultaneous requests of the same type (receiver or transmitter) occur on identical peripherals, the priority is determined by the numbering of the peripherals.

If transfer requests are not simultaneous, they are treated in the order they occurred. Requests from the receivers are handled first and then followed by transmitters requests.

## Peripheral Data Controller (PDC) User Interface

**Table 57.** Register Mapping

| Offset | Register                           | Register Name              | Read/Write | Reset |
|--------|------------------------------------|----------------------------|------------|-------|
| 0x100  | PDC Receive Pointer Register       | PERIPH <sup>(1)</sup> _RPR | Read/Write | 0x0   |
| 0x104  | PDC Receive Counter Register       | PERIPH_RCR                 | Read/Write | 0x0   |
| 0x108  | PDC Transmit Pointer Register      | PERIPH_TPR                 | Read/Write | 0x0   |
| 0x10C  | PDC Transmit Counter Register      | PERIPH_TCR                 | Read/Write | 0x0   |
| 0x110  | PDC Receive Next Pointer Register  | PERIPH_RNPR                | Read/Write | 0x0   |
| 0x114  | PDC Receive Next Counter Register  | PERIPH_RNCR                | Read/Write | 0x0   |
| 0x118  | PDC Transmit Next Pointer Register | PERIPH_TNPR                | Read/Write | 0x0   |
| 0x11C  | PDC Transmit Next Counter Register | PERIPH_TNCR                | Read/Write | 0x0   |
| 0x120  | PDC Transfer Control Register      | PERIPH_PTCR                | Write-only | -     |
| 0x114  | PDC Transfer Status Register       | PERIPH_PTISR               | Read-only  | 0x0   |

Note: 1. PERIPH: Ten registers are mapped in the peripheral memory space at the same offset. These can be defined by the user according to the function and the peripheral desired (DBGU, USART, SSC, SPI, MCI etc).

### PDC Receive Pointer Register

**Register Name:** PERIPH\_RPR

**Access Type:** Read/Write

|       |    |    |    |    |    |    |    |
|-------|----|----|----|----|----|----|----|
| 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 Address**

Address of the next receive transfer.

## PDC Receive Counter Register

**Register Name:** PERIPH\_RCR

**Access Type:** Read/Write

|       |    |    |    |    |    |    |    |
|-------|----|----|----|----|----|----|----|
| 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 Value**

Number of receive transfers to be performed.

## PDC Transmit Pointer Register

**Register Name:** PERIPH\_TPR

**Access Type:** Read/Write

|       |    |    |    |    |    |    |    |
|-------|----|----|----|----|----|----|----|
| 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 Address**

Address of the transmit buffer.

## PDC Transmit Counter Register

**Register Name:** PERIPH\_TCR

**Access Type:** Read/Write

|       |    |    |    |    |    |    |    |
|-------|----|----|----|----|----|----|----|
| 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 Value**

TXCTR is the size of the transmit transfer to be performed. At zero, the peripheral data transfer is stopped.

## PDC Receive Next Pointer Register

**Register Name:** PERIPH\_RNPR

**Access Type:** Read/Write

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

- **RXNPTR: Receive Next Pointer Address**

RXNPTR is the address of the next buffer to fill with received data when the current buffer is full.

## PDC Receive Next Counter Register

**Register Name:** PERIPH\_RNCR

**Access Type:** Read/Write

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

- **RXNCR: Receive Next Counter Value**

·RXNCR is the size of the next buffer to receive.

## PDC Transmit Next Pointer Register

**Register Name:** PERIPH\_TNPR

**Access Type:** Read/Write

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

- **TXNPTR: Transmit Next Pointer Address**

TXNPTR is the address of the next buffer to transmit when the current buffer is empty.

## PDC Transmit Next Counter Register

**Register Name:** PERIPH\_TNCR

**Access Type:** Read/Write

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

- **TXNCR: Transmit Next Counter Value**

·TXNCR is the size of the next buffer to transmit.

## PDC Transfer Control Register

**Register Name:** PERIPH\_PTCR

**Access Type:** Write-only

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

- **·RXTEN: Receiver Transfer Enable**

0 = No effect.

1 = Enables the receiver PDC transfer requests if RXTDIS is not set.

- **·RXTDIS: Receiver Transfer Disable**

0 = No effect.

1 = Disables the receiver PDC transfer requests.

- **·TXTEN: Transmitter Transfer Enable**

0 = No effect.

1 = Enables the transmitter PDC transfer requests.

- **·TXTDIS: Transmitter Transfer Disable**

0 = No effect.

1 = Disables the transmitter PDC transfer requests

## PDC Transfer Status Register

Register Name: PERIPH\_PTSR

Access Type: Read-only

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

- **-RXTEN: Receiver Transfer Enable**

0 = Receiver PDC transfer requests are disabled.

1 = Receiver PDC transfer requests are enabled.

- **-TXTEN: Transmitter Transfer Enable**

0 = Transmitter PDC transfer requests are disabled.

1 = Transmitter PDC transfer requests are enabled.

## Advanced Interrupt Controller (AIC)

### Overview

The Advanced Interrupt Controller (AIC) is an 8-level priority, individually maskable, vectored interrupt controller, providing handling of up to thirty-two interrupt sources. It is designed to substantially reduce the software and real-time overhead in handling internal and external interrupts.

The AIC drives the nFIQ (fast interrupt request) and the nIRQ (standard interrupt request) inputs of an ARM processor. Inputs of the AIC are either internal peripheral interrupts or external interrupts coming from the product's pins.

The 8-level Priority Controller allows the user to define the priority for each interrupt source, thus permitting higher priority interrupts to be serviced even if a lower priority interrupt is being treated.

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

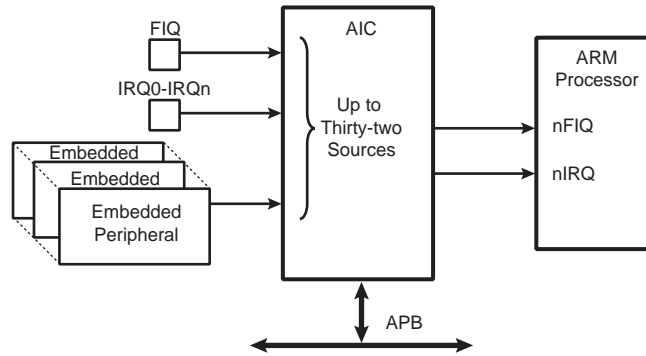
The fast forcing feature redirects any internal or external interrupt source to provide a fast interrupt rather than a normal interrupt.

Important Features of the AIC are:

- Controls the Interrupt Lines (nIRQ and nFIQ) of an ARM® Processor
- Thirty-two Individually Maskable and Vectored Interrupt Sources
  - Source 0 is Reserved for the Fast Interrupt Input (FIQ)
  - Source 1 is Reserved for System Peripherals (ST, RTC, PMC, DBGU...)
  - Source 2 to Source 31 Control up to Thirty Embedded Peripheral Interrupts or External Interrupts
  - Programmable Edge-triggered or Level-sensitive Internal Sources
  - Programmable Positive/Negative Edge-triggered or High/Low Level-sensitive External Sources
- 8-level Priority Controller
  - Drives the Normal Interrupt of the Processor
  - Handles Priority of the Interrupt Sources 1 to 31
  - Higher Priority Interrupts Can Be Served During Service of Lower Priority Interrupt
- Vectoring
  - Optimizes Interrupt Service Routine Branch and Execution
  - One 32-bit Vector Register per Interrupt Source
  - Interrupt Vector Register Reads the Corresponding Current Interrupt Vector
- Protect Mode
  - Easy Debugging by Preventing Automatic Operations when Protect Models Are Enabled
- Fast Forcing
  - Permits Redirecting any Normal Interrupt Source on the Fast Interrupt of the Processor
- General Interrupt Mask
  - Provides Processor Synchronization on Events Without Triggering an Interrupt

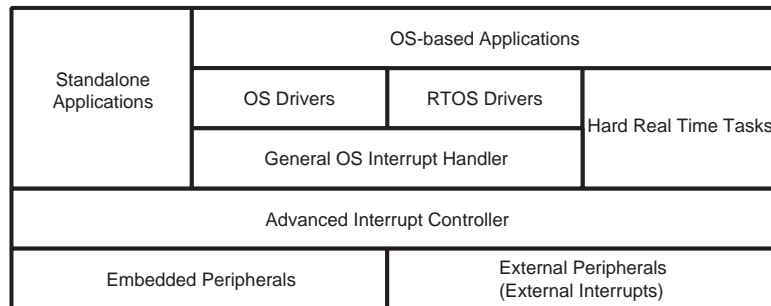
## Block Diagram

Figure 107. Block Diagram



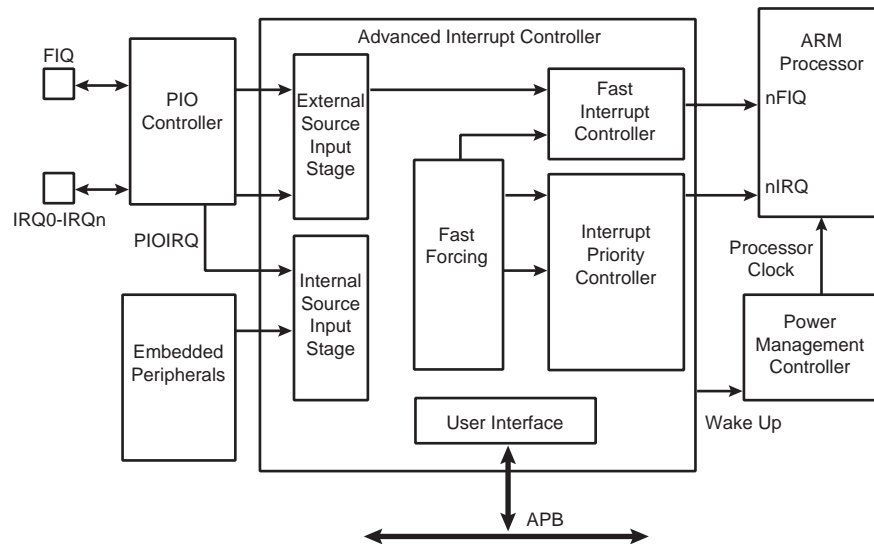
## Application Block Diagram

Figure 108. Description of the Application Block



## AIC Detailed Block Diagram

Figure 109. AIC Detailed Block Diagram





## I/O Line Description

**Table 58.** I/O Line Description

| Pin Name    | Pin Description           | Type  |
|-------------|---------------------------|-------|
| FIQ         | Fast Interrupt            | Input |
| IRQ0 - IRQn | Interrupt 0 - Interrupt n | Input |

## Product Dependencies

### I/O Lines

The interrupt signals FIQ and IRQ0 to IRQn are normally multiplexed through the PIO controllers. Depending on the features of the PIO controller used in the product, the pins must be programmed in accordance with their assigned interrupt function. This is not applicable when the PIO controller used in the product is transparent on the input path.

### Power Management

The Advanced Interrupt Controller is continuously clocked. The Power Management Controller has no effect on the Advanced Interrupt Controller behavior.

The assertion of the Advanced Interrupt Controller outputs, either nIRQ or nFIQ, wakes up the ARM processor while it is in Idle Mode. The General Interrupt Mask feature enables the AIC to wake up the processor without asserting the interrupt line of the processor, thus providing synchronization of the processor on an event.

### Interrupt Sources

The Interrupt Source 0 is always located at FIQ. If the product does not feature an FIQ pin, the Interrupt Source 0 cannot be used.

The Interrupt Source 1 is always located at System Interrupt. This is the result of the OR-wiring of the system peripheral interrupt lines, such as the System Timer, the Real Time Clock, the Power Management Controller and the Memory Controller. When a system interrupt occurs, the service routine must first distinguish the cause of the interrupt. This is performed by reading successively the status registers of the above mentioned system peripherals.

The interrupt sources 2 to 31 can either be connected to the interrupt outputs of an embedded user peripheral or to external interrupt lines. The external interrupt lines can be connected directly, or through the PIO Controller.

The PIO Controllers are considered as user peripherals in the scope of interrupt handling. Accordingly, the PIO Controller interrupt lines are connected to the Interrupt Sources 2 to 31.

The peripheral identification defined at the product level corresponds to the interrupt source number (as well as the bit number controlling the clock of the peripheral). Consequently, to simplify the description of the functional operations and the user interface, the interrupt sources are named FIQ, SYS, and PID2 to PID31.

## Functional Description

### Interrupt Source Control

**Interrupt Source Mode** The Advanced Interrupt Controller independently programs each interrupt source. The SRC-TYPE field of the corresponding AIC\_SMR (Source Mode Register) selects the interrupt condition of each source.

The internal interrupt sources wired on the interrupt outputs of the embedded peripherals can be programmed either in level-sensitive mode or in edge-triggered mode. The active level of the internal interrupts is not important for the user.

The external interrupt sources can be programmed either in high level-sensitive or low level-sensitive modes, or in positive edge-triggered or negative edge-triggered modes.

### Interrupt Source Enabling

Each interrupt source, including the FIQ in source 0, can be enabled or disabled by using the command registers; AIC\_IECR (Interrupt Enable Command Register) and AIC\_IDCR (Interrupt Disable Command Register). This set of registers conducts enabling or disabling in one instruction. The interrupt mask can be read in the AIC\_IMR register. A disabled interrupt does not affect servicing of other interrupts.

### Interrupt Clearing and Setting

All interrupt sources programmed to be edge-triggered (including the FIQ in source 0) can be individually set or cleared by writing respectively the AIC\_ISCR and AIC\_ICCR registers. Clearing or setting interrupt sources programmed in level-sensitive mode has no effect.

The clear operation is perfunctory, as the software must perform an action to reinitialize the “memorization” circuitry activated when the source is programmed in edge-triggered mode. However, the set operation is available for auto-test or software debug purposes. It can also be used to execute an AIC-implementation of a software interrupt.

The AIC features an automatic clear of the current interrupt when the AIC\_IVR (Interrupt Vector Register) is read. Only the interrupt source being detected by the AIC as the current interrupt is affected by this operation. (See “Priority Controller” on page 237.) The automatic clear reduces the operations required by the interrupt service routine entry code to reading the AIC\_IVR. Note that the automatic interrupt clear is disabled if the interrupt source has the Fast Forcing feature enabled as it is considered uniquely as a FIQ source. (For further details, See “Fast Forcing” on page 241.)

The automatic clear of the interrupt source 0 is performed when AIC\_FVR is read.

### Interrupt Status

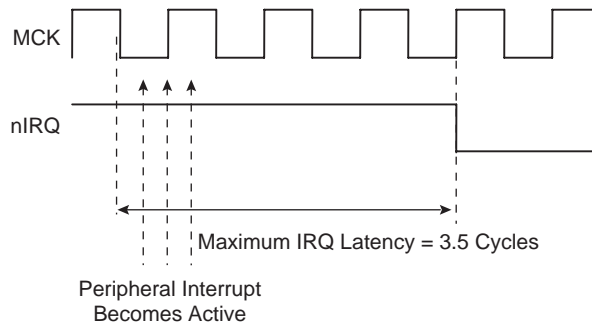
For each interrupt, the AIC operation originates in AIC\_IPR (Interrupt Pending Register) and its mask in AIC\_IMR (Interrupt Mask Register). AIC\_IPR enables the actual activity of the sources, whether masked or not.

The AIC\_ISR register reads the number of the current interrupt (see “Priority Controller” on page 237) and the register AIC\_CISR gives an image of the signals nIRQ and nFIQ driven on the processor.

Each status referred to above can be used to optimize the interrupt handling of the systems.

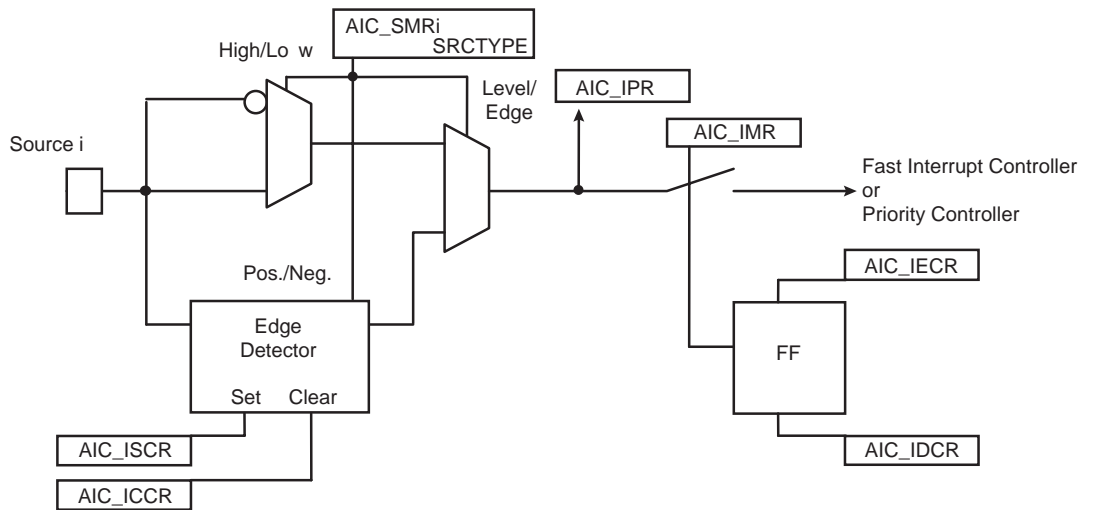
**Internal Interrupt Source Input Stage**

**Figure 110.** Internal Interrupt Source Input Stage



**External Interrupt Source Input Stage**

**Figure 111.** External Interrupt Source Input Stage



## Interrupt Latencies

Global interrupt latencies depend on several parameters, including:

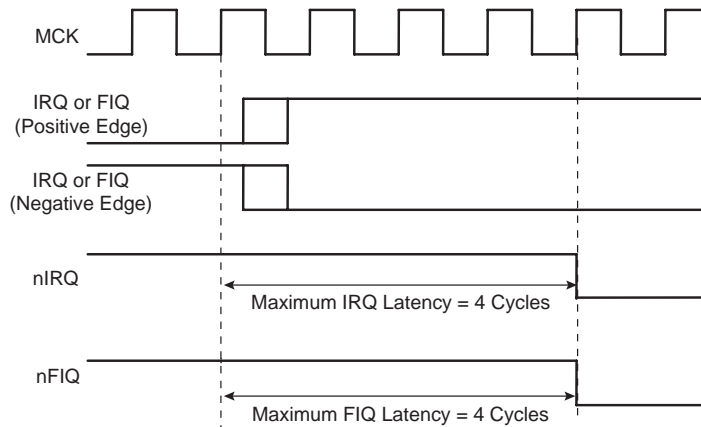
- The time the software masks the interrupts.
- Occurrence, either at the processor level or at the AIC level.
- The execution time of the instruction in progress when the interrupt occurs.
- The treatment of higher priority interrupts and the resynchronization of the hardware signals.

This section addresses only the hardware resynchronizations. It gives details of the latency times between the event on an external interrupt leading in a valid interrupt (edge or level) or the assertion of an internal interrupt source and the assertion of the nIRQ or nFIQ line on the processor. The resynchronization time depends on the programming of the interrupt source and on its type (internal or external). For the standard interrupt, resynchronization times are given assuming there is no higher priority in progress.

The PIO Controller multiplexing has no effect on the interrupt latencies of the external interrupt sources.

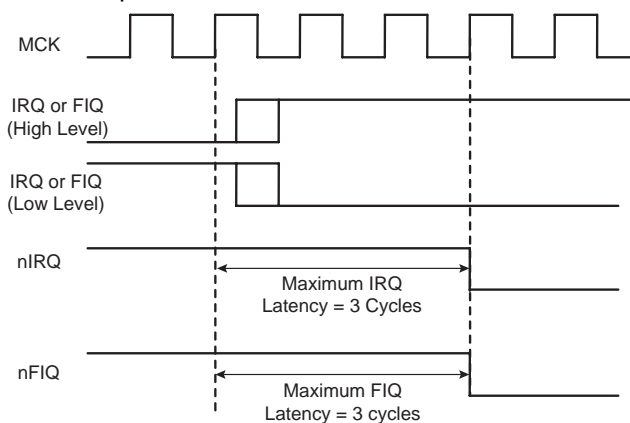
### External Interrupt Edge Triggered Source

**Figure 112.** External Interrupt Edge Triggered Source



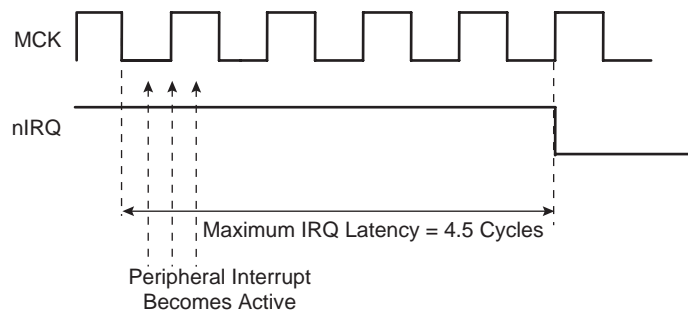
### External Interrupt Level Sensitive Source

**Figure 113.** External Interrupt Level Sensitive Source



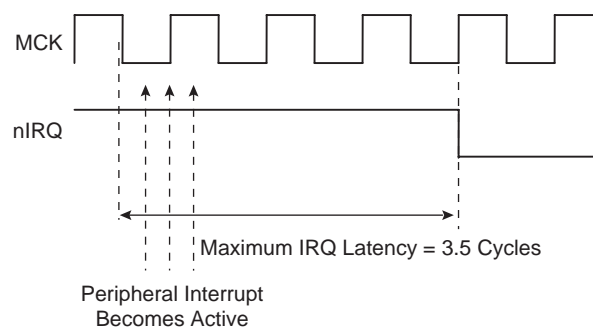
**Internal Interrupt Edge Triggered Source**

**Figure 114.** Internal Interrupt Edge Triggered Source



**Internal Interrupt Level Sensitive Source**

**Figure 115.** Internal Interrupt Level Sensitive Source



**Normal Interrupt**

**Priority Controller**

An 8-level priority controller drives the nIRQ line of the processor, depending on the interrupt conditions occurring on the interrupt sources 1 to 31 (except for those programmed in Fast Forcing).

Each interrupt source has a programmable priority level of 7 to 0, which is user-definable by writing the PRIOR field of the corresponding AIC\_SMR (Source Mode Register). Level 7 is the highest priority and level 0 the lowest.

As soon as an interrupt condition occurs, as defined by the SRCTYPE field of the AIC\_SVR (Source Vector Register), the nIRQ line is asserted. As a new interrupt condition might have happened on other interrupt sources since the nIRQ has been asserted, the priority controller determines the current interrupt at the time the AIC\_IVR (Interrupt Vector Register) is read. **The read of AIC\_IVR is the entry point of the interrupt handling** which allows the AIC to consider that the interrupt has been taken into account by the software.

The current priority level is defined as the priority level of the current interrupt.

If several interrupt sources of equal priority are pending and enabled when the AIC\_IVR is read, the interrupt with the lowest interrupt source number is serviced first.

The nIRQ line can be asserted only if an interrupt condition occurs on an interrupt source with a higher priority. If an interrupt condition happens (or is pending) during the interrupt treatment in progress, it is delayed until the software indicates to the AIC the end of the current service by writing the AIC\_EOICR (End of Interrupt Command Register). **The write of AIC\_EOICR is the exit point of the interrupt handling.**

## Interrupt Nesting

The priority controller utilizes interrupt nesting in order for the highest priority interrupt to be handled during the service of lower priority interrupts. This requires the interrupt service routines of the lower interrupts to re-enable the interrupt at the processor level.

When an interrupt of a higher priority happens during an already occurring interrupt service routine, the nIRQ line is re-asserted. If the interrupt is enabled at the core level, the current execution is interrupted and the new interrupt service routine should read the AIC\_IVR. At this time, the current interrupt number and its priority level are pushed into an embedded hardware stack, so that they are saved and restored when the higher priority interrupt servicing is finished and the AIC\_EOICR is written.

The AIC is equipped with an 8-level wide hardware stack in order to support up to eight interrupt nestings pursuant to having eight priority levels.

## Interrupt Vectoring

The interrupt handler addresses corresponding to each interrupt source can be stored in the registers AIC\_SVR1 to AIC\_SVR31 (Source Vector Register 1 to 31). When the processor reads AIC\_IVR (Interrupt Vector Register), the value written into AIC\_SVR corresponding to the current interrupt is returned.

This feature offers a way to branch in one single instruction to the handler corresponding to the current interrupt, as AIC\_IVR is mapped at the absolute address 0xFFFF F100 and thus accessible from the ARM interrupt vector at address 0x0000 0018 through the following instruction:

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

When the processor executes this instruction, it loads the read value in AIC\_IVR in its program counter, thus branching the execution on the correct interrupt handler.

This feature is often not used when the application is based on an operating system (either real time or not). Operating systems often have a single entry point for all the interrupts and the first task performed is to discern the source of the interrupt.

However, it is strongly recommended to port the operating system on AT91 products by supporting the interrupt vectoring. This can be performed by defining all the AIC\_SVR of the interrupt source to be handled by the operating system at the address of its interrupt handler. When doing so, the interrupt vectoring permits a critical interrupt to transfer the execution on a specific very fast handler and not onto the operating system's general interrupt handler. This facilitates the support of hard real-time tasks (input/outputs of voice/audio buffers and software peripheral handling) to be handled efficiently and independently of the application running under an operating system.

## Interrupt Handlers

This section gives an overview of the fast interrupt handling sequence when using the AIC. It is assumed that the programmer understands the architecture of the ARM processor, and especially the processor interrupt modes and the associated status bits.

It is assumed that:

1. The Advanced Interrupt Controller has been programmed, AIC\_SVR registers are loaded with corresponding interrupt service routine addresses and interrupts are enabled.
2. The instruction at the ARM interrupt exception vector address is required to work with the vectoring

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

When nIRQ is asserted, if the bit "I" of CPSR is 0, the sequence is as follows:

1. The CPSR is stored in SPSR\_irq, the current value of the Program Counter is loaded in the Interrupt 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 four.

2. The ARM core enters Interrupt mode, if it has not already done so.
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:
  - Sets the current interrupt to be the pending and enabled interrupt with the highest priority. The current level is the priority level of the current interrupt.
  - De-asserts the nIRQ line on the processor. Even if vectoring is not used, AIC\_IVR must be read in order to de-assert nIRQ.
  - Automatically clears the interrupt, if it has been programmed to be edge-triggered.
  - Pushes the current level and the current interrupt number on to the stack.
  - Returns the value written in the AIC\_SVR corresponding to the current interrupt.
4. The previous step has the effect of branching to the corresponding interrupt service routine. This should start by saving the link register (R14\_irq) and SPSR\_IRQ. The link register must be decremented by four when it is saved if it is to be restored directly into the program counter at the end of the interrupt. For example, the instruction `SUB PC, LR, #4` may be used.
5. Further interrupts can then be unmasked by clearing the “I” bit in CPSR, allowing re-assertion of the nIRQ to be taken into account by the core. This can happen if an interrupt with a higher priority than the current interrupt occurs.
6. The interrupt handler can then proceed as required, saving the registers that will be used and restoring them at the end. During this phase, an interrupt of higher priority than the current level will restart the sequence from step 1.

Note: 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 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 the 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. 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 SPSR\_irq.

Note: The “I” bit in SPSR is significant. If it is set, it indicates that the ARM core was on the verge of masking an interrupt when the mask instruction was interrupted. Hence, when SPSR is restored, the mask instruction is completed (interrupt is masked).

## Fast Interrupt

### Fast Interrupt Source

The interrupt source 0 is the only source which can raise a fast interrupt request to the processor except if fast forcing is used. The interrupt source 0 is generally connected to a FIQ pin of the product, either directly or through a PIO Controller.

### Fast Interrupt Control

The fast interrupt logic of the AIC has no priority controller. The mode of interrupt source 0 is programmed with the AIC\_SMR0 and the field PRIOR of this register is not used even if it

reads what has been written. The field SRCTYPE of AIC\_SMR0 enables programming the fast interrupt source to be positive-edge triggered or negative-edge triggered or high-level sensitive or low-level sensitive

Writing 0x1 in the AIC\_IECR (Interrupt Enable Command Register) and AIC\_IDCR (Interrupt Disable Command Register) respectively enables and disables the fast interrupt. The bit 0 of AIC\_IMR (Interrupt Mask Register) indicates whether the fast interrupt is enabled or disabled.

### Fast Interrupt Vectoring

The fast interrupt handler address can be stored in AIC\_SVR0 (Source Vector Register 0). The value written into this register is returned when the processor reads AIC\_FVR (Fast Vector Register). This offers a way to branch in one single instruction to the interrupt handler, as AIC\_FVR is mapped at the absolute address 0xFFFF F104 and thus accessible from the ARM fast interrupt vector at address 0x0000 001C through the following instruction:

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

When the processor executes this instruction it loads the value read in AIC\_FVR in its program counter, thus branching the execution on the fast interrupt handler. It also automatically performs the clear of the fast interrupt source if it is programmed in edge-triggered mode.

### Fast Interrupt Handlers

This section gives an overview of the fast interrupt handling sequence when using the AIC. It is assumed that the programmer understands the architecture of the ARM processor, and especially the processor interrupt modes and associated status bits.

Assuming that:

1. The Advanced Interrupt Controller has been programmed, AIC\_SVR0 is loaded with the fast interrupt service routine address, and the interrupt source 0 is enabled.
2. The Instruction at address 0x1C (FIQ exception vector address) is required to vector the fast interrupt:

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

3. The user does not need nested fast interrupts.

When nFIQ is asserted if the bit "F" 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 four.
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, 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 enables branching to the corresponding interrupt service routine. It is not necessary to save the link register R14\_fiq and 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 interrupt source 0.
6. Finally, the Link Register R14\_fiq is restored into the PC after decrementing it by four (with instruction `SUB PC, LR, #4` for example). This has the effect of returning from the interrupt to whatever was being executed before, loading the CPSR with the SPSR



and masking or unmasking the fast interrupt depending on the state saved in the SPSR.

Note: The "F" bit in 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).

Another way to handle the fast interrupt is to map the interrupt service routine at the address of the ARM vector 0x1C. This method does not use the vectoring, so that reading AIC\_FVR must be performed at the very beginning of the handler operation. However, this method saves the execution of a branch instruction.

## Fast Forcing

The Fast Forcing feature of the advanced interrupt controller provides redirection of any normal Interrupt source on the fast interrupt controller.

Fast Forcing is enabled or disabled by writing to the Fast Forcing Enable Register (AIC\_FFER) and the Fast Forcing Disable Register (AIC\_FFDR). Writing to these registers results in an update of the Fast Forcing Status Register (AIC\_FFSR) that controls the feature for each internal or external interrupt source.

When Fast Forcing is disabled, the interrupt sources are handled as described in the previous pages.

When Fast Forcing is enabled, the edge/level programming and, in certain cases, edge detection of the interrupt source is still active but the source cannot trigger a normal interrupt to the processor and is not seen by the priority handler.

If the interrupt source is programmed in level-sensitive mode and an active level is sampled, Fast Forcing results in the assertion of the nFIQ line to the core.

If the interrupt source is programmed in edge-triggered mode and an active edge is detected, Fast Forcing results in the assertion of the nFIQ line to the core.

The Fast Forcing feature does not affect the Source 0 pending bit in the Interrupt Pending Register (AIC\_IPR).

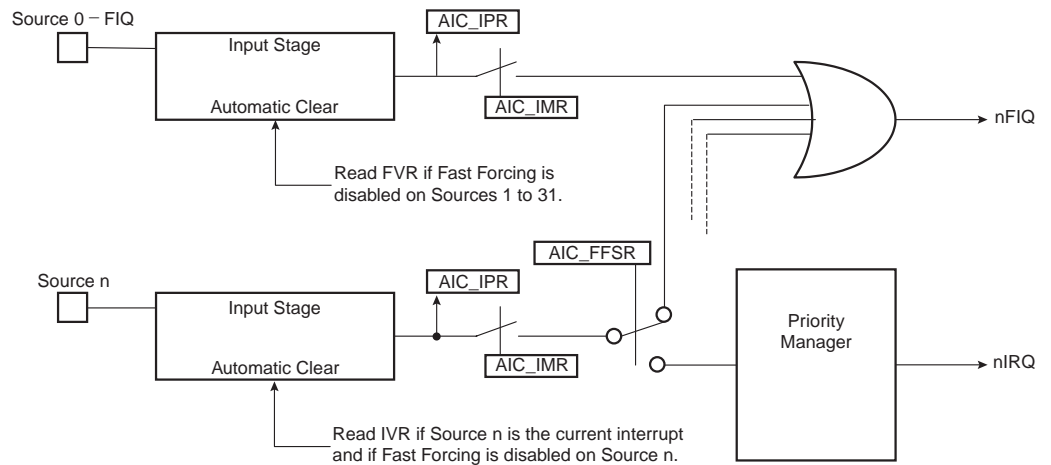
The Fast Interrupt Vector Register (AIC\_FVR) reads the contents of the Source Vector Register 0 (AIC\_SVR0), whatever the source of the fast interrupt may be. The read of the FVR does not clear the Source 0 when the fast forcing feature is used and the interrupt source should be cleared by writing to the Interrupt Clear Command Register (AIC\_ICCR).

All enabled and pending interrupt sources that have the fast forcing feature enabled and that are programmed in edge-triggered mode must be cleared by writing to the Interrupt Clear Command Register. In doing so, they are cleared independently and thus lost interrupts are prevented.

The read of AIC\_IVR does not clear the source that has the fast forcing feature enabled.

The source 0, reserved to the fast interrupt, continues operating normally and becomes one of the Fast Interrupt sources.

**Figure 116. Fast Forcing**



## Protect Mode

The Protect Mode permits reading the Interrupt Vector Register without performing the associated automatic operations. This is necessary when working with a debug system. When a debugger, working either with a Debug Monitor or the ARM processor's ICE, stops the applications and updates the opened windows, it might read the AIC User Interface and thus the IVR. This has undesirable consequences:

- 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 is necessary to acknowledge and to restore the context of the AIC. This operation is generally not performed by the debug system as the debug system would become strongly intrusive and cause the application to enter an undesired state.

This is avoided by using the Protect Mode. Writing DBGM in AIC\_DCR (Debug Control Register) at 0x1 enables the Protect Mode.

When the 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 AIC\_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 perform the same operations. However, it is recommended to not stop the processor between the read and the write of AIC\_IVR of the interrupt service routine to make sure the debugger does not modify the AIC context.

To summarize, in normal operating mode, the read of AIC\_IVR performs the following operations within the AIC:

1. Calculates active interrupt (higher than current or spurious).
2. Determines and returns the vector of the active interrupt.
3. Memorizes the interrupt.
4. Pushes the current priority level onto the internal stack.
5. Acknowledges the interrupt.

However, while the Protect Mode is activated, only operations 1 to 3 are performed when AIC\_IVR is read. Operations 4 and 5 are only performed by the AIC when AIC\_IVR is written.

Software that has been written and debugged using the Protect Mode runs 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.

## Spurious Interrupt

The Advanced Interrupt Controller features protection against spurious interrupts. A spurious interrupt is defined as being the assertion of an interrupt source long enough for the AIC to assert the nIRQ, but no longer present when AIC\_IVR is read. This is most prone to occur when:

- An external interrupt source is programmed in level-sensitive mode and an active level occurs for only a short time.
- An internal interrupt source is programmed in level sensitive and the output signal of the corresponding embedded peripheral is activated for a short time. (As in the case for the Watchdog.)
- An interrupt occurs just a few cycles before the software begins to mask it, thus resulting in a pulse on the interrupt source.

The AIC detects a spurious interrupt at the time the AIC\_IVR is read while no enabled interrupt source is pending. When this happens, the AIC returns the value stored by the programmer in AIC\_SPU (Spurious Vector Register). The programmer must store the address of a spurious interrupt handler in AIC\_SPU as part of the application, to enable an as fast as possible return to the normal execution flow. This handler writes in AIC\_EOICR and performs a return from interrupt.

## General Interrupt Mask

The AIC features a General Interrupt Mask bit to prevent interrupts from reaching the processor. Both the nIRQ and the nFIQ lines are driven to their inactive state if the bit GMSK in AIC\_DCR (Debug Control Register) is set. However, this mask does not prevent waking up the processor if it has entered Idle Mode. This function facilitates synchronizing the processor on a next event and, as soon as the event occurs, performs subsequent operations without having to handle an interrupt. It is strongly recommended to use this mask with caution.

## Advanced Interrupt Controller (AIC) User Interface

### Base Address

The AIC is mapped at the address **0xFFFF F000**. It has a total 4-Kbyte addressing space. This permits the vectoring feature, as the PC-relative load/store instructions of the ARM processor supports only an  $\pm 4$ -Kbyte offset.

**Table 59.** Register Mapping

| Offset | Register                           | Name      | Access     | Reset Value        |
|--------|------------------------------------|-----------|------------|--------------------|
| 0000   | Source Mode Register 0             | AIC_SMR0  | Read/Write | 0x0                |
| 0x04   | Source Mode Register 1             | AIC_SMR1  | Read/Write | 0x0                |
| –      | –                                  | –         | –          | –                  |
| 0x7C   | Source Mode Register 31            | AIC_SMR31 | Read/Write | 0x0                |
| 0x80   | Source Vector Register 0           | AIC_SVR0  | Read/Write | 0x0                |
| 0x84   | Source Vector Register 1           | AIC_SVR1  | Read/Write | 0x0                |
| –      | –                                  | –         | –          | –                  |
| 0xFC   | Source Vector Register 31          | AIC_SVR31 | Read/Write | 0x0                |
| 0x100  | Interrupt Vector Register          | AIC_IVR   | Read-only  | 0x0                |
| 0x104  | Fast Interrupt Vector Register     | AIC_FVR   | Read-only  | 0x0                |
| 0x108  | Interrupt Status Register          | AIC_ISR   | Read-only  | 0x0                |
| 0x10C  | Interrupt Pending Register         | AIC_IPR   | Read-only  | 0x0 <sup>(1)</sup> |
| 0x110  | Interrupt Mask Register            | AIC_IMR   | Read-only  | 0x0                |
| 0x114  | Core Interrupt Status Register     | AIC_CISR  | Read-only  | 0x0                |
| 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 Interrupt Vector Register | AIC_SPU   | Read/Write | 0x0                |
| 0x138  | Debug Control Register             | AIC_DCR   | Read/Write | 0x0                |
| 0x13C  | Reserved                           | –         | –          | –                  |
| 0x140  | Fast Forcing Enable Register       | AIC_FFER  | Write-only | –                  |
| 0x144  | Fast Forcing Disable Register      | AIC_FFDR  | Write-only | –                  |
| 0x148  | Fast Forcing Status Register       | AIC_FFSR  | Read-only  | 0x0                |

Note: 1. The reset value of the Interrupt Pending Register depends on the level of the external interrupt source. All other sources are cleared at reset, thus not pending.

## AIC Source Mode Register

**Register Name:** AIC\_SMR0..AIC\_SMR31

**Access Type:** Read/write

**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  |   |
| –  | SRCTYPE |    | –  | –  | PRIOR |    |    | – |

- **PRIOR: Priority Level**

Programs the priority level for all sources except FIQ source (source 0).

The priority level can be between 0 (lowest) and 7 (highest).

The priority level is not used for the FIQ in the related SMR register AIC\_SMRx.

- **SRCTYPE: Interrupt Source Type**

The active level or edge is not programmable for the internal interrupt sources.

| SRCTYPE |   | Internal Interrupt Sources |
|---------|---|----------------------------|
| 0       | 0 | Level Sensitive            |
| 0       | 1 | Edge Triggered             |
| 1       | 0 | Level Sensitive            |
| 1       | 1 | Edge Triggered             |

## AIC Source Vector Register

**Register Name:** AIC\_SVR0..AIC\_SVR31

**Access Type:** Read/Write

**Reset Value:** 0x0

|        |    |    |    |    |    |    |    |
|--------|----|----|----|----|----|----|----|
| 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: Source Vector**

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

**Reset Value:** 0

|      |    |    |    |    |    |    |    |
|------|----|----|----|----|----|----|----|
| 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 Interrupt Vector Register contains the vector programmed by the user in the Source Vector Register corresponding to the current interrupt.

The Source Vector Register is indexed using the current interrupt number when the Interrupt Vector Register is read.

When there is no current interrupt, the Interrupt Vector Register reads the value stored in AIC\_SPU.

**AIC FIQ Vector Register**

**Register Name:** AIC\_FVR

**Access Type:** Read-only

**Reset Value:** 0

|      |    |    |    |    |    |    |    |
|------|----|----|----|----|----|----|----|
| 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. When there is no fast interrupt, the Fast Interrupt Vector Register reads the value stored in AIC\_SPU.

### AIC Interrupt Status Register

**Register Name:** AIC\_ISR  
**Access Type:** Read-only  
**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  |   |
| –  | –  | –  | IRQID |    |    |    |    | – |

- IRQID: Current Interrupt Identifier**

The Interrupt Status Register returns the current interrupt source number.

### AIC Interrupt Pending Register

**Register Name:** AIC\_IPR  
**Access Type:** Read-only  
**Reset Value:** 0

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 31    | 30    | 29    | 28    | 27    | 26    | 25    | 24    |
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23    | 22    | 21    | 20    | 19    | 18    | 17    | 16    |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9  | PID8  |
| 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
| PID7  | PID6  | PID5  | PID4  | PID3  | PID2  | SYS   | FIQ   |

- FIQ, SYS, PID2-PID31: Interrupt Pending**

0 = Corresponding interrupt is no pending.  
 1 = Corresponding interrupt is pending.

## AIC Interrupt Mask Register

**Register Name:** AIC\_IMR

**Access Type:** Read-only

**Reset Value:** 0

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 31    | 30    | 29    | 28    | 27    | 26    | 25    | 24    |
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23    | 22    | 21    | 20    | 19    | 18    | 17    | 16    |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9  | PID8  |
| 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
| PID7  | PID6  | PID5  | PID4  | PID3  | PID2  | SYS   | FIQ   |

- **FIQ, SYS, PID2-PID31: 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

**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    |
| –  | –  | –  | –  | –  | –  | NIRQ | NFIQ |

- **NFIQ: NFIQ Status**

0 = nFIQ line is deactivated.

1 = nFIQ line is active.

- **NIRQ: NIRQ Status**

0 = nIRQ line is deactivated.

1 = nIRQ line is active.



### AIC Interrupt Enable Command Register

Register Name: AIC\_IECR

Access Type: Write-only

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 31    | 30    | 29    | 28    | 27    | 26    | 25    | 24    |
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23    | 22    | 21    | 20    | 19    | 18    | 17    | 16    |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9  | PID8  |
| 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
| PID7  | PID6  | PID5  | PID4  | PID3  | PID2  | SYS   | FIQ   |

• **FIQ, SYS, PID2-PID3: Interrupt Enable**

0 = No effect.

1 = Enables corresponding interrupt.

### AIC Interrupt Disable Command Register

Register Name: AIC\_IDCR

Access Type: Write-only

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 31    | 30    | 29    | 28    | 27    | 26    | 25    | 24    |
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23    | 22    | 21    | 20    | 19    | 18    | 17    | 16    |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9  | PID8  |
| 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
| PID7  | PID6  | PID5  | PID4  | PID3  | PID2  | SYS   | FIQ   |

• **FIQ, SYS, PID2-PID31: Interrupt Disable**

0 = No effect.

1 = Disables corresponding interrupt.

## AIC Interrupt Clear Command Register

Register Name: AIC\_ICCR

Access Type: Write-only

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 31    | 30    | 29    | 28    | 27    | 26    | 25    | 24    |
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23    | 22    | 21    | 20    | 19    | 18    | 17    | 16    |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9  | PID8  |
| 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
| PID7  | PID6  | PID5  | PID4  | PID3  | PID2  | SYS   | FIQ   |

- **FIQ, SYS, PID2-PID31: Interrupt Clear**

0 = No effect.

1 = Clears corresponding interrupt.

## AIC Interrupt Set Command Register

Register Name: AIC\_ISCR

Access Type: Write-only

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 31    | 30    | 29    | 28    | 27    | 26    | 25    | 24    |
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23    | 22    | 21    | 20    | 19    | 18    | 17    | 16    |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9  | PID8  |
| 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
| PID7  | PID6  | PID5  | PID4  | PID3  | PID2  | SYS   | FIQ   |

- **FIQ, SYS, PID2-PID31: Interrupt Set**

0 = No effect.

1 = Sets corresponding interrupt.

### AIC End of Interrupt Command Register

Register Name: AIC\_EOICR

Access Type: Write-only

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 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 Interrupt Vector Register

Register Name: AIC\_SPU

Access Type: Read/Write

Reset Value: 0

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

- **SIQV: Spurious Interrupt Vector Register**

The user may store the address of a spurious interrupt handler in this register. The written value is returned in AIC\_IVR in case of a spurious interrupt and in AIC\_FVR in case of a spurious fast interrupt.

## AIC Debug Control Register

Register Name: AIC\_DEBUG

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    |
| –  | –  | –  | –  | –  | –  | GMSK | PROT |

- **PROT: Protection Mode**

0 = The Protection Mode is disabled.

1 = The Protection Mode is enabled.

- **GMSK: General Mask**

0 = The nIRQ and nFIQ lines are normally controlled by the AIC.

1 = The nIRQ and nFIQ lines are tied to their inactive state.

### AIC Fast Forcing Enable Register

Register Name: AIC\_FFER

Access Type: Write-only

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 31    | 30    | 29    | 28    | 27    | 26    | 25    | 24    |
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23    | 22    | 21    | 20    | 19    | 18    | 17    | 16    |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9  | PID8  |
| 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
| PID7  | PID6  | PID5  | PID4  | PID3  | PID2  | SYS   | –     |

• **SYS, PID2-PID31: Fast Forcing Enable**

0 = No effect.

1 = Enables the fast forcing feature on the corresponding interrupt.

### AIC Fast Forcing Disable Register

Register Name: AIC\_FFDR

Access Type: Write-only

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 31    | 30    | 29    | 28    | 27    | 26    | 25    | 24    |
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23    | 22    | 21    | 20    | 19    | 18    | 17    | 16    |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9  | PID8  |
| 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
| PID7  | PID6  | PID5  | PID4  | PID3  | PID2  | SYS   | –     |

• **SYS, PID2-PID31: Fast Forcing Disable**

0 = No effect.

1 = Disables the Fast Forcing feature on the corresponding interrupt.

## AIC Fast Forcing Status Register

Register Name: AIC\_FF SR

Access Type: Read-only

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 31    | 30    | 29    | 28    | 27    | 26    | 25    | 24    |
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23    | 22    | 21    | 20    | 19    | 18    | 17    | 16    |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9  | PID8  |
| 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
| PID7  | PID6  | PID5  | PID4  | PID3  | PID2  | SYS   | -     |

- **SYS, PID2 - PID31: Fast Forcing Status**

0 = The Fast Forcing feature is disabled on the corresponding interrupt.

1 = The Fast Forcing feature is enable on the corresponding interrupt.

## Power Management Controller (PMC)

### Overview

The Power Management Controller (PMC) generates all the clocks of a system thanks to the integration of two oscillators and two PLLs.

The PMC provides clocks to the embedded processor and enables the idle mode by stopping the processor clock until the next interrupt.

The PMC independently provides and controls up to thirty peripheral clocks and four programmable clocks that can be used as outputs on pins to feed external devices. The integration of the PLLs supplies the USB devices and host ports with a 48 MHz clock, as required by the bus speed, and the rest of the system with a clock at another frequency. Thus, the fully-featured Power Management Controller optimizes power consumption of the whole system and supports the Normal, Idle, Slow Clock and Standby operating modes.

The main features of the PMC are:

- Optimize the Power Consumption of the Whole System
- Embeds and Controls:
  - One Main Oscillator and One Slow Clock Oscillator (32.768 kHz)
  - Two Phase Locked Loops (PLLs) and Dividers
  - Clock Prescalers
- Provides:
  - the Processor Clock PCK
  - the Master Clock MCK
  - the USB Clocks, UHPCK and UDPCK, Respectively for the USB Host Port and the USB Device Port
  - Programmable Automatic PLL Switch-off in USB Device Suspend Conditions
  - up to Thirty Peripheral Clocks
  - up to Four Programmable Clock Outputs
- Four Operating Modes:
  - Normal Mode, Idle Mode, Slow Clock Mode, Standby Mode

## Product Dependencies

### I/O Lines

The Power Management Controller is capable of handling up to four Programmable Clocks, PCK0 to PCK3.

A Programmable Clock is generally multiplexed on a PIO Controller. The user must first program the PIO controllers to assign the pins of the Programmable Clock to its peripheral function.

### Interrupt

The Power Management Controller has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling the PMC interrupt requires programming the AIC before configuring the PMC.

### Oscillator and PLL Characteristics

The electrical characteristics of the embedded oscillators and PLLs are product-dependent, even if the way to control them is similar.

All of the parameters for both oscillators and the PLLs are given in the DC Characteristics section of the product datasheet. These figures are used not only for the hardware design, as they affect the external components to be connected to the pins, but also the software configuration, as they determine the waiting time for the startup and lock times to be programmed.

### Peripheral Clocks

The Power Management Controller provides and controls up to thirty peripheral clocks. The bit number permitting the control of a peripheral clock is the Peripheral ID of the embedded peripheral.

When the Peripheral ID does not correspond to a peripheral, either because this is an external interrupt or because there are less than thirty peripherals, the control bits of the Peripheral ID are not implemented in the PMC and programming them has no effect on the behavior of the PMC.

### USB Clocks

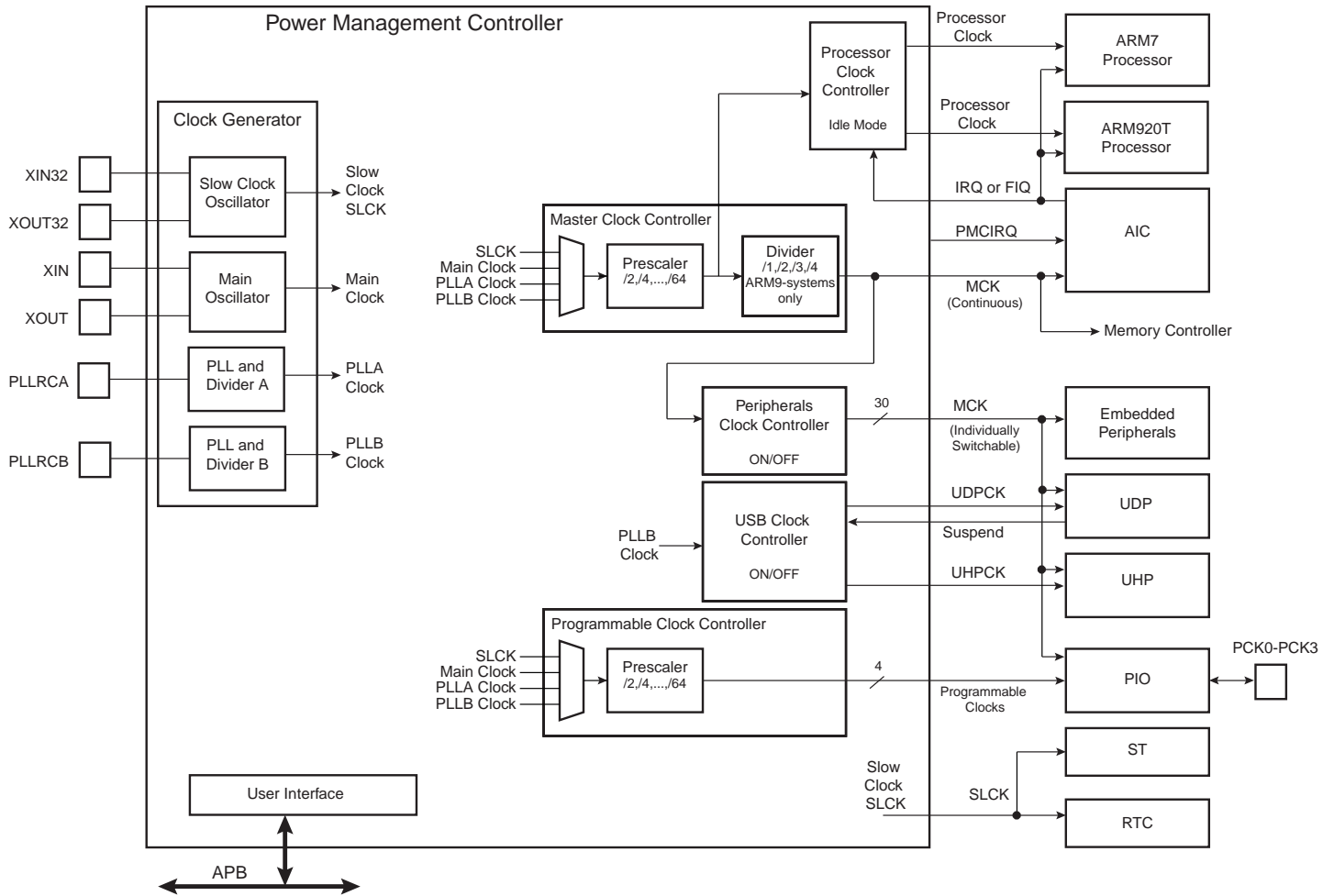
The Power Management Controller provides and controls two USB Clocks, the UHPCK for the USB Host Port, and the UDPCK for the USB Device.

If the product does not embed either the USB Host Port or the USB Device Port, the associated control bits and registers are not implemented in the PMC and programming them has no effect on the behavior of the PMC.



# Block Diagram

Figure 117. Power Management Controller Block Diagram



## Functional Description

### Operating Modes Definition

The following operating modes are supported by the PMC and offer different power consumption levels and event response latency times:

- Normal Mode: The ARM processor clock is enabled and peripheral clocks are enabled depending on application requirements.
- Idle Mode: The ARM processor clock is disabled and waiting for the next interrupt (or a main reset). The peripheral clocks are enabled depending on application requirements. PDC transfers are still possible.
- Slow Clock Mode: Slow clock mode is similar to normal mode, but the main oscillator and the PLL are switched off to save power and the processor and the peripherals run in Slow Clock mode. Note that slow clock mode is the mode selected after the reset.
- Standby Mode: Standby mode is a combination of Slow Clock mode and Idle Mode. It enables the processor to respond quickly to a wake-up event by keeping power consumption very low.

### Clock Definitions

The Power Management Controller provides the following clocks:

- Slow Clock (SLCK), typically at 32.768 kHz, is the only permanent clock within the system.
- Master Clock (MCK), programmable from a few hundred Hz to the maximum operating frequency of the device. It is available to the modules running permanently, such as the AIC and the Memory Controller.
- Processor Clock (PCK), typically the Master Clock for ARM7-based systems and a faster clock on ARM9-based systems, switched off when entering idle mode.
- Peripheral Clocks, typically MCK, provided to the embedded peripherals (USART, SSC, SPI, TWI, TC, MCI, etc.) and independently controllable. In order to reduce the number of clock names in a product, the Peripheral Clocks are named MCK in the product datasheet.
- UDP Clock (UDPCK), typically at 48 MHz, required by the USB Device Port operations.
- UHP Clock (UHPCK), typically at 48 MHz, required by the USB Host Port operations.
- Programmable Clock Outputs (PCK0 to PCK3) can be selected from the clocks provided by the clock generator and driven on the PCK0 to PCK3 pins.

### Clock Generator

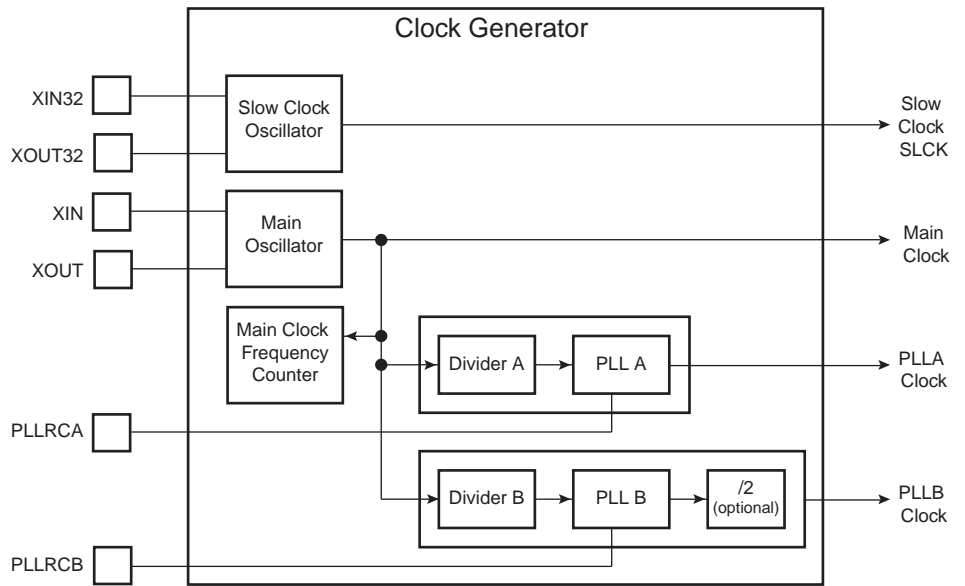
The Clock Generator embeds:

- the Slow Clock Oscillator
- the Main Oscillator
- two PLL and divider blocks, A and B

The Clock Generator may optionally integrate a divider by 2. The ARM7-based systems generally embed PLLs able to output between 20 MHz and 100 MHz and do not embed the divider by 2. The ARM9-based systems generally embed PLLs able to output between 80 MHz and 240 MHz. As the 48 MHz required by the USB cannot be reached by such a PLL, the optional divider by 2 is implemented.

The block diagram of the Clock Generator is shown in Figure 118.

Figure 118. Clock Generator Block Diagram

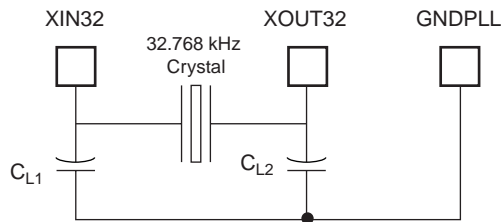


## Slow Clock Oscillator

### Slow Clock Oscillator Connection

The Clock Generator integrates a low-power 32.768 kHz oscillator. The XIN32 and XOUT32 pins must be connected to a 32.768 kHz crystal. Two external capacitors must be wired as shown in Figure 119.

Figure 119. Typical Slow Clock Oscillator Connection



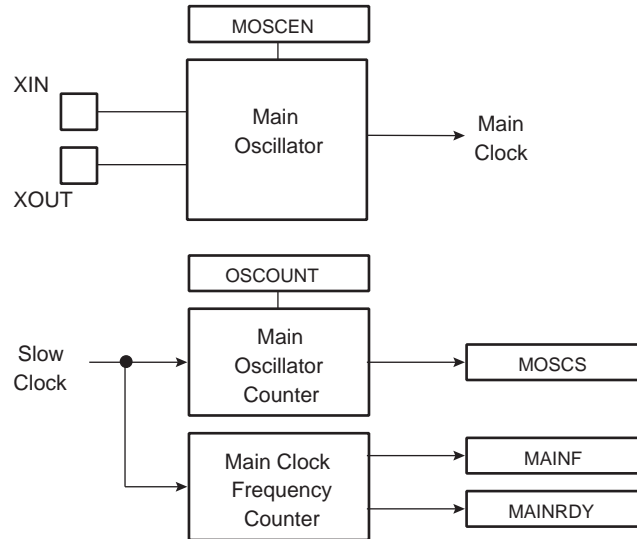
### Slow Clock Oscillator Startup Time

The startup time of the Slow Clock Oscillator is given in the DC Characteristics section of the product datasheet. As it is often higher than 500 ms and the processor requires an assertion of the reset until it has stabilized, the user must implement an external reset supervisor covering this startup time. However, this startup is only required in case of cold reset, i.e., in case of system power-up. When a warm reset occurs, the length of the reset pulse may be much lower. For further details, see “AT91RM9200 Reset Controller” on page 119.

## Main Oscillator

Figure 120 shows the Main Oscillator block diagram.

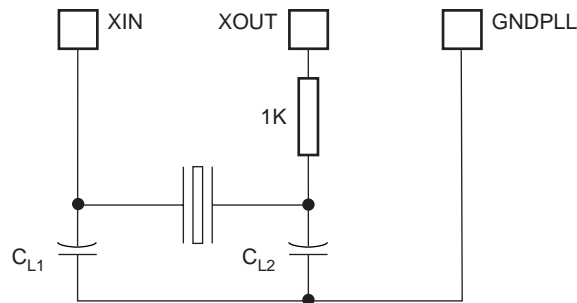
**Figure 120.** Main Oscillator Block Diagram



## Main Oscillator Connections

The Clock Generator integrates a Main Oscillator that is designed for a 3 to 20 MHz fundamental crystal. The typical crystal connection is illustrated in Figure 121. The 1 k $\Omega$  resistor is only required for crystals with frequencies lower than 8 MHz. The oscillator contains twenty-five pF capacitors on each XIN and XOUT pin. Consequently, CL1 and CL2 can be removed when a crystal with a load capacitance of 12.5 pF is used. For further details on the electrical characteristics of the Main Oscillator, see the DC Characteristics section of the product datasheet.

**Figure 121.** Typical Crystal Connection



## Main Oscillator Startup Time

The startup time of the Main Oscillator is given in the DC Characteristics section of the product datasheet. The startup time depends on the crystal frequency and increases when the frequency rises.

## Main Oscillator Control

To minimize the power required to start up the system, the Main Oscillator is disabled after reset and the Slow Clock mode is selected.

The software enables or disables the Main Oscillator so as to reduce power consumption by clearing the **MOSCEN** bit in the Main Oscillator Register (**CKGR\_MOR**). When disabling the Main Oscillator by clearing the **MOSCEN** bit in **CKGR\_MOR**, the **MOSCS** bit in **PMC\_SR** is automatically cleared indicating the Main Clock is off.

When enabling the Main Oscillator, the user must initiate the Main Oscillator counter with a value corresponding to the startup time of the oscillator. This startup time depends on the crystal frequency connected to the main oscillator. When the MOSCEN bit and the OSCOUNT are written in CKGR\_MOR to enable the Main Oscillator, the MOSCS bit is cleared and the counter starts counting down on the Slow Clock divided by 8 from the OSCOUNT value. Since the OSCOUNT value is coded with 8 bits, the maximum startup time is about 62 ms.

When the counter reaches 0, the MOSCS bit is set, indicating that the Main Clock is valid. Setting the MOSCS bit in PMC\_IMR can trigger an interrupt to the processor on this event.

### Main Clock Frequency Counter

The Main Oscillator features a Main Clock frequency counter that provides the quartz frequency connected to the Main Oscillator. Generally, this value is known by the system designer; however, it is useful for the boot program to configure the device with the correct clock speed, independently of the application.

The Main Clock frequency counter starts incrementing at the Main Clock speed after the next rising edge of the Slow Clock as soon as the Main Oscillator is stable, i.e., as soon as the MOSCS bit is set. Then, at the 16th falling edge of Slow Clock, the bit MAINRDY in CKGR\_MCFR (Main Clock Frequency Register) is set and the counter stops counting. Its value can be read in the MAINF field of CKGR\_MCFR and gives the number of Main Clock cycles during 16 periods of Slow Clock, so that the frequency of the crystal connected on the Main Oscillator can be determined.

### Main Oscillator Bypass

The user can input a clock on the device instead of connecting a crystal. In this case, the user has to provide the external clock signal on the pin XIN. The input characteristics of the XIN pin under these conditions are given in the product Electrical Characteristics section. The programmer has to be sure not to modify the MOSCEN bit in the Main Oscillator Register (CKGR\_MOR). This bit must remain at 0, its reset value, for the external clock to operate properly. While this bit is at 0, the pin XIN is tied low to prevent any internal oscillation regardless of pin connected.

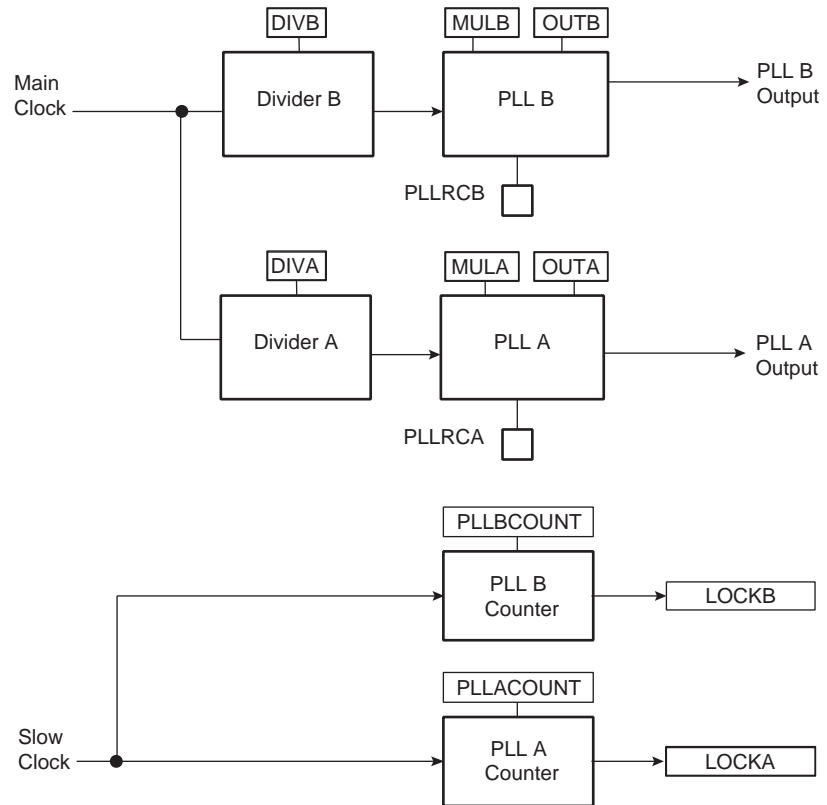
The external clock signal must meet the requirements relating to the power supply VDDPLL (i.e., between 1.65V and 1.95V) and cannot exceed 50 MHz.

## Divider and PLL Blocks

The Clock Generator features two Divider/PLL Blocks that generates a wide range of frequencies. Additionally, they provide a 48 MHz signal to the embedded USB device and/or host ports, regardless of the frequency of the Main Clock.

Figure 122 shows the block diagram of the divider and PLL blocks.

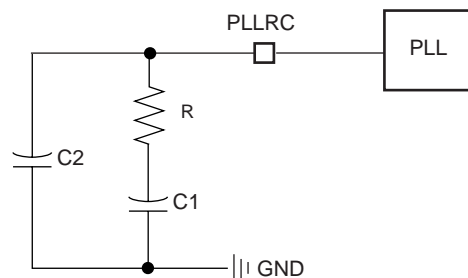
**Figure 122.** Divider and PLL Blocks Block Diagram



## PLL Filters

The two PLLs require connection to an external second-order filter through the pins PLLRC. Figure 123 shows a schematic of these filters.

**Figure 123.** PLL Capacitors and Resistors



Values of R, C1 and C2 to be connected to the PLLRC pins must be calculated as a function of the PLL input frequency, the PLL output frequency and the phase margin. A trade-off has to be found between output signal overshoot and startup time.

**PLL Source Clock**

The source of PLLs A and B is respectively the output of Divider A, i.e. the Main Clock divided by DIVA, and the output of Divider B, i.e. the Main Clock divided by DIVB.

As the input frequency of the PLLs is limited, the user has to make sure that the programming of DIVA and DIVB are compliant with the input frequency range of the PLLs, which is given in the DC Characteristics section of the product datasheet.

**Divider and Phase Lock Loop Programming**

The two dividers increase the accuracy of the PLLA and the PLLB clocks independently of the input frequency.

The Main Clock can be divided by programming the DIVB field in CKGR\_PLLBR and the DIVA field in CKGR\_PLLAR. Each divider can be set between 1 and 255 in steps of 1. When the DIVA and DIVB fields are set to 0, the output of the divider and the PLL outputs A and B are a continuous signal at level 0. On reset, the DIVA and DIVB fields are set to 0, thus both PLL input clocks are set to 0.

The two PLLs of the clock generator allow multiplication of the divider's outputs. The PLLA and the PLLB clock signals have a frequency that depends on the respective source signal frequency and on the parameters DIV (DIVA, DIVB) and MUL (MULA, MULB). The factor applied to the source signal frequency is  $(MUL + 1)/DIV$ . When MULA or MULB is written to 0, the corresponding PLL is disabled and its power consumption is saved. Re-enabling the PLLA or the PLLB can be performed by writing a value higher than 0 in the MULA or MULB field, respectively.

Whenever a PLL is re-enabled or one of its parameters is changed, the LOCKA or LOCKB bit in PMC\_SR is automatically cleared. The values written in the PLLACOUNT or PLLBCOUNT fields in CKGR\_PPLAR and CKGR\_PLLBR, respectively, are loaded in the corresponding PLL counter. The PLL counter then decrements at the speed of the Slow Clock until it reaches 0. At this time, the corresponding LOCK bit is set in PMC\_SR and can trigger an interrupt to the processor. The user has to load the number of Slow Clock cycles required to cover the PLL transient time into the PLLACOUNT and PLLBCOUNT field. The transient time depends on the PLL filters. The initial state of the PLL and its target frequency can be calculated using a specific tool provided by Atmel.

**PLLB Divider by 2**

In ARM9-based systems, the PLLB clock may be divided by two. This divider can be enabled by setting the bit USB\_96M of CKGR\_PLLBR. In this case, the divider by 2 is enabled and the PLLB must be programmed to output 96 MHz and not 48 MHz, thus ensuring correct operation of the USB bus.

**Clock Controllers**

The Power Management Controller provides the clocks to the different peripherals of the system, either internal or external. It embeds the following elements:

- the Master Clock Controller, which selects the Master Clock.
- the Processor Clock Controller, which implements the Idle Mode.
- the Peripheral Clock Controller, which provides power saving by controlling clocks of the embedded peripherals.
- the USB Clock Controller, which distributes the 48 MHz clock to the USB controllers.
- the Programmable Clock Controller, which allows generation of up to four programmable clock signals on external pins.

**Master Clock Controller**

The Master Clock Controller provides selection and division of the Master Clock (MCK). MCK is the clock provided to all the peripherals and the memory controller.

The Master Clock is selected from one of the clocks provided by the Clock Generator. Selecting the Slow Clock enables Slow Clock Mode by providing a 32.768 kHz signal to the whole device. Selecting the Main Clock saves power consumption of both PLLs, but



prevents using the USB ports. Selecting the PLLB Clock saves the power consumption of the PLLA by running the processor and the peripheral at 48 MHz required by the USB ports. Selecting the PLLA Clock runs the processor and the peripherals at their maximum speed while running the USB ports at 48 MHz.

The Master Clock Controller is made up of a clock selector and a prescaler, as shown in Figure 124. It also contains an optional Master Clock divider in products integrating an ARM9 processor. This allows the processor clock to be faster than the Master Clock.

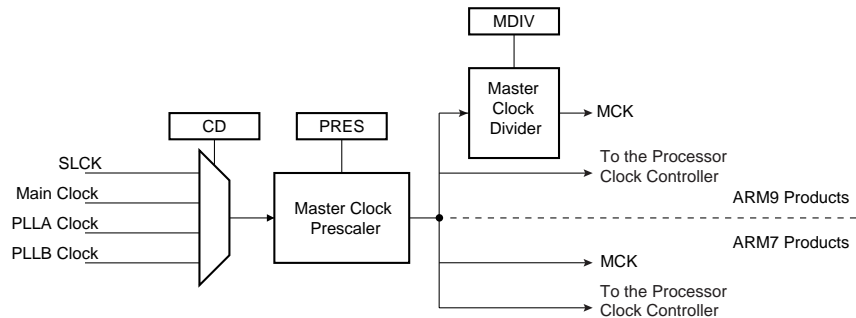
The Master Clock selection is made by writing the CSS field (Clock Source Selection) in PMC\_MCKR (Master Clock Register). The prescaler supports the division by a power of 2 of the selected clock between 1 and 64. The PRES field in PMC\_MCKR programs the prescaler.

When the Master Clock divider is implemented, it can be programmed between 1 and 4 through the MDIV field in PMC\_MCKR.

Each time PMC\_MCKR is written to define a new Master Clock, the MCKRDY bit is cleared in PMC\_SR. It reads 0 until the Master Clock is established. Then, the MCKRDY bit is set and can trigger an interrupt to the processor. This feature is useful when switching from a high-speed clock to a lower one to inform the software when the change is actually done.

Note: A new value to be written in PMC\_MCKR must not be the same as the current value in PMC\_MCKR.

**Figure 124.** Master Clock Controller



**Processor Clock Controller**

The PMC features a Processor Clock Controller that implements the Idle Mode. The Processor Clock 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).

*Processor Clock Source*

The clock provided to the processor is determined by the Master Clock controller. On ARM7-based systems, the Processor Clock source is directly the Master Clock.

On ARM9-based systems, the Processor Clock source might be 2, 3 or 4 times the Master Clock. This ratio value is determined by programming the field MDIV of the Master Clock Register (PMC\_MCKR).

*Idle Mode*

The Processor Clock is enabled after a reset and is automatically re-enabled by any enabled interrupt. The Idle Mode is achieved by disabling the Processor Clock, which is automatically re-enabled by any enabled fast or normal interrupt, or by the reset of the product.



When the Processor Clock is disabled, the current instruction is finished before the clock is stopped, but this does not prevent data transfers from other masters of the system bus.

## Peripheral Clock Controller

The PMC controls the clocks of each embedded peripheral. The user can individually enable and disable the Master Clock on the peripherals 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 wait 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.

The bit number within the Peripheral Clock Control registers (PMC\_PCER, PMC\_PCDR, and PMC\_PCSR) is the Peripheral Identifier defined at the product level. Generally, the bit number corresponds to the interrupt source number assigned to the peripheral.

## USB Clock Controller

If using one of the USB ports, the user has to program the Divider and PLL B block to output a 48 MHz signal with an accuracy of  $\pm 0.25\%$ .

When the clock for the USB is stable, the USB device and host clocks, UDPCK and UHPCK, can be enabled. They can be disabled when the USB transactions are finished, so that the power consumption generated by the 48 MHz signal on these peripherals is saved.

The USB ports require both the 48 MHz signal and the Master Clock. The Master Clock may be controlled via the Peripheral Clock Controller.

### *USB Device Clock Control*

The USB Device Port clock UDPCK can be enabled by writing 1 at the UDP bit in PMC\_SCER (System Clock Enable Register) and disabled by writing 1 at the bit UDP in PMC\_SCDR (System Clock Disable Register). The activity of UDPCK is shown in the bit UDP of PMC\_SCSR (System Clock Status Register).

### *USB Device Port Suspend*

When the USB Device Port detects a suspend condition, the 48 MHz clock is automatically disabled, i.e., the UDP bit in PMC\_SCSR is cleared. It is also possible to automatically disable the Master Clock provided to the USB Device Port on a suspend condition. The MCKUDP bit in PMC\_SCSR configures this feature and can be set or cleared by writing one in the same bit of PMC\_SCER and PMC\_SCDR.

### *USB Host Clock Control*

The USB Host Port clock UHPCK can be enabled by writing 1 at the UHP bit in PMC\_SCER (System Clock Enable Register) and disabled by writing 1 at the UHP bit in PMC\_SCDR (System Clock Disable Register). The activity of UDPCK is shown in the bit UHP of PMC\_SCSR (System Clock Status Register).

## Programmable Clock Output Controller

The PMC controls up to four signals to be output on external pins PCK0 to PCK3. Each signal can be independently programmed via the registers PMC\_PCK0 to PMC\_PCK3.

PCK0 to PCK3 can be independently selected between the four clocks provided by the Clock Generator by writing the CSS field in PMC\_PCK0 to PMC\_PCK3. Each output signal can also be divided by a power of 2 between 1 and 64 by writing the field PRES (Prescaler) in PMC\_PCK0 to PMC\_PCK3.



Each output signal can be enabled and disabled by writing 1 in the corresponding bit PCK0 to PCK3 of PMC\_SCER and PMC\_SCDR, respectively. Status of the active programmable output clocks are given in the bits PCK0 to PCK3 of PMC\_SCSR (System Clock Status Register).

Moreover, like the MCK, a status bit in PMC\_SR indicates that the Programmable Clock is actually what has been programmed in the Programmable Clock registers.

As the Programmable Clock Controller does not manage with glitch prevention when switching clocks, it is strongly recommended to disable the Programmable Clock before any configuration change and to re-enable it after the change is actually performed.

Note also that it is required to assign the pin to the Programmable Clock operation in the PIO Controller to enable the signal to be driven on the pin.

## Clock Switching Details

### Master Clock Switching Timings

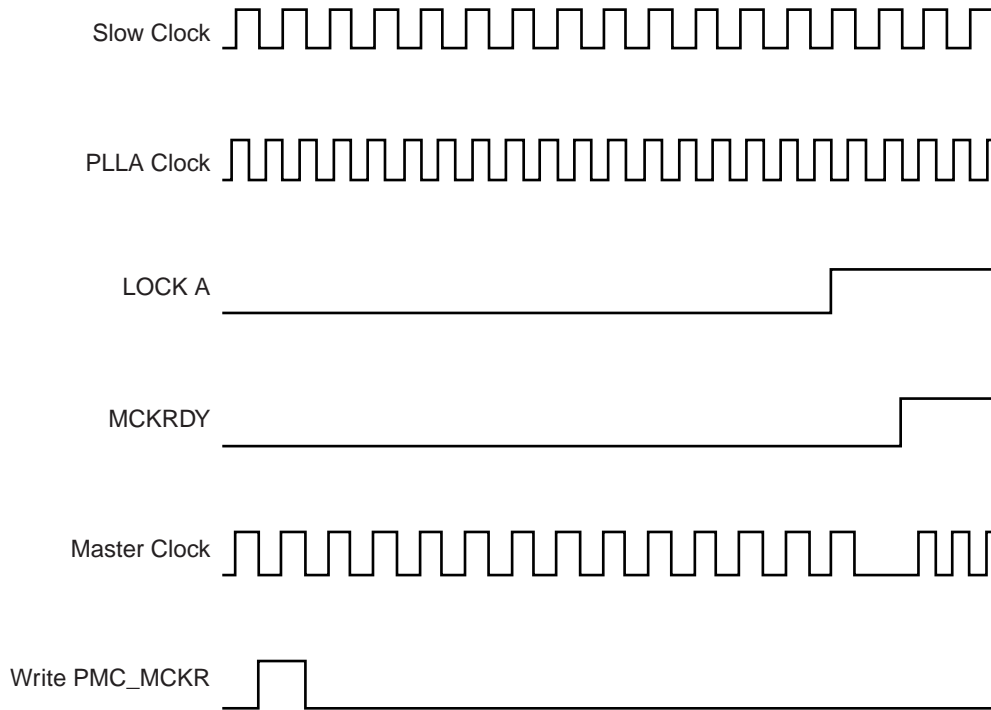
Table 60 gives the worst case timing required for the Master Clock to switch from one selected clock to another one. This is in the event that the prescaler is de-activated. When the prescaler is activated, an additional time of 64 clock cycles of the new selected clock has to be added.

**Table 60.** Clock Switching Timings (Worst Case)

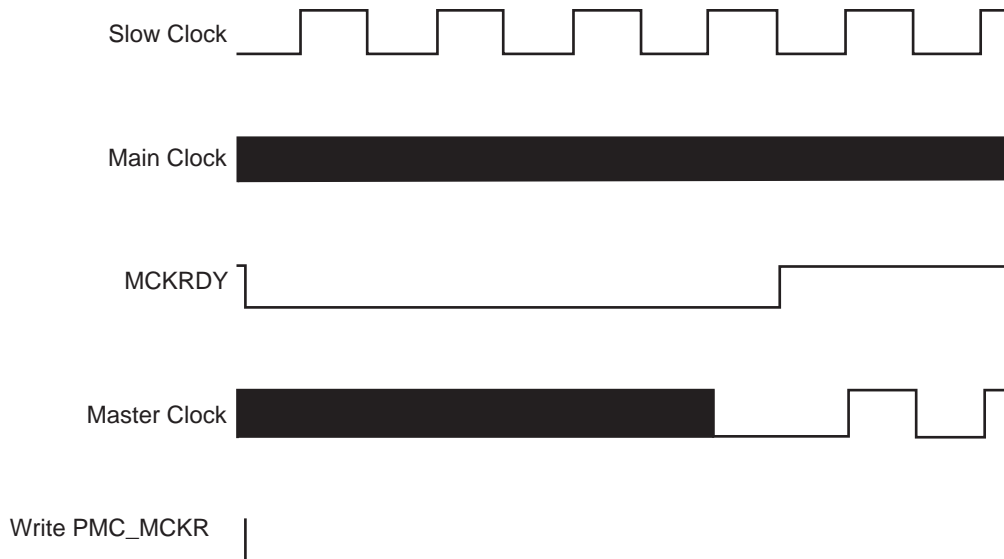
| To         | From | Main Clock   | SLCK   | PLLA Clock   | PLLB Clock   |
|------------|------|--|--|--|--|
| Main Clock |      | –  | 4 x SLCK +<br>2.5 x Main Clock                       | 3 x PLLA Clock +<br>4 x SLCK +<br>1 x Main Clock       | 3 x PLLB Clock +<br>4 x SLCK +<br>1 x Main Clock     |
| SLCK       |      | 0.5 x Main Clock +<br>4.5 x SLCK   | –  | 3 x PLLA Clock +<br>5 x SLCK                           | 3 x PLLB Clock +<br>5 x SLCK                         |
| PLLA Clock |      | 0.5 x Main Clock +<br>4 x SLCK +<br>PLLACOUNT x SLCK +<br>2.5 x PLLA Clock | 2.5 x PLLA Clock +<br>5 x SLCK +<br>PLLACOUNT x SLCK | 2.5 x PLLA Clock +<br>4 x SLCK +<br>PLL B COUNT x SLCK | 3 x PLLA Clock +<br>4 x SLCK +<br>1.5 x PLLA Clock   |
| PLLB Clock |      | 0.5 x Main Clock +<br>4 x SLCK +<br>PLLBCOUNT x SLCK +<br>2.5 x PLLB Clock | 2.5 x PLLB Clock +<br>5 x SLCK +<br>PLLBCOUNT x SLCK | 3 x PLLB Clock +<br>4 x SLCK +<br>1.5 x PLLB Clock     | 2.5 x PLLB Clock +<br>4 x SLCK +<br>PLLACOUNT x SLCK |

## Clock Switching Waveforms

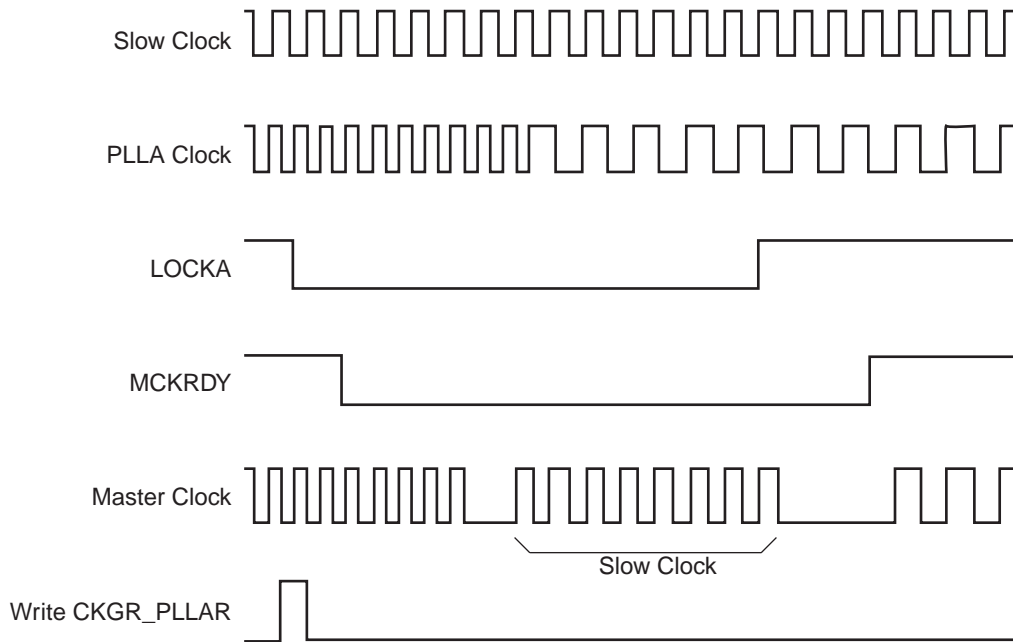
**Figure 125.** Switch Master Clock from Slow Clock to PLLA Clock



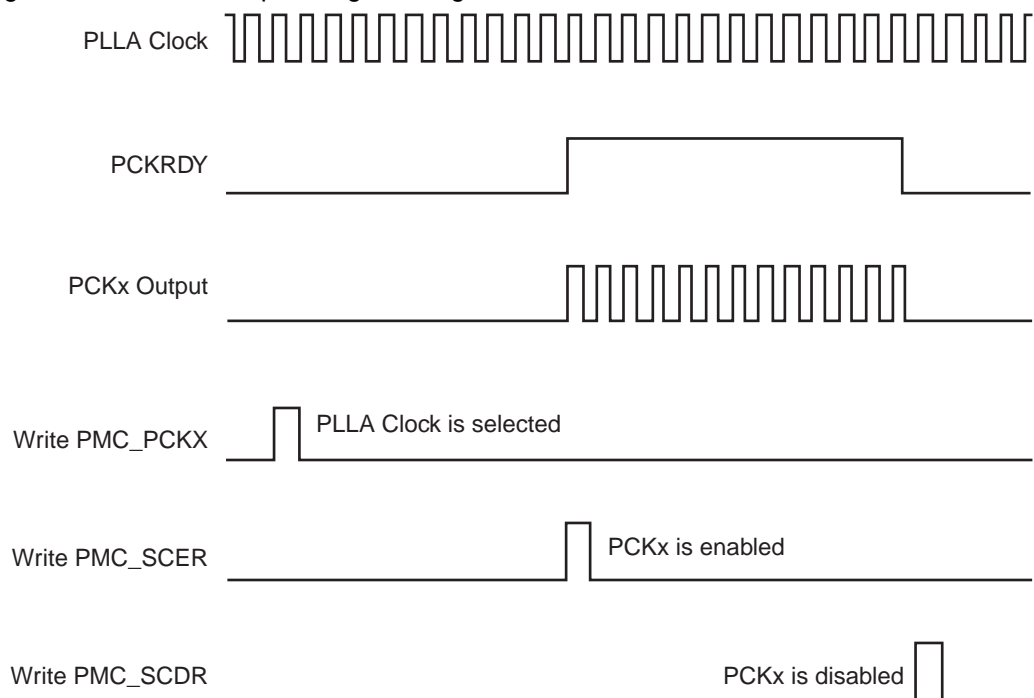
**Figure 126.** Switch Master Clock from Main Clock to Slow Clock



**Figure 127.** Change PLLA Programming



**Figure 128.** Programmable Clock Output Programming



## Power Management Controller (PMC) User Interface

**Table 61.** Register Mapping

| Offset | Register                          | Name       | Access     | Reset Value |
|--------|-----------------------------------|------------|------------|-------------|
| 0x0000 | System Clock Enable Register      | PMC_SCER   | Write-only | –           |
| 0x0004 | System Clock Disable Register     | PMC_SCDR   | Write-only | –           |
| 0x0008 | System Clock Status Register      | PMC_SCSR   | Read-only  | 0x01        |
| 0x000C | Reserved                          | –          | –          | –           |
| 0x0010 | Peripheral Clock Enable Register  | PMC_PCER   | Write-only | –           |
| 0x0014 | Peripheral Clock Disable Register | PMC_PCDR   | Write-only | –           |
| 0x0018 | Peripheral Clock Status Register  | PMC_PCSR   | Read-only  | 0x0         |
| 0x001C | Reserved                          | –          | –          | –           |
| 0x0020 | Main Oscillator Register          | CKGR_MOR   | ReadWrite  | 0x0         |
| 0x0024 | Main Clock Frequency Register     | CKGR_MCFR  | Read-only  | -           |
| 0x0028 | PLL A Register                    | CKGR_PLLAR | ReadWrite  | 0x3F00      |
| 0x002C | PLL B Register                    | CKGR_PLLBR | ReadWrite  | 0x3F00      |
| 0x0030 | Master Clock Register             | PMC_MCKR   | Read/Write | 0x00        |
| 0x0034 | Reserved                          | –          | –          | –           |
| 0x0038 | Reserved                          | –          | –          | –           |
| 0x003C | Reserved                          | –          | –          | –           |
| 0x0040 | Programmable Clock 0 Register     | PMC_PCK0   | Read/Write | 0x0         |
| 0x0044 | Programmable Clock 1 Register     | PMC_PCK1   | Read/Write | 0x0         |
| 0x0048 | Programmable Clock 2 Register     | PMC_PCK2   | Read/Write | 0x0         |
| 0x004C | Programmable Clock 3 Register     | PMC_PCK3   | Read/Write | 0x0         |
| 0x0050 | Reserved                          | –          | –          | –           |
| 0x0054 | Reserved                          | –          | –          | –           |
| 0x0058 | Reserved                          | –          | –          | –           |
| 0x005C | Reserved                          | –          | –          | –           |
| 0x0060 | Interrupt Enable Register         | PMC_IER    | Write-only | --          |
| 0x0064 | Interrupt Disable Register        | PMC_IDR    | Write-only | --          |
| 0x0068 | Status Register                   | PMC_SR     | Read-only  | --          |
| 0x006C | Interrupt Mask Register           | PMC_IMR    | Read-only  | 0x0         |

**PMC System Clock Enable Register**

Register Name: PMC\_SCER

Access Type: Write-only

|    |    |    |     |      |        |      |      |
|----|----|----|-----|------|--------|------|------|
| 31 | 30 | 29 | 28  | 27   | 26     | 25   | 24   |
| –  | –  | –  | –   | –    | –      | –    | –    |
| 23 | 22 | 21 | 20  | 19   | 18     | 17   | 16   |
| –  | –  | –  | –   | –    | –      | –    | –    |
| 15 | 14 | 13 | 12  | 11   | 10     | 9    | 8    |
| –  | –  | –  | –   | PCK3 | PCK2   | PCK1 | PCK0 |
| 7  | 6  | 5  | 4   | 3    | 2      | 1    | 0    |
| –  | –  | –  | UHP | –    | MCKUDP | UDP  | PCK  |

- **PCK: Processor Clock Enable**

0 = No effect.

1 = Enables the Processor Clock.

- **UDP: USB Device Port Clock Enable**

0 = No effect.

1 = Enables the 48 MHz clock of the USB Device Port.

- **MCKUDP: USB Device Port Master Clock Automatic Disable on Suspend Enable**

0 = No effect.

1 = Enables the automatic disable of the Master Clock of the USB Device Port when a suspend condition occurs.

- **UHP: USB Host Port Clock Enable**

0 = No effect.

1 = Enables the 48 MHz clock of the USB Host Port.

- **PCK0...PCK3: Programmable Clock Output Enable**

0 = No effect.

1 = Enables the corresponding Programmable Clock output.

## PMC System Clock Disable Register

Register Name: PMC\_SCDR

Access Type: Write-only

|    |    |    |     |      |        |      |      |
|----|----|----|-----|------|--------|------|------|
| 31 | 30 | 29 | 28  | 27   | 26     | 25   | 24   |
| –  | –  | –  | –   | –    | –      | –    | –    |
| 23 | 22 | 21 | 20  | 19   | 18     | 17   | 16   |
| –  | –  | –  | –   | –    | –      | –    | –    |
| 15 | 14 | 13 | 12  | 11   | 10     | 9    | 8    |
| –  | –  | –  | –   | PCK3 | PCK2   | PCK1 | PCK0 |
| 7  | 6  | 5  | 4   | 3    | 2      | 1    | 0    |
| –  | –  | –  | UHP | –    | MCKUDP | UDP  | PCK  |

- **PCK: Processor Clock Disable**

0 = No effect.

1 = Disables the Processor Clock.

- **UDP: USB Device Port Clock Disable**

0 = No effect.

1 = Disables the 48 MHz clock of the USB Device Port.

- **MCKUDP: USB Device Port Master Clock Automatic Disable on Suspend Disable**

0 = No effect.

1 = Disables the automatic disable of the Master Clock of the USB Device Port when a suspend condition occurs.

- **UHP: USB Host Port Clock Disable**

0 = No effect.

1 = Disables the 48 MHz clock of the USB Host Port.

- **PCK0...PCK3: Programmable Clock Output Disable**

0 = No effect.

1 = Disables the corresponding Programmable Clock output.



**PMC System Clock Status Register**

Register Name: PMC\_SCSR

Access Type: Read-only

|    |    |    |     |      |        |      |      |
|----|----|----|-----|------|--------|------|------|
| 31 | 30 | 29 | 28  | 27   | 26     | 25   | 24   |
| –  | –  | –  | –   | –    | –      | –    | –    |
| 23 | 22 | 21 | 20  | 19   | 18     | 17   | 16   |
| –  | –  | –  | –   | –    | –      | –    | –    |
| 15 | 14 | 13 | 12  | 11   | 10     | 9    | 8    |
| –  | –  | –  | –   | PCK3 | PCK2   | PCK1 | PCK0 |
| 7  | 6  | 5  | 4   | 3    | 2      | 1    | 0    |
| –  | –  | –  | UHP | –    | MCKUDP | UDP  | PCK  |

• **PCK: Processor Clock Status**

0 = The Processor Clock is disabled.

1 = The Processor Clock is enabled.

• **UDP: USB Device Port Clock Status**

0 = The 48 MHz clock of the USB Device Port is disabled.

1 = The 48 MHz clock of the USB Device Port is enabled.

• **MCKUDP: USB Device Port Master Clock Automatic Disable on Suspend Status**

0 = The automatic disable of the Master clock of the USB Device Port when suspend condition occurs is disabled.

1 = The automatic disable of the Master clock of the USB Device Port when suspend condition occurs is enabled.

• **UHP: USB Host Port Clock Status**

0 = The 48 MHz clock of the USB Host Port is disabled.

1 = The 48 MHz clock of the USB Host Port is enabled.

• **PCK0...PCK3: Programmable Clock Output Status**

0 = The corresponding Programmable Clock output is disabled.

1 = The corresponding Programmable Clock output is enabled.



## PMC Peripheral Clock Enable Register

Register Name: PMC\_PCER

Access Type: Write-only

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 31    | 30    | 29    | 28    | 27    | 26    | 25    | 24    |
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23    | 22    | 21    | 20    | 19    | 18    | 17    | 16    |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9  | PID8  |
| 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
| PID7  | PID6  | PID5  | PID4  | PID3  | PID2  | –     | –     |

- **PID2...PID31: Peripheral Clock Enable**

0 = No effect.

1 = Enables the corresponding peripheral clock.

## PMC Peripheral Clock Disable Register

Register Name: PMC\_PCDR

Access Type: Write-only

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 31    | 30    | 29    | 28    | 27    | 26    | 25    | 24    |
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23    | 22    | 21    | 20    | 19    | 18    | 17    | 16    |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9  | PID8  |
| 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
| PID7  | PID6  | PID5  | PID4  | PID3  | PID2  | –     | –     |

- **PID2...PID31: Peripheral Clock Disable**

0 = No effect.

1 = Disables the corresponding peripheral clock.

**PMC Peripheral Clock Status Register**

Register Name: PMC\_PCSR

Access Type: Read-only

|       |       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 31    | 30    | 29    | 28    | 27    | 26    | 25    | 24    |
| PID31 | PID30 | PID29 | PID28 | PID27 | PID26 | PID25 | PID24 |
| 23    | 22    | 21    | 20    | 19    | 18    | 17    | 16    |
| PID23 | PID22 | PID21 | PID20 | PID19 | PID18 | PID17 | PID16 |
| 15    | 14    | 13    | 12    | 11    | 10    | 9     | 8     |
| PID15 | PID14 | PID13 | PID12 | PID11 | PID10 | PID9  | PID8  |
| 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
| PID7  | PID6  | PID5  | PID4  | PID3  | PID2  | –     | –     |

• **PID2...PID31: Peripheral Clock Status**

0 = The corresponding peripheral clock is disabled.

1 = The corresponding peripheral clock is enabled.



## PMC Clock Generator Main Oscillator Register

Register Name: CKGR\_MOR

Access Type: Read/Write

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

- **MOSCEN: Main Oscillator Enable**

0 = The Main Oscillator is disabled. Main Clock is the signal connected on XIN.

1 = The Main Oscillator is enabled. A crystal must be connected between XIN and XOUT.

- **OSCOUNT: Main Oscillator Start-up Time**

Specifies the number of Slow Clock cycles for the Main Oscillator start-up time.

**PMC Clock Generator Main Clock Frequency Register**

Register Name: CKGR\_MCFR

Access Type: Read-only

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

- **MAINF: Main Clock Frequency**

Gives the number of Main Clock cycles within 16 Slow Clock periods.

- **MAINRDY: Main Clock Ready**

0 = MAINF value is not valid or the Main Oscillator is disabled.

1 = The Main Oscillator has been enabled previously and MAINF value is available.



## PMC Clock Generator PLL A Register

Register Name: CKGR\_PLLAR

Access Type: Read/Write

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

Possible limitations on PLL A input frequencies and multiplier factors should be checked before using the Clock Generator.

- **DIVA: Divider A**

| DIVA    | Divider Selected                                  |
|---------|---|
| 0       | Divider output is 0                               |
| 1       | Divider is bypassed                               |
| 2 - 255 | Divider output is the Main Clock divided by DIVA. |

- **PLLACOUNT: PLL A Counter**

Specifies the number of Slow Clock cycles before the LOCKA bit is set in PMC\_SR after CKGR\_PLLAR is written.

- **OUTA: PLL A Clock Frequency Range**

| OUTA |   | PLL A Frequency Output Range |
|------|---|------------------------------|
| 0    | 0 | 80 MHz to 160 MHz            |
| 0    | 1 | Reserved                     |
| 1    | 0 | 150 MHz to 240 MHz           |
| 1    | 1 | Reserved                     |

- **MULA: PLL A Multiplier**

0 = The PLL A is deactivated.

1 up to 2047 = The PLL A Clock frequency is the PLL A input frequency multiplied by MULA + 1.

## PMC Clock Generator PLL B Register

Register Name: CKGR\_PLLBR

Access Type: Read/Write

|      |    |           |         |    |      |    |    |
|------|----|-----------|---------|----|------|----|----|
| 31   | 30 | 29        | 28      | 27 | 26   | 25 | 24 |
| –    | –  | –         | USB_96M | –  | MULB |    |    |
| 23   | 22 | 21        | 20      | 19 | 18   | 17 | 16 |
| MULB |    |           |         |    |      |    |    |
| 15   | 14 | 13        | 12      | 11 | 10   | 9  | 8  |
| OUTB |    | PLLBCOUNT |         |    |      |    |    |
| 7    | 6  | 5         | 4       | 3  | 2    | 1  | 0  |
| DIVB |    |           |         |    |      |    |    |

- DIVB: Divider B**

| DIVB    | Divider Selected                                      |
|---------|---|
| 0       | Divider output is 0                                   |
| 1       | Divider is bypassed                                   |
| 2 - 255 | Divider output is the selected clock divided by DIVB. |

- PLLBCOUNT: PLL B Counter**

Specifies the number of slow clock cycles before the LOCKB bit is set in PMC\_SR after CKGR\_PLLBR is written.

- OUTB: PLL B Clock Frequency Range**

| OUTB |   | PLL B Clock Frequency Range |
|------|---|-----------------------------|
| 0    | 0 | 80 MHz to 160 MHz           |
| 0    | 1 | Reserved                    |
| 1    | 0 | 150 MHz to 240 MHz          |
| 1    | 1 | Reserved                    |

- MULB: PLL B Multiplier**

0 = The PLL B is deactivated.

1 up to 2047 = The PLL B Clock frequency is the PLL B input frequency multiplied by MULB + 1.

- USB\_96M: Divider by 2 Enable (only on ARM9-based Systems)**

0 = USB ports clocks are PLL B Clock, therefore the PMC Clock Generator must be programmed for the PLL B Clock to be 48 MHz.

1 = USB ports clocks are PLL B Clock divided by 2, therefore the PMC Clock Generator must be programmed for the PLL B Clock to be 96 MHz.

## PMC Master Clock Register

Register Name: PMC\_MCKR

Access Type: Read/Write

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

Note: Value to be written in PMC\_MCKR must not be the same as current value in PMC\_MCKR.

- **CSS: Master Clock Selection**

| CSS |   | Clock Source Selection  |
|-----|---|-------------------------|
| 0   | 0 | Slow Clock is selected  |
| 0   | 1 | Main Clock is selected  |
| 1   | 0 | PLL A Clock is selected |
| 1   | 1 | PLL B Clock is selected |

- **PRES: Master Clock Prescaler**

| PRES |   |   | Master Clock                 |
|------|---|---|------------------------------|
| 0    | 0 | 0 | Selected clock               |
| 0    | 0 | 1 | Selected clock divided by 2  |
| 0    | 1 | 0 | Selected clock divided by 4  |
| 0    | 1 | 1 | Selected clock divided by 8  |
| 1    | 0 | 0 | Selected clock divided by 16 |
| 1    | 0 | 1 | Selected clock divided by 32 |
| 1    | 1 | 0 | Selected clock divided by 64 |
| 1    | 1 | 1 | Reserved                     |

- **MDIV: Master Clock Division (on ARM9-based systems only)**

0 = The Master Clock and the Processor Clock are the same.

1 = The Processor Clock is twice as fast as the Master Clock.

2 = The Processor Clock is three times faster than the Master Clock.

3 = The Processor Clock is four times faster than the Master Clock.



**PMC Programmable Clock Register 0 to 3**

Register Name: PMC\_PCK0..PMC\_PCK3

Access Type: Read/Write

|    |    |    |      |    |    |     |    |
|----|----|----|------|----|----|-----|----|
| 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  |
| –  | –  | –  | PRES |    |    | CSS |    |

• **CSS: Master Clock Selection**

| CSS |   | Clock Source Selection |                         |
|-----|---|------------------------|-------------------------|
| 0   | 0 | 0                      | Slow Clock is selected  |
| 0   | 1 | 1                      | Main Clock is selected  |
| 1   | 0 | 0                      | PLL A Clock is selected |
| 1   | 1 | 1                      | PLL B Clock is selected |

• **PRES: Programmable Clock Prescaler**

| PRES |   |   | Master Clock                 |
|------|---|---|------------------------------|
| 0    | 0 | 0 | Selected clock               |
| 0    | 0 | 1 | Selected clock divided by 2  |
| 0    | 1 | 0 | Selected clock divided by 4  |
| 0    | 1 | 1 | Selected clock divided by 8  |
| 1    | 0 | 0 | Selected clock divided by 16 |
| 1    | 0 | 1 | Selected clock divided by 32 |
| 1    | 1 | 0 | Selected clock divided by 64 |
| 1    | 1 | 1 | Reserved                     |

## PMC Interrupt Enable Register

Register Name: PMC\_IER

Access Type: Write-only

|    |    |    |    |         |         |         |         |
|----|----|----|----|---------|---------|---------|---------|
| 31 | 30 | 29 | 28 | 27      | 26      | 25      | 24      |
| –  | –  | –  | –  | –       | –       | –       | –       |
| 23 | 22 | 21 | 20 | 19      | 18      | 17      | 16      |
| –  | –  | –  | –  | –       | –       | –       | –       |
| 15 | 14 | 13 | 12 | 11      | 10      | 9       | 8       |
| –  | –  | –  | –  | PCK3RDY | PCK2RDY | PCK1RDY | PCK0RDY |
| 7  | 6  | 5  | 4  | 3       | 2       | 1       | 0       |
| –  | –  | –  | –  | MCKRDY  | LOCKB   | LOCKA   | MOSCS   |

- **MOSCS: Main Oscillator Status**
- **LOCKA: PLL A Lock**
- **LOCKB: PLL B Lock**
- **MCKRDY: Master Clock Ready**
- **PCK0RDY - PCK3RDY: Programmable Clock Ready**

0 = No effect.

1 = Enables the corresponding interrupt.

## PMC Interrupt Disable Register

Register Name: PMC\_IDR

Access Type: Write-only

|    |    |    |    |         |         |         |         |
|----|----|----|----|---------|---------|---------|---------|
| 31 | 30 | 29 | 28 | 27      | 26      | 25      | 24      |
| –  | –  | –  | –  | –       | –       | –       | –       |
| 23 | 22 | 21 | 20 | 19      | 18      | 17      | 16      |
| –  | –  | –  | –  | –       | –       | –       | –       |
| 15 | 14 | 13 | 12 | 11      | 10      | 9       | 8       |
| –  | –  | –  | –  | PCK3RDY | PCK2RDY | PCK1RDY | PCK0RDY |
| 7  | 6  | 5  | 4  | 3       | 2       | 1       | 0       |
| –  | –  | –  | –  | MCKRDY  | LOCKB   | LOCKA   | MOSCS   |

- **MOSCS: Main Oscillator Status**
- **LOCKA: PLL A Lock**
- **LOCKB: PLL B Lock**
- **MCKRDY: Master Clock Ready**
- **PCK0RDY - PCK3RDY: Programmable Clock Ready**

0 = No effect.

1 = Disables the corresponding interrupt.

**PMC Status Register**

Register Name: PMC\_SR

Access Type: Read-only

|    |    |    |    |         |         |         |         |
|----|----|----|----|---------|---------|---------|---------|
| 31 | 30 | 29 | 28 | 27      | 26      | 25      | 24      |
| –  | –  | –  | –  | –       | –       | –       | –       |
| 23 | 22 | 21 | 20 | 19      | 18      | 17      | 16      |
| –  | –  | –  | –  | –       | –       | –       | –       |
| 15 | 14 | 13 | 12 | 11      | 10      | 9       | 8       |
| –  | –  | –  | –  | PCK3RDY | PCK2RDY | PCK1RDY | PCK0RDY |
| 7  | 6  | 5  | 4  | 3       | 2       | 1       | 0       |
| –  | –  | –  | –  | MCKRDY  | LOCKB   | LOCKA   | MOSCS   |

• **MOSCS: MOSCS Flag Status**

0 = Main oscillator is not stabilized.

1 = Main oscillator is stabilized.

• **LOCKA: PLLA Lock Status**

0 = PLL A is not locked

1 = PLL A is locked.

• **LOCKB: PLLB Lock Status**

0 = PLL B is not locked.

1 = PLL B is locked.

• **MCKRDY: Master Clock Status**

0 = MCK is not ready.

1 = MCK is ready.

• **PCK0RDY - PCK3RDY: Programmable Clock Ready Status**

0 = Programmable Clock 0 to 3 is not ready.

1 = Programmable Clock 0 to 3 is ready.

## PMC Interrupt Mask Register

Register Name: PMC\_IMR

Access Type: Read-only

|    |    |    |    |         |         |         |         |
|----|----|----|----|---------|---------|---------|---------|
| 31 | 30 | 29 | 28 | 27      | 26      | 25      | 24      |
| –  | –  | –  | –  | –       | –       | –       | –       |
| 23 | 22 | 21 | 20 | 19      | 18      | 17      | 16      |
| –  | –  | –  | –  | –       | –       | –       | –       |
| 15 | 14 | 13 | 12 | 11      | 10      | 9       | 8       |
| –  | –  | –  | –  | PCK3RDY | PCK2RDY | PCK1RDY | PCK0RDY |
| 7  | 6  | 5  | 4  | 3       | 2       | 1       | 0       |
| –  | –  | –  | –  | MCKRDY  | LOCKB   | LOCKA   | MOSCS   |

- **MOSCS: Main Oscillator Status**
- **LOCKA: PLL A Lock**
- **LOCKB: PLL B Lock**
- **MCKRDY: Master Clock Ready**
- **PCK0RDY - PCK3RDY: Programmable Clock Ready**
- **MOSCS: MOSCS Interrupt Mask**

0 = The corresponding interrupt is enabled.

1 = The corresponding interrupt is disabled.

## System Timer (ST)

### Overview

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

- A Period Interval Timer (PIT) that sets the time base for an operating system.
- A Watchdog Timer (WDT) with system reset capabilities in case of software deadlock.
- A Real-Time Timer (RTT) counting elapsed seconds.

These timers count using the Slow Clock provided by the Power Management Controller. Typically, this clock has a frequency of 32.768 kHz, but the System Timer might be configured to support another frequency.

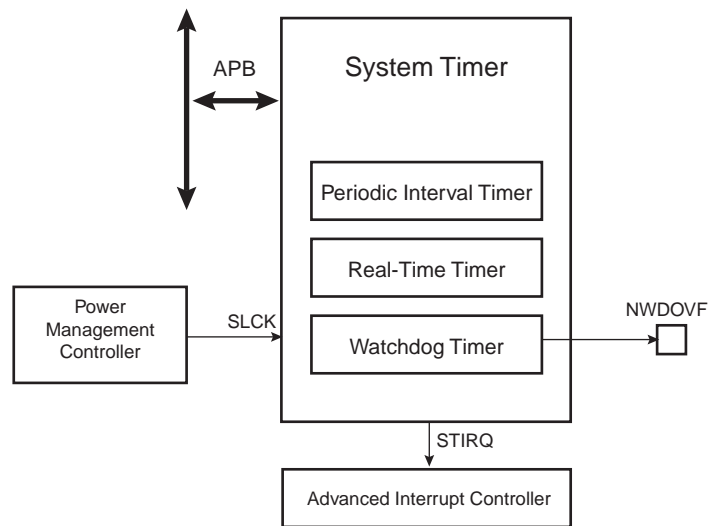
The System Timer provides an interrupt line connected to one of the sources of the Advanced Interrupt Controller (AIC). Interrupt handling requires programming the AIC before configuring the System Timer. Usually, the System Timer interrupt line is connected to the first interrupt source line and shares this entry with the Debug Unit (DBGU) and the Real Time Clock (RTC). This sharing requires the programmer to determine the source of the interrupt when the source 1 is triggered.

Important features of the System Timer include:

- One Period Interval Timer, 16-bit Programmable Counter
- One Watchdog Timer, 16-bit Programmable Counter
- One Real-time Timer, 20-bit Free-running Counter
- Interrupt Generation on Event

### Block Diagram

Figure 129. System Timer Block Diagram



### Application Block Diagram

Figure 130. Application Block Diagram

|                      |                              |                       |
|----------------------|------------------------------|-----------------------|
| OS or RTOS Scheduler | Date, Time and Alarm Manager | System Survey Manager |
| PIT                  | RTT                          | WDT                   |

## Product Dependencies

### Power Management

The System Timer is continuously clocked at 32768 Hz. The power management controller has no effect on the system timer behavior.

### Interrupt Sources

The System Timer interrupt is generally connected to the source 1 of the Advanced Interrupt Controller. This interrupt line is the result of the OR-wiring of the system peripheral interrupt lines (System Timer, Real Time Clock, Power Management Controller, Memory Controller). When a system interrupt happens, the service routine must first determine the cause of the interrupt. This is accomplished by reading successively the status registers of the above mentioned system peripherals.

### Watchdog Overflow

The System Timer is capable of driving the NWDOVF pin. This pin might be implemented or not in a product. When it is implemented, this pin might or not be multiplexed on the PIO Controllers even though it is recommended to dedicate a pin to the watchdog function. If the NWDOVF is multiplexed on a PIO Controller, this last should be first programmed to assign the pin to the watchdog function before using the pin as NWDOVF.

When it is not implemented, programming the associated bits and registers has no effect on the behavior of the System Timer.

## Functional Description

### System Timer Clock

The System Timer uses only the SLCK clock so that it is capable to provide periodic, watchdog, second change or alarm interrupt even if the Power Management Controller is programmed to put the product in Slow Clock Mode. If the product has the capability to back up the Slow Clock oscillator and the System Timer, the System Timer can continue to operate.

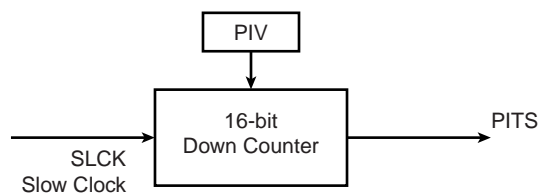
### Period Interval Timer (PIT)

The Period Interval Timer can be used to provide periodic interrupts for use by operating systems. The reset value of the PIT is 0 corresponding to the maximum value. 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.

**Warning:** 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.

**Figure 131.** Period Interval Timer



## Watchdog Timer (WDT)

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).

At reset, the value of the ST\_WDMR is 0x00020000, corresponding to the maximum value of the counter. The watchdog overflow signal is tied low during 8 slow clock cycles when a watchdog overflow occurs (EXTEN bit set in ST\_WDMR).

It uses the Slow Clock divided by 128 to establish the maximum watchdog period to be 256 seconds (with a typical slow clock of 32.768 kHz).

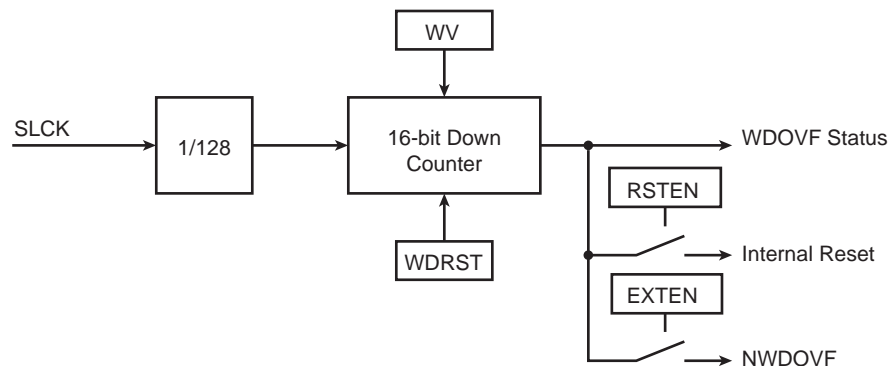
In normal operation, the user reloads the Watchdog at regular intervals before the timer overflow occurs, by setting the bit WDRST in the ST\_CR (Control Register).

If an overflow does occur, the watchdog timer:

- Sets the WDOVF bit in ST\_SR (Status Register), from which an interrupt can be generated.
- Generates a pulse for 8 slow clock cycles on the external signal watchdog overflow 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 counter is immediately reloaded from ST\_WDMR and restarted and the Slow Clock 128 divider is also immediately reset and restarted.

**Figure 132.** Watchdog Timer



## Real-time Timer (RTT)

The Real-Time Timer is used to count elapsed seconds. It is built around a 20-bit counter fed by 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.

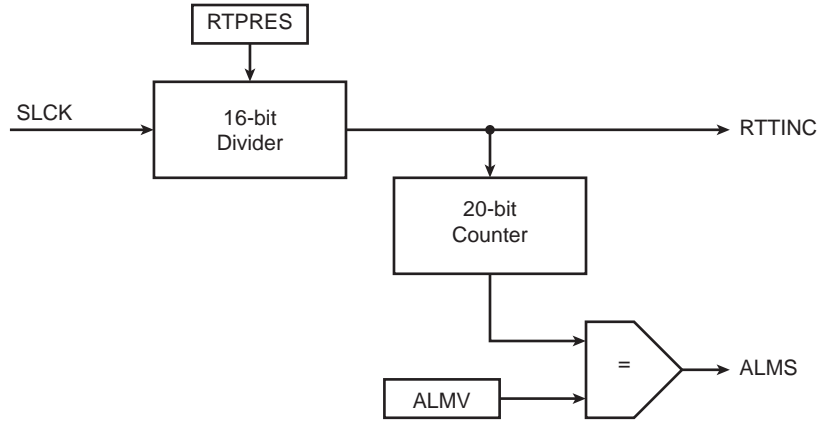
This current value of the counter is compared with the value written in the alarm register ST\_RTAR (Real-time Alarm Register). If the counter value matches the alarm, the bit ALMS in TC\_SR is set. The alarm register is set to its maximum value, corresponding to 0, after a reset.

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.

**Warning:** 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.

**Figure 133.** Real Time Timer





## System Timer (ST) User Interface

**Table 62.** System Timer Registers

| Offset | Register                      | Name    | Access     | Reset Value |
|--------|-------------------------------|---------|------------|-------------|
| 0x0000 | Control Register              | ST_CR   | Write-only | –           |
| 0x0004 | Period Interval Mode Register | ST_PIMR | Read/Write | 0x00000000  |
| 0x0008 | Watchdog Mode Register        | ST_WDMR | Read/Write | 0x00020000  |
| 0x000C | Real-time Mode Register       | ST_RTMR | Read/Write | 0x00008000  |
| 0x0010 | Status Register               | ST_SR   | Read-only  | –           |
| 0x0014 | Interrupt Enable Register     | ST_IER  | Write-only | –           |
| 0x0018 | Interrupt Disable Register    | ST_IDR  | Write-only | –           |
| 0x001C | Interrupt Mask Register       | ST_IMR  | Write-only | 0x0         |
| 0x0020 | Real-time Alarm Register      | ST_RTAR | Read/Write | 0x0         |
| 0x0024 | Current Real-time Register    | ST_CRTR | Read-only  | 0x0         |

### ST Control Register

**Register Name:** ST\_CR

**Access Type:** Write-only

|    |    |    |    |    |    |    |       |
|----|----|----|----|----|----|----|-------|
| 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.

## ST Period Interval Mode Register

Register Name: ST\_PIMR

Access Type: Read/Write

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
|---|---|---|---|---|---|---|---|

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | – | – |
|---|---|---|---|---|---|---|---|

|     |  |  |  |  |  |  |  |
|-----|--|--|--|--|--|--|--|
| PIV |  |  |  |  |  |  |  |
|-----|--|--|--|--|--|--|--|

|     |  |  |  |  |  |  |  |
|-----|--|--|--|--|--|--|--|
| 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.

## ST Watchdog Mode Register

Register Name: ST\_WDMR

Access Type: Read/Write

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 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 x 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 watchdog\_overflow is not tied low when a watchdog overflow occurs.

1 = The watchdog\_overflow is tied low during 8 slow clock cycles when a watchdog overflow occurs.

### ST Real-Time Mode Register

Register Name: ST\_RTMR  
 Access Type: Read/Write

|        |    |    |    |    |    |    |    |
|--------|----|----|----|----|----|----|----|
| 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.

### ST Status Register

Register Name: ST\_SR  
 Access Type: Read-only

|    |    |    |    |      |        |       |      |
|----|----|----|----|------|--------|-------|------|
| 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.

## ST Interrupt Enable Register

Register Name: ST\_IER

Access Type: Write-only

|    |    |    |    |      |        |       |      |
|----|----|----|----|------|--------|-------|------|
| 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**
- **WDOVF: Watchdog Overflow Interrupt Enable**
- **RTTINC: Real-time Timer Increment Interrupt Enable**
- **ALMS: Alarm Status Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

## ST Interrupt Disable Register

Register Name: ST\_IDR

Access Type: Write-only

|    |    |    |    |      |        |       |      |
|----|----|----|----|------|--------|-------|------|
| 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**
- **WDOVF: Watchdog Overflow Interrupt Disable**
- **RTTINC: Real-time Timer Increment Interrupt Disable**
- **ALMS: Alarm Status Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

## ST Interrupt Mask Register

Register Name: ST\_IMR

Access Type: Read-only

|    |    |    |    |      |        |       |      |
|----|----|----|----|------|--------|-------|------|
| 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**
- **WDOVF: Watchdog Overflow Interrupt Mask**
- **RTTINC: Real-time Timer Increment Interrupt Mask**
- **ALMS: Alarm Status Interrupt Mask**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

## ST Real-time Alarm Register

Register Name: ST\_RTAR

Access Type: Read/Write

|      |    |    |    |      |    |    |    |
|------|----|----|----|------|----|----|----|
| 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.



## ST Current Real-Time Register

Register Name: ST\_CRTR

Access Type: Read-only

|      |    |    |    |      |    |    |    |
|------|----|----|----|------|----|----|----|
| 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.

## Real Time Controller (RTC)

### Overview

The Real-time Clock (RTC) peripheral is designed for very low power consumption.

It combines a complete time-of-day clock with alarm and a two-hundred-year Gregorian calendar, complemented by a programmable periodic interrupt. The alarm and calendar registers are accessed by a 32-bit data bus.

The time and calendar values are coded in binary-coded decimal (BCD) format. The time format can be 24-hour mode or 12-hour mode with an AM/PM indicator.

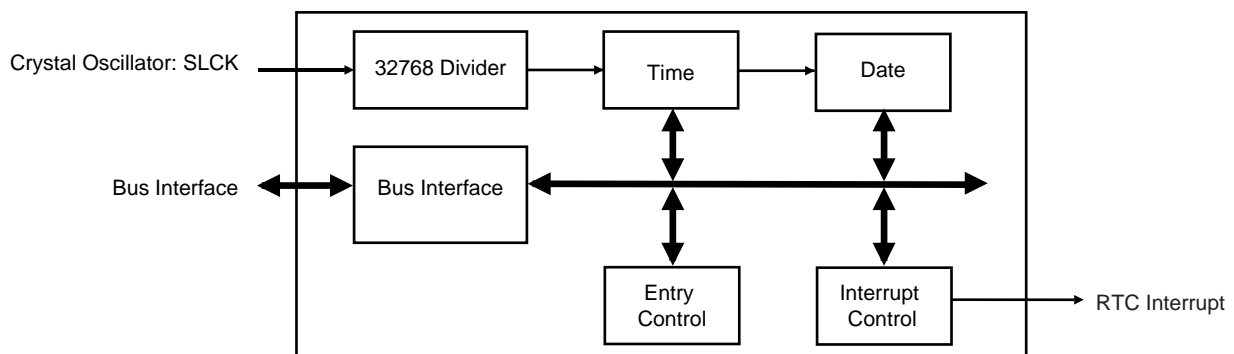
Updating time and calendar fields and configuring the alarm fields are performed by a parallel capture on the 32-bit data bus. An entry control is performed to avoid loading registers with incompatible BCD format data or with an incompatible date according to the current month/year/century.

Important features of the RTC include:

- Low Power Consumption
- Full Asynchronous Design
- Two Hundred Year Calendar
- Programmable Periodic Interrupt
- Alarm and Update Parallel Load
- Control of Alarm and Update Time/Calendar Data In

### Block Diagram

Figure 134. RTC Block Diagram



### Product Dependencies

### Power Management

The Real-time Clock is continuously clocked at 32768 Hz. The Power Management Controller has no effect on RTC behavior.

### Interrupt

The RTC Interrupt is connected to interrupt source 1 (IRQ1) of the advanced interrupt controller. This interrupt line is due to the OR-wiring of the system peripheral interrupt lines (System Timer, Real Time Clock, Power Management Controller, Memory Controller, etc.). When a

system interrupt occurs, the service routine must first determine the cause of the interrupt. This is done by reading the status registers of the above system peripherals successively.

## Functional Description

The RTC provides a full binary-coded decimal (BCD) clock that includes century (19/20), year (with leap years), month, date, day, hours, minutes and seconds.

The valid year range is 1900 to 2099, a two-hundred-year Gregorian calendar achieving full Y2K compliance.

The RTC can operate in 24-hour mode or in 12-hour mode with an AM/PM indicator.

Corrections for leap years are included (all years divisible by 4 being leap years, including year 2000). This is correct up to the year 2099.

After hardware reset, the calendar is initialized to Thursday, January 1, 1998.

## Reference Clock

The reference clock is Slow Clock (SLCK). It can be driven by the Atmel cell OSC55 or OSC56 (or an equivalent cell) and an external 32.768 kHz crystal.

During low power modes of the processor (idle mode), the oscillator runs and power consumption is critical. The crystal selection has to take into account the current consumption for power saving and the frequency drift due to temperature effect on the circuit for time accuracy.

## Timing

The RTC is updated in real time at one-second intervals in normal mode for the counters of seconds, at one-minute intervals for the counter of minutes and so on.

Due to the asynchronous operation of the RTC with respect to the rest of the chip, to be certain that the value read in the RTC registers (century, year, month, date, day, hours, minutes, seconds) are valid and stable, it is necessary to read these registers twice. If the data is the same both times, then it is valid. Therefore, a minimum of two and a maximum of three accesses are required.

## Alarm

The RTC has five programmable fields: month, date, hours, minutes and seconds.

Each of these fields can be enabled or disabled to match the alarm condition:

- If all the fields are enabled, an alarm flag is generated (the corresponding flag is asserted and an interrupt generated if enabled) at a given month, date, hour/minute/second.
- If only the “seconds” field is enabled, then an alarm is generated every minute.

Depending on the combination of fields enabled, a large number of possibilities are available to the user ranging from minutes to 365/366 days.

## Error Checking

Verification on user interface data is performed when accessing the century, year, month, date, day, hours, minutes, seconds and alarms. A check is performed on illegal BCD entries such as illegal date of the month with regard to the year and century configured.

If one of the time fields is not correct, the data is not loaded into the register/counter and a flag is set in the validity register. The user can not reset this flag. It is reset as soon as an acceptable value is programmed. This avoids any further side effects in the hardware. The same procedure is done for the alarm.

The following checks are performed:

1. Century (check if it is in range 19 - 20)
2. Year (BCD entry check)
3. Date (check range 01 - 31)



4. Month (check if it is in BCD range 01 - 12, check validity regarding "date")
5. Day (check range 1 - 7)
6. Hour (BCD checks: in 24-hour mode, check range 00 - 23 and check that AM/PM flag is not set if RTC is set in 24-hour mode; in 12-hour mode check range 01 - 12)
7. Minute (check BCD and range 00 - 59)
8. Second (check BCD and range 00 - 59)

Note: If the 12-hour mode is selected by means of the RTC\_MODE register, a 12-hour value can be programmed and the returned value on RTC\_TIME will be the corresponding 24-hour value. The entry control checks the value of the AM/PM indicator (bit 22 of RTC\_TIME register) to determine the range to be checked.

## Updating Time/Calendar

To update any of the time/calendar fields, the user must first stop the RTC by setting the corresponding field in the Control Register. Bit UPDTIM must be set to update time fields (hour, minute, second) and bit UPDCAL must be set to update calendar fields (century, year, month, date, day).

Then the user must poll or wait for the interrupt (if enabled) of bit ACKUPD in the Status Register. Once the bit reads 1, the user can write to the appropriate register.

Once the update is finished, the user must reset (0) UPDTIM and/or UPDCAL in the Control Register.

When programming the calendar fields, the time fields remain enabled. This avoids a time slip in case the user stays in the calendar update phase for several tens of seconds or more. In successive update operations, the user must wait at least one second after resetting the UPDTIM/UPDCAL bit in the RTC\_CR (Control Register) before setting these bits again. This is done by waiting for the SEC flag in the Status Register before setting UPDTIM/UPDCAL bit. After resetting UPDTIM/UPDCAL, the SEC flag must also be cleared.

## Real Time Controller (RTC) User Interface

**Table 63.** RTC Register Mapping

| Offset | Register                          | Register Name | Read/Write | Reset      |
|--------|-----------------------------------|---------------|------------|------------|
| 0x00   | RTC Control Register              | RTC_CR        | Read/Write | 0x0        |
| 0x04   | RTC Mode Register                 | RTC_MR        | Read/Write | 0x0        |
| 0x08   | RTC Time Register                 | RTC_TIMR      | Read/Write | 0x0        |
| 0x0C   | RTC Calendar Register             | RTC_CALR      | Read/Write | 0x01819819 |
| 0x10   | RTC Time Alarm Register           | RTC_TIMALR    | Read/Write | 0x0        |
| 0x14   | RTC Calendar Alarm Register       | RTC_CALALR    | Read/Write | 0x01010000 |
| 0x18   | RTC Status Register               | RTC_SR        | Read only  | 0x0        |
| 0x1C   | RTC Status Clear Command Register | RTC_SCCR      | Write only | ---        |
| 0x20   | RTC Interrupt Enable Register     | RTC_IER       | Write only | ---        |
| 0x24   | RTC Interrupt Disable Register    | RTC_IDR       | Write only | ---        |
| 0x28   | RTC Interrupt Mask Register       | RTC_IMR       | Read only  | 0x0        |
| 0x2C   | RTC Valid Entry Register          | RTC_VER       | Read only  | 0x0        |

**RTC Control Register**

**Name:** RTC\_CR

**Access Type:** Read/Write

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

• **UPDTIM: Update Request Time Register**

0 = No effect.

1 = Stops the RTC time counting.

Time counting consists of second, minute and hour counters. Time counters can be programmed once this bit is set and acknowledged by the bit ACKUPD of the Status Register.

• **UPDCAL: Update Request Calendar Register**

0 = No effect.

1 = Stops the RTC calendar counting.

Calendar counting consists of day, date, month, year and century counters. Calendar counters can be programmed once this bit is set.

• **TIMEVSEL: Time Event Selection**

The event that generates the flag TIMEV in RTC\_SR (Status Register) depends on the value of TIMEVSEL.

0 = Minute change.

1 = Hour change.

2 = Every day at midnight.

3 = Every day at noon.

• **CALEVSEL: Calendar Event Selection**

The event that generates the flag CALEV in RTC\_SR depends on the value of CALEVSEL.

0 = Week change (every Monday at time 00:00:00).

1 = Month change (every 01 of each month at time 00:00:00).

2, 3 = Year change (every January 1 at time 00:00:00).



## RTC Mode Register

Name: RTC\_MR

Access Type: Read/Write

|    |    |    |    |    |    |    |       |
|----|----|----|----|----|----|----|-------|
| 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     |
| –  | –  | –  | –  | –  | –  | –  | HRMOD |

- **HRMOD: 12-/24-hour Mode**

0 = 24-hour mode is selected.

1 = 12-hour mode is selected.

All non-significant bits read zero.

**RTC Time Register**

**Name:** RTC\_TIMR

**Access Type:** Read/Write

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

- **SEC: Current Second**

The range that can be set is 0 - 59 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **MIN: Current Minute**

The range that can be set is 0 - 59 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **HOUR: Current Hour**

The range that can be set is 1 - 12 (BCD) in 12-hour mode or 0 - 23 (BCD) in 24-hour mode.

- **AMPM: Ante Meridiem Post Meridiem Indicator**

This bit is the AM/PM indicator in 12-hour mode.

0 = AM.

1 = PM.

All non-significant bits read zero.



## RTC Calendar Register

Name: RTC\_CALR

Access Type: Read/Write

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

- **CENT: Current Century**

The range that can be set is 19 - 20 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **YEAR: Current Year**

The range that can be set is 00 - 99 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **MONTH: Current Month**

The range that can be set is 01 - 12 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **DAY: Current Day**

The range that can be set is 1 - 7 (BCD).

The coding of the number (which number represents which day) is user-defined as it has no effect on the date counter.

- **DATE: Current Date**

The range that can be set is 01 - 31 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

All non-significant bits read zero.

### RTC Time Alarm Register

Name: RTC\_TIMALR

Access Type: Read/Write

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

- **SEC: Second Alarm**

This field is the alarm field corresponding to the BCD-coded second counter.

- **SECEN: Second Alarm Enable**

0 = The second-matching alarm is disabled.

1 = The second-matching alarm is enabled.

- **MIN: Minute Alarm**

This field is the alarm field corresponding to the BCD-coded minute counter.

- **MINEN: Minute Alarm Enable**

0 = The minute-matching alarm is disabled.

1 = The minute-matching alarm is enabled.

- **HOUR: Hour Alarm**

This field is the alarm field corresponding to the BCD-coded hour counter.

- **AMPM: AM/PM Indicator**

This field is the alarm field corresponding to the BCD-coded hour counter.

- **HOUREN: Hour Alarm Enable**

0 = The hour-matching alarm is disabled.

1 = The hour-matching alarm is enabled.



## RTC Calendar Alarm Register

Name: RTC\_CALALR

Access Type: Read/Write

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

- **MONTH: Month Alarm**

This field is the alarm field corresponding to the BCD-coded month counter.

- **MTHEN: Month Alarm Enable**

0 = The month-matching alarm is disabled.

1 = The month-matching alarm is enabled.

- **DATE: Date Alarm**

This field is the alarm field corresponding to the BCD-coded date counter.

- **DATEEN: Date Alarm Enable**

0 = The date-matching alarm is disabled.

1 = The date-matching alarm is enabled.



**RTC Status Register**

**Name:** RTC\_SR

**Access Type:** Read-only

|    |    |    |       |       |     |       |        |
|----|----|----|-------|-------|-----|-------|--------|
| 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      |
| –  | –  | –  | CALEV | TIMEV | SEC | ALARM | ACKUPD |

- **ACKUPD: Acknowledge for Update**

0 = Time and calendar registers cannot be updated.

1 = Time and calendar registers can be updated.

- **ALARM: Alarm Flag**

0 = No alarm matching condition occurred.

1 = An alarm matching condition has occurred.

- **SEC: Second Event**

0 = No second event has occurred since the last clear.

1 = At least one second event has occurred since the last clear.

- **TIMEV: Time Event**

0 = No time event has occurred since the last clear.

1 = At least one time event has occurred since the last clear.

The time event is selected in the TIMEVSEL field in RTC\_CTRL (Control Register) and can be any one of the following events: minute change, hour change, noon, midnight (day change).

- **CALEV: Calendar Event**

0 = No calendar event has occurred since the last clear.

1 = At least one calendar event has occurred since the last clear.

The calendar event is selected in the CALEVSEL field in RTC\_CR and can be any one of the following events: week change, month change and year change.

## RTC Status Clear Command Register

**Name:** RTC\_SCCR

**Access Type:** Write-only

|    |    |    |        |        |        |        |        |
|----|----|----|--------|--------|--------|--------|--------|
| 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      |
| –  | –  | –  | CALCLR | TIMCLR | SECCLR | ALRCLR | ACKCLR |

- **Status Flag Clear**

0 = No effect.

1 = Clears corresponding status flag in the Status Register (RTC\_SR).

### RTC Interrupt Enable Register

Name: RTC\_IER

Access Type: Write-only

|    |    |    |       |       |       |       |       |
|----|----|----|-------|-------|-------|-------|-------|
| 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     |
| –  | –  | –  | CALEN | TIMEN | SECEN | ALREN | ACKEN |

- **ACKEN: Acknowledge Update Interrupt Enable**

0 = No effect.

1 = The acknowledge for update interrupt is enabled.

- **ALREN: Alarm Interrupt Enable**

0 = No effect.

1 = The alarm interrupt is enabled.

- **SECEN: Second Event Interrupt Enable**

0 = No effect.

1 = The second periodic interrupt is enabled.

- **TIMEN: Time Event Interrupt Enable**

0 = No effect.

1 = The selected time event interrupt is enabled.

- **CALEN: Calendar Event Interrupt Enable**

0 = No effect.

1 = The selected calendar event interrupt is enabled.

## RTC Interrupt Disable Register

**Name:** RTC\_IDR

**Access Type:** Write-only

|    |    |    |        |        |        |        |        |
|----|----|----|--------|--------|--------|--------|--------|
| 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      |
| –  | –  | –  | CALDIS | TIMDIS | SECDIS | ALRDIS | ACKDIS |

- **ACKDIS: Acknowledge Update Interrupt Disable**

0 = No effect.

1 = The acknowledge for update interrupt is disabled.

- **ALRDIS: Alarm Interrupt Disable**

0 = No effect.

1 = The alarm interrupt is disabled.

- **SECDIS: Second Event Interrupt Disable**

0 = No effect.

1 = The second periodic interrupt is disabled.

- **TIMDIS: Time Event Interrupt Disable**

0 = No effect.

1 = The selected time event interrupt is disabled.

- **CALDIS: Calendar Event Interrupt Disable**

0 = No effect.

1 = The selected calendar event interrupt is disabled.

### RTC Interrupt Mask Register

Name: RTC\_IMR

Access Type: Read-only

|    |    |    |     |     |     |     |     |
|----|----|----|-----|-----|-----|-----|-----|
| 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   |
| –  | –  | –  | CAL | TIM | SEC | ALR | ACK |

- **ACK: Acknowledge Update Interrupt Mask**

0 = The acknowledge for update interrupt is disabled.

1 = The acknowledge for update interrupt is enabled.

- **ALR: Alarm Interrupt Mask**

0 = The alarm interrupt is disabled.

1 = The alarm interrupt is enabled.

- **SEC: Second Event Interrupt Mask**

0 = The second periodic interrupt is disabled.

1 = The second periodic interrupt is enabled.

- **TIM: Time Event Interrupt Mask**

0 = The selected time event interrupt is disabled.

1 = The selected time event interrupt is enabled.

- **CAL: Calendar Event Interrupt Mask**

0 = The selected calendar event interrupt is disabled.

1 = The selected calendar event interrupt is enabled.

## RTC Valid Entry Register

**Name:** RTC\_VER

**Access Type:** Read-only

|    |    |    |    |         |          |       |       |
|----|----|----|----|---------|----------|-------|-------|
| 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     |
| –  | –  | –  | –  | NVCALAR | NVTIMALR | NVCAL | NVTIM |

- **NVTIM: Non valid Time**

0 = No invalid data has been detected in RTC\_TIMR (Time Register).

1 = RTC\_TIMR has contained invalid data since it was last programmed.

- **NVCAL: Non valid Calendar**

0 = No invalid data has been detected in RTC\_CALR (Calendar Register).

1 = RTC\_CALR has contained invalid data since it was last programmed.

- **NVTIMALR: Non valid Time Alarm**

0 = No invalid data has been detected in RTC\_TIMALR (Time Alarm Register).

1 = RTC\_TIMALR has contained invalid data since it was last programmed.

- **NVCALALR: Non valid Calendar Alarm**

0 = No invalid data has been detected in RTC\_CALALR (Calendar Alarm Register).

1 = RTC\_CALALR has contained invalid data since it was last programmed.

## Debug Unit (DBGU)

### Overview

The Debug Unit provides a single entry point from the processor for access to all the debug capabilities of Atmel's ARM-based systems.

The Debug Unit features a two-pin UART that can be used for several debug and trace purposes and offers an ideal medium for in-situ programming solutions and debug monitor communications. Moreover, the association with two peripheral data controller channels permits packet handling for these tasks with processor time reduced to a minimum.

The Debug Unit also makes the Debug Communication Channel (DCC) signals provided by the In-circuit Emulator of the ARM processor visible to the software. These signals indicate the status of the DCC read and write registers and generate an interrupt to the ARM processor, making possible the handling of the DCC under interrupt control.

Chip Identifier registers permit recognition of the device and its revision. These registers inform as to the sizes and types of the on-chip memories, as well as the set of embedded peripherals.

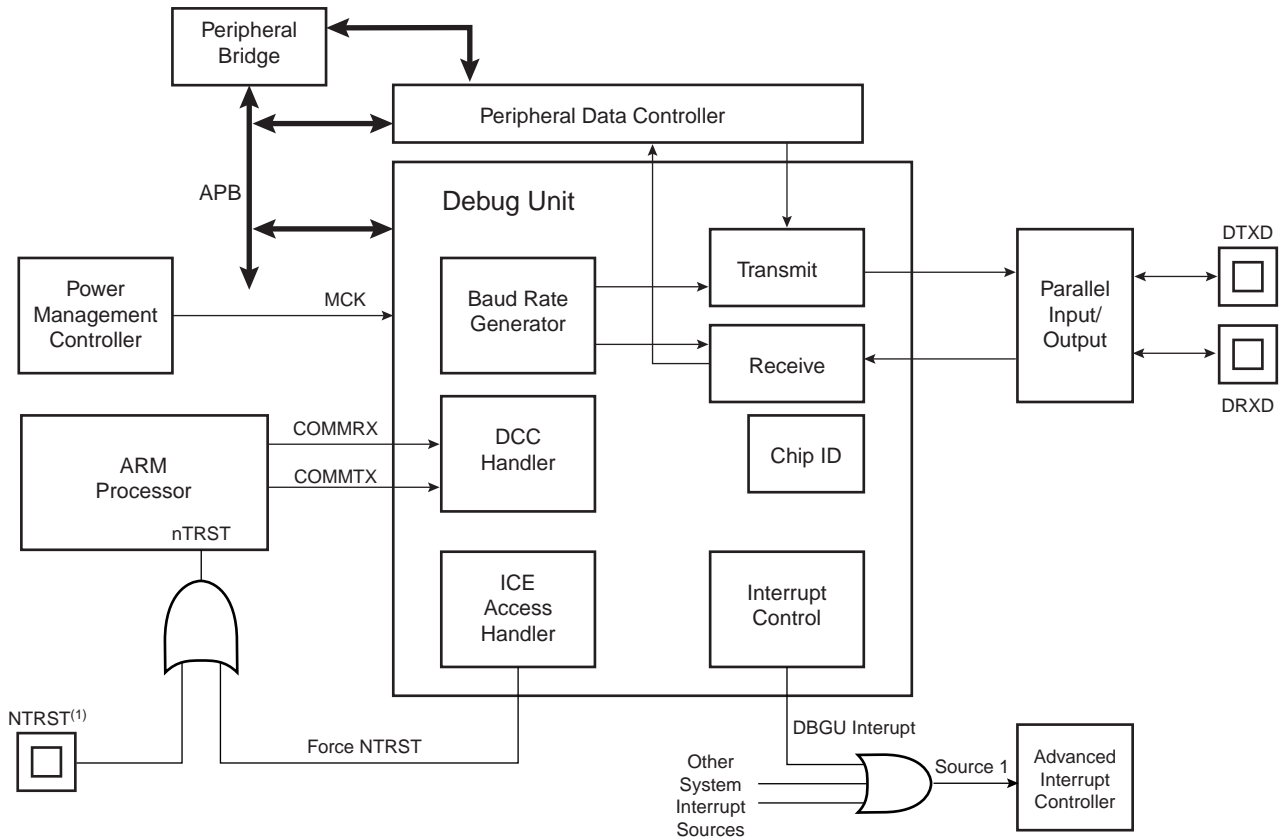
Finally, the Debug Unit features a Force NTRST capability that enables the software to decide whether to prevent access to the system via the In-circuit Emulator. This permits protection of the code, stored in ROM.

Important features of the Debug Unit are:

- System Peripheral to Facilitate Debug of Atmel's ARM-based Systems
- Composed of Four Functions
  - Two-pin UART
  - Debug Communication Channel (DCC) Support
  - Chip ID Registers
  - ICE Access Prevention
- Two-pin UART
  - Implemented Features are 100% Compatible with the Standard Atmel USART
  - Independent Receiver and Transmitter with a Common Programmable Baud Rate Generator
  - Even, Odd, Mark or Space Parity Generation
  - Parity, Framing and Overrun Error Detection
  - Automatic Echo, Local Loopback and Remote Loopback Channel Modes
  - Interrupt Generation
  - Support for Two PDC Channels with Connection to Receiver and Transmitter
- Debug Communication Channel Support
  - Offers Visibility of COMMRX and COMMTX Signals from the ARM Processor
  - Interrupt Generation
- Chip ID Registers
  - Identification of the Device Revision, Sizes of the Embedded Memories, Set of Peripherals
- ICE Access Prevention
  - Enables Software to Prevent System Access Through the ARM Processor's ICE
  - Prevention is Made by Asserting the NTRST Line of the ARM Processor's ICE

## Block Diagram

**Figure 135.** Debug Unit Functional Block Diagram

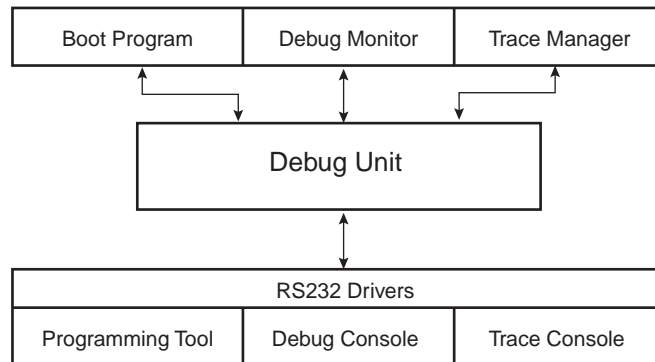


Note: 1. If NTRST pad is not bonded out, it is connected to NRST.

**Table 64.** Debug Unit Pin Description

| Pin Name | Description         | Type   |
|----------|---------------------|--------|
| DRXD     | Debug Receive Data  | Input  |
| DTXD     | Debug Transmit Data | Output |

**Figure 136.** Debug Unit Application Example





## Product Dependencies

### I/O Lines

Depending on product integration, the Debug Unit pins may be multiplexed with PIO lines. In this case, the programmer must first configure the corresponding PIO Controller to enable I/O lines operations of the Debug Unit.

### Power Management

Depending on product integration, the Debug Unit clock may be controllable through the Power Management Controller. In this case, the programmer must first configure the PMC to enable the Debug Unit clock. Usually, the peripheral identifier used for this purpose is 1.

### Interrupt Source

Depending on product integration, the Debug Unit interrupt line is connected to one of the interrupt sources of the Advanced Interrupt Controller. Interrupt handling requires programming of the AIC before configuring the Debug Unit. Usually, the Debug Unit interrupt line connects to the interrupt source 1 of the AIC, which may be shared with the real-time clock, the system timer interrupt lines and other system peripheral interrupts, as shown in Figure 135. This sharing requires the programmer to determine the source of the interrupt when the source 1 is triggered.

## UART Operations

The Debug Unit operates as a UART, (asynchronous mode only) and supports only 8-bit character handling (with parity). It has no clock pin.

The Debug Unit's UART is made up of a receiver and a transmitter that operate independently, and a common baud rate generator. Receiver timeout and transmitter time guard are not implemented. However, all the implemented features are compatible with those of a standard USART.

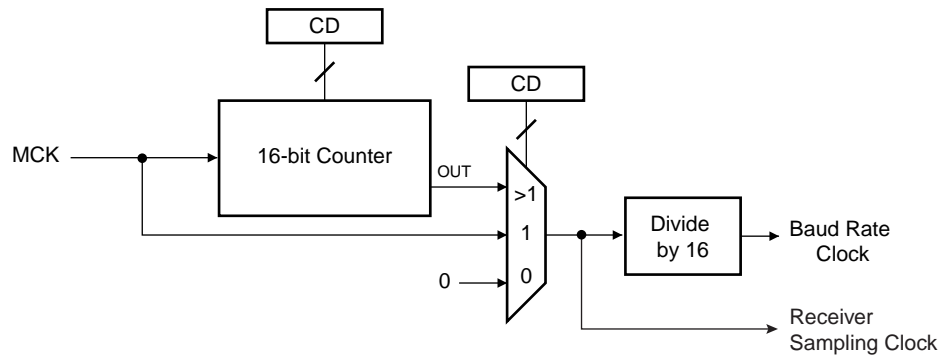
### Baud Rate Generator

The baud rate generator provides the bit period clock named baud rate clock to both the receiver and the transmitter.

The baud rate clock is the master clock divided by 16 times the value (CD) written in DBGU\_BRGR (Baud Rate Generator Register). If DBGU\_BRGR is set to 0, the baud rate clock is disabled and the Debug Unit's UART remains inactive. The maximum allowable baud rate is Master Clock divided by 16. The minimum allowable baud rate is Master Clock divided by (16 x 65536).

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

**Figure 137. Baud Rate Generator**



## Receiver

### Receiver Reset, Enable and Disable

After device reset, the Debug Unit receiver is disabled and must be enabled before being used. The receiver can be enabled by writing the control register `DBGU_CR` with the bit `RXEN` at 1. At this command, the receiver starts looking for a start bit.

The programmer can disable the receiver by writing `DBGU_CR` with the bit `RXDIS` at 1. If the receiver is waiting for a start bit, it is immediately stopped. However, if the receiver has already detected a start bit and is receiving the data, it waits for the stop bit before actually stopping its operation.

The programmer can also put the receiver in its reset state by writing `DBGU_CR` with the bit `RSTRX` at 1. In doing so, the receiver immediately stops its current operations and is disabled, whatever its current state. If `RSTRX` is applied when data is being processed, this data is lost.

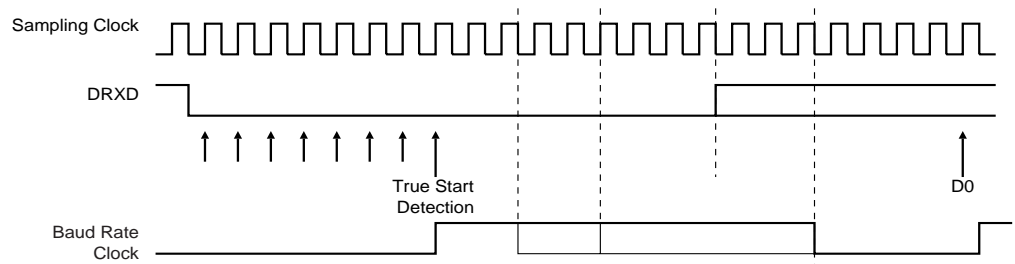
### Start Detection and Data Sampling

The Debug Unit only supports asynchronous operations, and this affects only its receiver. The Debug Unit receiver detects the start of a received character by sampling the `DRXD` signal until it detects a valid start bit. A low level (space) on `DRXD` 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 that 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 `DRXD` at the theoretical midpoint of each bit. It is assumed that each bit lasts 16 cycles of the sampling clock (1-bit period) so the bit sampling point is eight cycles (0.5-bit period) 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.

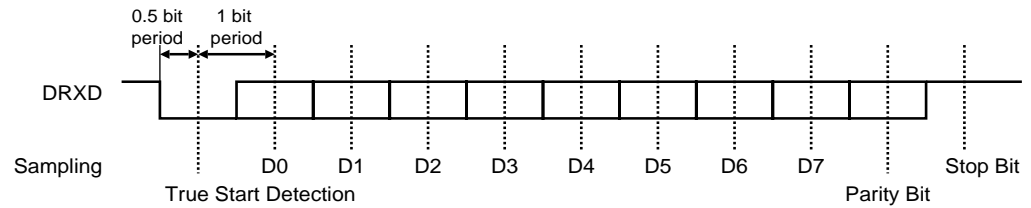
Each subsequent bit is sampled 16 cycles (1-bit period) after the previous one.

**Figure 138. Start Bit Detection**



**Figure 139. Character Reception**

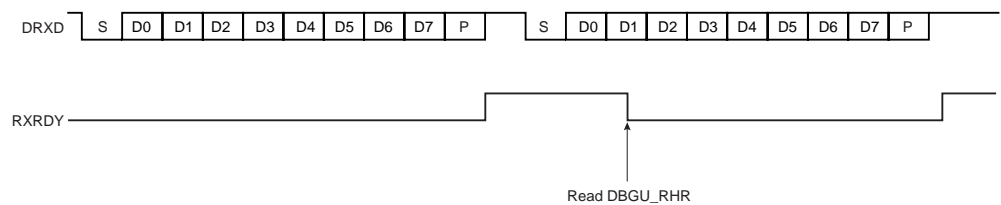
Example: 8-bit, parity enabled 1 stop



**Receiver Ready**

When a complete character is received, it is transferred to the DBGU\_RHR and the RXRDY status bit in DBGU\_SR (Status Register) is set. The bit RXRDY is automatically cleared when the receive holding register DBGU\_RHR is read.

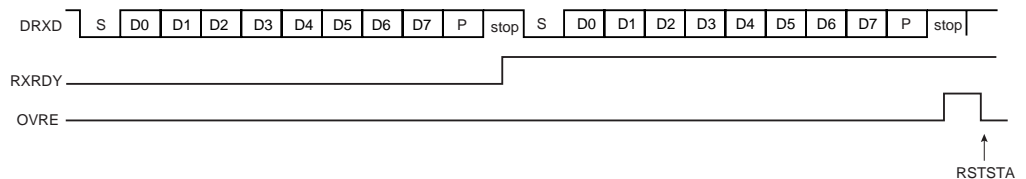
**Figure 140. Receiver Ready**



**Receiver Overrun**

If DBGU\_RHR has not been read by the software (or the Peripheral Data Controller) since the last transfer, the RXRDY bit is still set and a new character is received, the OVRE status bit in DBGU\_SR is set. OVRE is cleared when the software writes the control register DBGU\_CR with the bit RSTSTA (Reset Status) at 1.

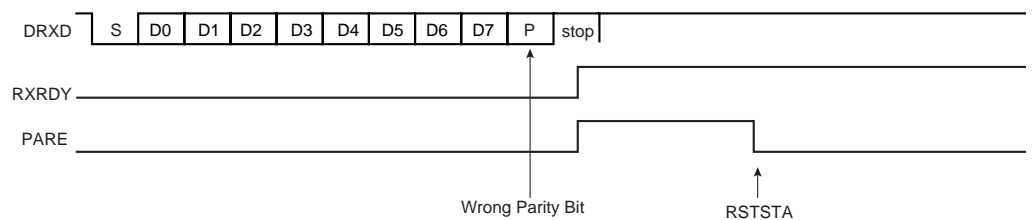
**Figure 141. Receiver Overrun**



**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 DBGU\_MR. It then compares the result with the received parity bit. If different, the parity error bit PARE in DBGU\_SR is set at the same time the RXRDY is set. The parity bit is cleared when the control register DBGU\_CR is written with the bit RSTSTA (Reset Status) at 1. If a new character is received before the reset status command is written, the PARE bit remains at 1.

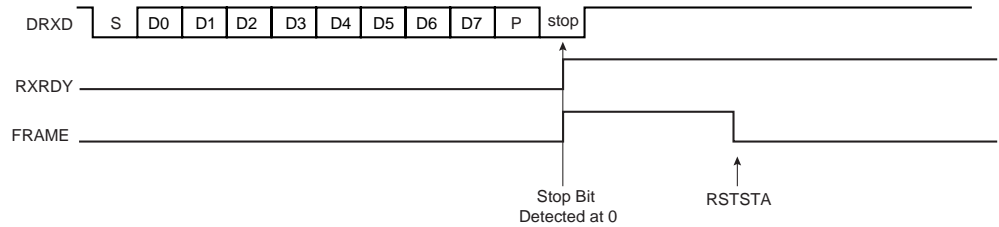
**Figure 142. Parity Error**



## Receiver Framing Error

When a start bit is detected, it generates a character reception when all the data bits have been sampled. The stop bit is also sampled and when it is detected at 0, the FRAME (Framing Error) bit in DBGU\_SR is set at the same time the RXRDY bit is set. The bit FRAME remains high until the control register DBGU\_CR is written with the bit RSTSTA at 1.

**Figure 143.** Receiver Framing Error



## Transmitter

### Transmitter Reset, Enable and Disable

After device reset, the Debug Unit transmitter is disabled and it must be enabled before being used. The transmitter is enabled by writing the control register DBGU\_CR with the bit TXEN at 1. From this command, the transmitter waits for a character to be written in the Transmit Holding Register DBGU\_THR before actually starting the transmission.

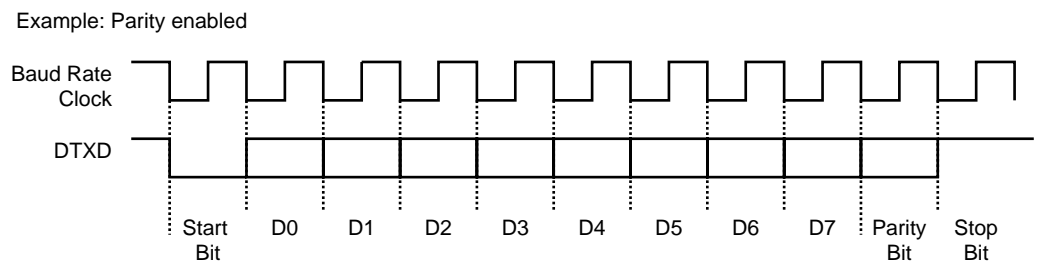
The programmer can disable the transmitter by writing DBGU\_CR with the bit TXDIS at 1. If the transmitter is not operating, it is immediately stopped. However, if a character is being processed into the Shift Register and/or a character has been written in the Transmit Holding Register, the characters are completed before the transmitter is actually stopped.

The programmer can also put the transmitter in its reset state by writing the DBGU\_CR with the bit RSTTX at 1. This immediately stops the transmitter, whether or not it is processing characters.

### Transmit Format

The Debug Unit transmitter drives the pin DTXD at the baud rate clock speed. The line is driven depending on the format defined in the Mode Register and the data stored in the Shift Register. One start bit at level 0, then the 8 data bits, from the lowest to the highest bit, one optional parity bit and one stop bit at 1 are consecutively shifted out as shown on the following figure. The field PARE in the mode register DBGU\_MR defines whether or not a parity bit is shifted out. When a parity bit is enabled, it can be selected between an odd parity, an even parity, or a fixed space or mark bit.

**Figure 144.** Character Transmission



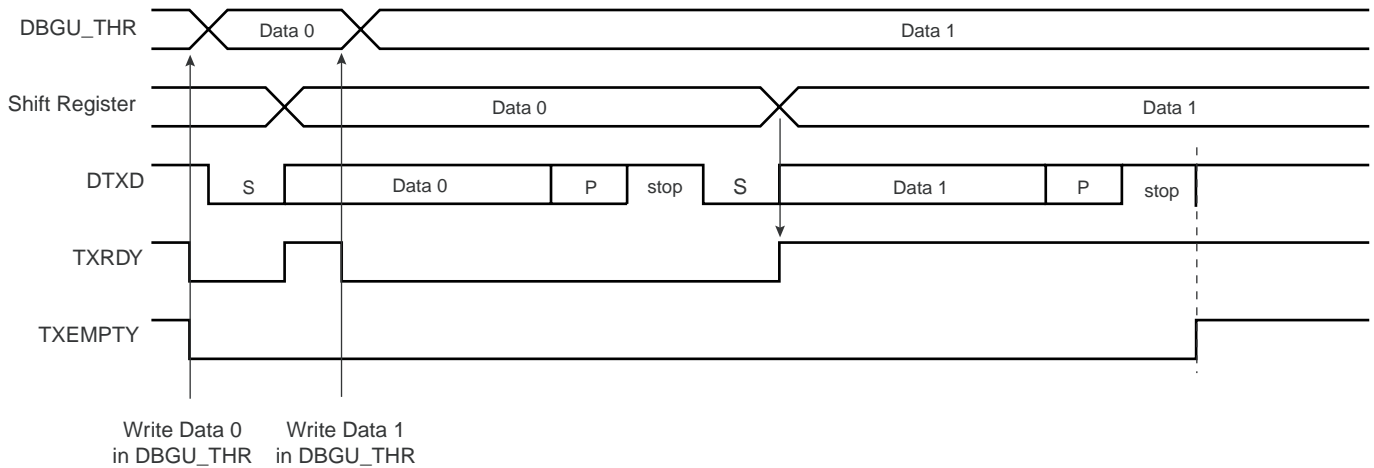
### Transmitter Control

When the transmitter is enabled, the bit TXRDY (Transmitter Ready) is set in the status register DBGU\_SR. The transmission starts when the programmer writes in the Transmit Holding Register DBGU\_THR, and after the written character is transferred from DBGU\_THR to the Shift Register. The bit TXRDY remains high until a second character is written in DBGU\_THR.

As soon as the first character is completed, the last character written in DBGU\_THR is transferred into the shift register and TXRDY rises again, showing that the holding register is empty.

When both the Shift Register and the DBGU\_THR are empty, i.e., all the characters written in DBGU\_THR have been processed, the bit TXEMPTY rises after the last stop bit has been completed.

**Figure 145. Transmitter Control**



## Peripheral Data Controller

Both the receiver and the transmitter of the Debug Unit's UART are generally connected to a Peripheral Data Controller (PDC) channel.

The peripheral data controller channels are programmed via registers that are mapped within the Debug Unit user interface from the offset 0x100. The status bits are reported in the Debug Unit status register DBGU\_SR and can generate an interrupt.

The RXRDY bit triggers the PDC channel data transfer of the receiver. This results in a read of the data in DBGU\_RHR. The TXRDY bit triggers the PDC channel data transfer of the transmitter. This results in a write of a data in DBGU\_THR.

## Test Modes

The Debug Unit supports three tests modes. These modes of operation are programmed by using the field CHMODE (Channel Mode) in the mode register DBGU\_MR.

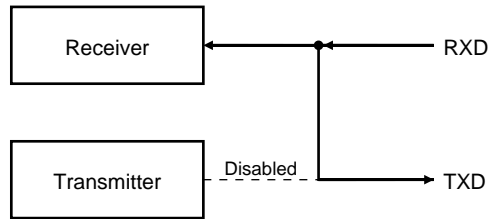
The Automatic Echo mode allows bit-by-bit retransmission. When a bit is received on the DRXD line, it is sent to the DTXD line. The transmitter operates normally, but has no effect on the DTXD line.

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

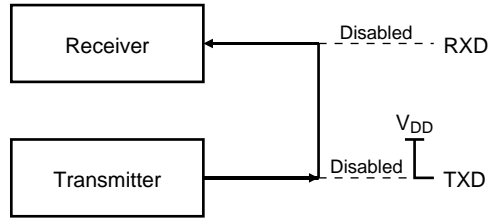
The Remote Loopback mode directly connects the DRXD pin to the DTXD line. The transmitter and the receiver are disabled and have no effect. This mode allows a bit-by-bit retransmission.

**Figure 146. Test Modes**

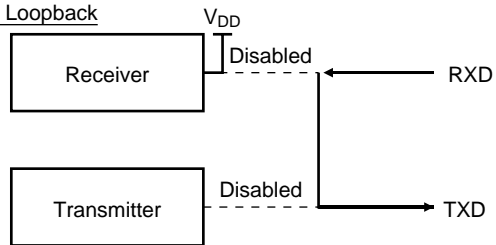
Automatic Echo



Local Loopback



Remote Loopback



## Debug Communication Channel Support

The Debug Unit handles the signals COMMRX and COMMTX that come from the Debug Communication Channel of the ARM Processor and are driven by the In-circuit Emulator.

The Debug Communication Channel contains two registers that are accessible through the ICE Breaker on the JTAG side and through the coprocessor 0 on the ARM Processor side.

As a reminder, the following instructions are used to read and write the Debug Communication Channel:

```
MRC    p14, 0, Rd, c1, c0, 0
```

Returns the debug communication data read register into Rd

```
MCR    p14, 0, Rd, c1, c0, 0
```

Writes the value in Rd to the debug communication data write register.

The bits COMMRX and COMMTX, which indicate, respectively, that the read register has been written by the debugger but not yet read by the processor, and that the write register has been written by the processor and not yet read by the debugger, are wired on the two highest bits of the status register DBGU\_SR. These bits can generate an interrupt. This feature permits handling under interrupt a debug link between a debug monitor running on the target system and a debugger.

## Chip Identifier

The Debug Unit features two chip identifier registers, DBGU\_CIDR (Chip ID Register) and DBGU\_EXID (Extension ID). Both registers contain a hard-wired value that is read-only. The first register contains the following fields:

- EXT - shows the use of the extension identifier register
- NVPTYP and NVPSIZ - identifies the type of embedded non-volatile memory and its size
- ARCH - identifies the set of embedded peripheral
- SRAMSIZ - indicates the size of the embedded SRAM
- EPROC - indicates the embedded ARM processor
- VERSION - gives the revision of the silicon

The second register is device-dependent and reads 0 if the bit EXT is 0.

## ICE Access Prevention

The Debug Unit allows blockage of access to the system through the ARM processor's ICE interface. This feature is implemented via the register Force NTRST (DBGU\_FNR), that allows assertion of the NTRST signal of the ICE Interface. Writing the bit FNTRST (Force NTRST) to 1 in this register prevents any activity on the TAP controller.

On standard devices, the bit FNTRST resets to 0 and thus does not prevent ICE access.

This feature is especially useful on custom ROM devices for customers who do not want their on-chip code to be visible.

## Debug Unit User Interface

**Table 65.** Debug Unit Memory Map

| Offset          | Register                     | Name      | Access     | Reset Value |
|-----------------|------------------------------|-----------|------------|-------------|
| 0x0000          | Control Register             | DBGU_CR   | Write-only | –           |
| 0x0004          | Mode Register                | DBGU_MR   | Read/Write | 0x0         |
| 0x0008          | Interrupt Enable Register    | DBGU_IER  | Write-only | –           |
| 0x000C          | Interrupt Disable Register   | DBGU_IDR  | Write-only | –           |
| 0x0010          | Interrupt Mask Register      | DBGU_IMR  | Read-only  | 0x0         |
| 0x0014          | Status Register              | DBGU_SR   | Read-only  | –           |
| 0x0018          | Receive Holding Register     | DBGU_RHR  | Read-only  | 0x0         |
| 0x001C          | Transmit Holding Register    | DBGU_THR  | Write-only | –           |
| 0x0020          | Baud Rate Generator Register | DBGU_BRGR | Read/Write | 0x0         |
| 0x0024 - 0x003C | Reserved                     | –         | –          | –           |
| 0x0040          | Chip ID Register             | DBGU_CIDR | Read-only  | –           |
| 0x0044          | Chip ID Extension Register   | DBGU_EXID | Read-only  | –           |
| 0x0048          | Force NTRST Register         | DBGU_FNR  | Read/Write | 0x0         |
| 0x004C - 0x00FC | Reserved                     | –         | –          | –           |
| 0x0100 - 0x0124 | PDC Area                     | –         | –          | –           |



### Debug Unit Control Register

Name: DBGU\_CR

Access Type: Write-only

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

• **RSTRX: Reset Receiver**

0 = No effect.

1 = The receiver logic is reset and disabled. If a character is being received, the reception is aborted.

• **RSTTX: Reset Transmitter**

0 = No effect.

1 = The transmitter logic is reset and disabled. If a character is being transmitted, the transmission is aborted.

• **RXEN: Receiver Enable**

0 = No effect.

1 = The receiver is enabled if RXDIS is 0.

• **RXDIS: Receiver Disable**

0 = No effect.

1 = The receiver is disabled. If a character is being processed and RSTRX is not set, the character is completed before the receiver is stopped.

• **TXEN: Transmitter Enable**

0 = No effect.

1 = The transmitter is enabled if TXDIS is 0.

• **TXDIS: Transmitter Disable**

0 = No effect.

1 = The transmitter is disabled. If a character is being processed and a character has been written the DBGU\_THR and RSTTX is not set, both characters are completed before the transmitter is stopped.

• **RSTSTA: Reset Status Bits**

0 = No effect.

1 = Resets the status bits PARE, FRAME and OVRE in the DBGU\_SR.

## Debug Unit Mode Register

Name: DBGU\_MR

Access Type: Read/Write

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

- **PAR: Parity Type**

| PAR |   |   | Parity Type               |
|-----|---|---|---------------------------|
| 0   | 0 | 0 | Even parity               |
| 0   | 0 | 1 | Odd parity                |
| 0   | 1 | 0 | Space: parity forced to 0 |
| 0   | 1 | 1 | Mark: parity forced to 1  |
| 1   | x | x | No parity                 |

- **CHMODE: Channel Mode**

| CHMODE |   | Mode Description |
|--------|---|------------------|
| 0      | 0 | Normal Mode      |
| 0      | 1 | Automatic Echo   |
| 1      | 0 | Local Loopback   |
| 1      | 1 | Remote Loopback  |

### Debug Unit Interrupt Enable Register

Name: DBGU\_IER

Access Type: Write-only

|        |        |      |        |        |    |         |       |
|--------|--------|------|--------|--------|----|---------|-------|
| 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     |
| –      | –      | –    | RXBUFF | TXBUFE | –  | TXEMPTY | –     |
| 7      | 6      | 5    | 4      | 3      | 2  | 1       | 0     |
| PARE   | FRAME  | OVRE | ENDTX  | ENDRX  | –  | TXRDY   | RXRDY |

- **RXRDY: Enable RXRDY Interrupt**
- **TXRDY: Enable TXRDY Interrupt**
- **ENDRX: Enable End of Receive Transfer Interrupt**
- **ENDTX: Enable End of Transmit Interrupt**
- **OVRE: Enable Overrun Error Interrupt**
- **FRAME: Enable Framing Error Interrupt**
- **PARE: Enable Parity Error Interrupt**
- **TXEMPTY: Enable TXEMPTY Interrupt**
- **TXBUFE: Enable Buffer Empty Interrupt**
- **RXBUFF: Enable Buffer Full Interrupt**
- **COMMTX: Enable COMMTX (from ARM) Interrupt**
- **COMMRX: Enable COMMRX (from ARM) Interrupt**

0 = No effect.

1 = Enables the corresponding interrupt.

## Debug Unit Interrupt Disable Register

Name: DBGU\_IDR

Access Type: Write-only

|        |        |      |        |        |    |         |       |
|--------|--------|------|--------|--------|----|---------|-------|
| 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     |
| –      | –      | –    | RXBUFF | TXBUFE | –  | TXEMPTY | –     |
| 7      | 6      | 5    | 4      | 3      | 2  | 1       | 0     |
| PARE   | FRAME  | OVRE | ENDTX  | ENDRX  | –  | TXRDY   | RXRDY |

- **RXRDY: Disable RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **ENDRX: Disable End of Receive Transfer Interrupt**
- **ENDTX: Disable End of Transmit Interrupt**
- **OVRE: Disable Overrun Error Interrupt**
- **FRAME: Disable Framing Error Interrupt**
- **PARE: Disable Parity Error Interrupt**
- **TXEMPTY: Disable TXEMPTY Interrupt**
- **TXBUFE: Disable Buffer Empty Interrupt**
- **RXBUFF: Disable Buffer Full Interrupt**
- **COMMTX: Disable COMMTX (from ARM) Interrupt**
- **COMMRX: Disable COMMRX (from ARM) Interrupt**

0 = No effect.

1 = Disables the corresponding interrupt.

### Debug Unit Interrupt Mask Register

Name: DBGU\_IMR

Access Type: Read-only

|        |        |      |        |        |    |         |       |
|--------|--------|------|--------|--------|----|---------|-------|
| 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     |
| –      | –      | –    | RXBUFF | TXBUFE | –  | TXEMPTY | –     |
| 7      | 6      | 5    | 4      | 3      | 2  | 1       | 0     |
| PARE   | FRAME  | OVRE | ENDTX  | ENDRX  | –  | TXRDY   | RXRDY |

- **RXRDY: Mask RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **ENDRX: Mask End of Receive Transfer Interrupt**
- **ENDTX: Mask End of Transmit Interrupt**
- **OVRE: Mask Overrun Error Interrupt**
- **FRAME: Mask Framing Error Interrupt**
- **PARE: Mask Parity Error Interrupt**
- **TXEMPTY: Mask TXEMPTY Interrupt**
- **TXBUFE: Mask TXBUFE Interrupt**
- **RXBUFF: Mask RXBUFF Interrupt**
- **COMMTX: Mask COMMTX Interrupt**
- **COMMRX: Mask COMMRX Interrupt**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

## Debug Unit Status Register

**Name:** DBGU\_SR

**Access Type:** Read-only

|        |        |      |        |        |    |         |       |
|--------|--------|------|--------|--------|----|---------|-------|
| 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     |
| –      | –      | –    | RXBUFF | TXBUFE | –  | TXEMPTY | –     |
| 7      | 6      | 5    | 4      | 3      | 2  | 1       | 0     |
| PARE   | FRAME  | OVRE | ENDTX  | ENDRX  | –  | TXRDY   | RXRDY |

- **RXRDY: Receiver Ready**

0 = No character has been received since the last read of the DBGU\_RHR or the receiver is disabled.

1 = At least one complete character has been received, transferred to DBGU\_RHR and not yet read.

- **TXRDY: Transmitter Ready**

0 = A character has been written to DBGU\_THR and not yet transferred to the Shift Register, or the transmitter is disabled.

1 = There is no character written to DBGU\_THR not yet transferred to the Shift Register.

- **ENDRX: End of Receiver Transfer**

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

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

- **ENDTX: End of Transmitter Transfer**

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

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

- **OVRE: Overrun Error**

0 = No overrun error has occurred since the last RSTSTA.

1 = At least one overrun error has occurred since the last RSTSTA.

- **FRAME: Framing Error**

0 = No framing error has occurred since the last RSTSTA.

1 = At least one framing error has occurred since the last RSTSTA.

- **PARE: Parity Error**

0 = No parity error has occurred since the last RSTSTA.

1 = At least one parity error has occurred since the last RSTSTA.

- **TXEMPTY: Transmitter Empty**

0 = There are characters in DBGU\_THR, or characters being processed by the transmitter, or the transmitter is disabled.

1 = There are no characters in DBGU\_THR and there are no characters being processed by the transmitter.

- **TXBUFE: Transmission Buffer Empty**

0 = The buffer empty signal from the transmitter PDC channel is inactive.

1 = The buffer empty signal from the transmitter PDC channel is active.

- **RXBUFF: Receive Buffer Full**

0 = The buffer full signal from the receiver PDC channel is inactive.

1 = The buffer full signal from the receiver PDC channel is active.

- **COMMTX: Debug Communication Channel Write Status**

0 = COMMTX from the ARM processor is inactive.

1 = COMMTX from the ARM processor is active.

- **COMMRX: Debug Communication Channel Read Status**

0 = COMMRX from the ARM processor is inactive.

1 = COMMRX from the ARM processor is active.

## Debug Unit Receiver Holding Register

Name: DBGU\_RHR

Access Type: Read-only

|       |    |    |    |    |    |    |    |
|-------|----|----|----|----|----|----|----|
| 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  |
| RXCHR |    |    |    |    |    |    |    |

- **RXCHR: Received Character**

Last received character if RXRDY is set.

## Debug Unit Transmit Holding Register

Name: DBGU\_THR

Access Type: Write-only

|       |    |    |    |    |    |    |    |
|-------|----|----|----|----|----|----|----|
| 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  |
| TXCHR |    |    |    |    |    |    |    |

- **TXCHR: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set.



**Debug Unit Baud Rate Generator Register**

Name: DBGU\_BRGR

Access Type: Read/Write

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 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**

| CD         | Baud Rate Clock        |
|------------|------------------------|
| 0          | Disabled               |
| 1          | MCK                    |
| 2 to 65535 | $MCK / (CD \times 16)$ |



## Debug Unit Chip ID Register

Name: DBGU\_CIDR

Access Type: Read-only

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

- **VERSION:** Version of the device
- **EPROC:** Embedded Processor

| EPROC |   |   | Processor |
|-------|---|---|-----------|
| 0     | 0 | 1 | ARM946ES  |
| 0     | 1 | 0 | ARM7TDMI  |
| 1     | 0 | 0 | ARM920T   |

- **NVPSIZ:** Nonvolatile Program Memory Size

| NVPSIZ |   |   |   | Size       |
|--------|---|---|---|------------|
| 0      | 0 | 0 | 0 | None       |
| 0      | 0 | 0 | 1 | 8K bytes   |
| 0      | 0 | 1 | 0 | 16K bytes  |
| 0      | 0 | 1 | 1 | 32K bytes  |
| 0      | 1 | 0 | 0 | Reserved   |
| 0      | 1 | 0 | 1 | 64K bytes  |
| 0      | 1 | 1 | 0 | Reserved   |
| 0      | 1 | 1 | 1 | 128K bytes |
| 1      | 0 | 0 | 0 | Reserved   |
| 1      | 0 | 0 | 1 | 256K bytes |
| 1      | 0 | 1 | 0 | Reserved   |
| 1      | 0 | 1 | 1 | Reserved   |
| 1      | 1 | 0 | 0 | Reserved   |
| 1      | 1 | 0 | 1 | Reserved   |
| 1      | 1 | 1 | 0 | Reserved   |
| 1      | 1 | 1 | 1 | Reserved   |

- **SRAMSIZ: Internal SRAM Size**

| SRAMSIZ |   |   |   | Size       |
|---------|---|---|---|------------|
| 0       | 0 | 0 | 0 | Reserved   |
| 0       | 0 | 0 | 1 | 1K bytes   |
| 0       | 0 | 1 | 0 | 2K bytes   |
| 0       | 0 | 1 | 1 | Reserved   |
| 0       | 1 | 0 | 0 | Reserved   |
| 0       | 1 | 0 | 1 | 4K bytes   |
| 0       | 1 | 1 | 0 | Reserved   |
| 0       | 1 | 1 | 1 | Reserved   |
| 1       | 0 | 0 | 0 | 8K bytes   |
| 1       | 0 | 0 | 1 | 16K bytes  |
| 1       | 0 | 1 | 0 | 32K bytes  |
| 1       | 0 | 1 | 1 | 64K bytes  |
| 1       | 1 | 0 | 0 | 128K bytes |
| 1       | 1 | 0 | 1 | 256K bytes |
| 1       | 1 | 1 | 0 | 96K bytes  |
| 1       | 1 | 1 | 1 | 512K bytes |

- **ARCH: Architecture Identifier**

| ARCH |           | Architecture   |
|------|-----------|----------------|
| Hex  | Dec       |                |
| 0x40 | 0100 0000 | AT91x40 Series |
| 0x63 | 0110 0011 | AT91x63 Series |
| 0x55 | 0101 0101 | AT91x55 Series |
| 0x42 | 0100 0010 | AT91x42 Series |
| 0x92 | 1001 0010 | AT91x92 Series |
| 0x34 | 0011 0100 | AT91x34 Series |

- **NVPTYP: Nonvolatile Program Memory Type**

| NVPTYP |   |   | Memory                   |
|--------|---|---|--------------------------|
| 0      | 0 | 0 | ROM                      |
| 0      | 0 | 1 | ROMless or on-chip Flash |
| 1      | 0 | 0 | SRAM emulating ROM       |

- **EXT: Extension Flag**

0 = Chip ID has a single register definition without extension

1 = An extended Chip ID exists.

## Debug Unit Chip ID Extension Register

**Name:** DBGU\_EXID

**Access Type:** Read-only

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

- **EXID: Chip ID Extension**

Reads 0 if the bit EXT in DBGU\_CIDR is 0.

## Debug Unit Force NTRST Register

**Name:** DBGU\_FNR

**Access Type:** Read/Write

|    |    |    |    |    |    |    |        |
|----|----|----|----|----|----|----|--------|
| 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      |
| –  | –  | –  | –  | –  | –  | –  | FNTRST |

- **FNTRST: Force NTRST**

0 = NTRST of the ARM processor's TAP controller is driven by the NTRST pin.

1 = NTRST of the ARM processor's TAP controller is held low.

## Parallel Input/Output Controller (PIO)

### Overview

The Parallel Input/Output Controller (PIO) manages up to 32 fully programmable input/output lines. Each I/O line may be dedicated as a general-purpose I/O or be assigned to a function of an embedded peripheral. This assures effective optimization of the pins of a product.

Each I/O line is associated with a bit number in all of the 32-bit registers of the 32-bit wide User Interface.

Each I/O line of the PIO Controller features:

- An input change interrupt enabling level change on any I/O line.
- A glitch filter providing rejection of pulses lower than one-half of clock cycle.
- Multi-drive capability similar to an open drain I/O line.
- Control of the the pull-up of the I/O line.
- Input visibility and output control.

The PIO Controller also features a synchronous output providing up to 32 bits of data output in a single write operation.

Important features of the PIO also include:

- Up to 32 Programmable I/O Lines
- Fully Programmable through Set/Clear Registers
- Multiplexing of Two Peripheral Functions per I/O Line
- For each I/O Line (Whether Assigned to a Peripheral or Used as General Purpose I/O)
  - Input Change Interrupt
  - Glitch Filter
  - Multi-drive Option Enables Driving in Open Drain
  - Programmable Pull Up on Each I/O Line
  - Pin Data Status Register, Supplies Visibility of the Level on the Pin at Any Time
- Synchronous Output, Provides Set and Clear of Several I/O lines in a Single Write

# Block Diagram

Figure 147. Block Diagram

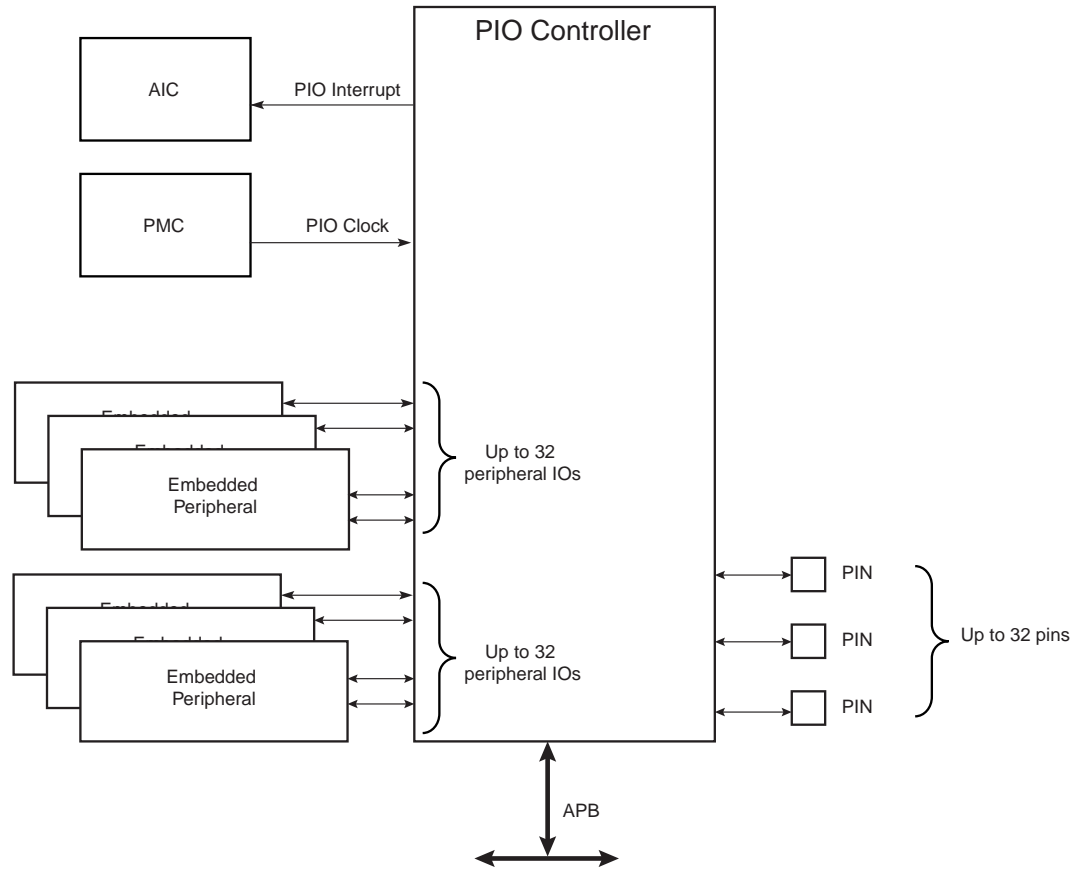
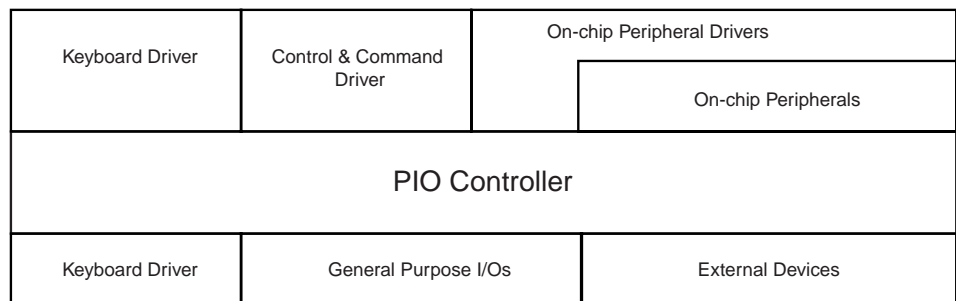


Figure 148. Application Block Diagram



## Product Dependencies

### Pin Multiplexing

Each pin is configurable, according to product definition as either a general-purpose I/O line only, or as an I/O line multiplexed with one or two peripheral I/Os. As the multiplexing is hardware-defined and thus product-dependent, the hardware designer and programmer must carefully determine the configuration of the PIO controllers required by their application. When an I/O line is general-purpose only, i.e. not multiplexed with any peripheral I/O, programming of the PIO Controller regarding the assignment to a peripheral has no effect and only the PIO Controller can control how the pin is driven by the product.

### External Interrupt Lines

The interrupt signals FIQ and IRQ0 to IRQn are most generally multiplexed through the PIO Controllers. However, it is not necessary to assign the I/O line to the interrupt function as the PIO Controller has no effect on inputs and the interrupt lines (FIQ or IRQs) are used only as inputs.

### Power Management

The Power Management Controller controls the PIO Controller clock in order to save power. Writing any of the registers of the user interface does not require the PIO Controller clock to be enabled. This means that the configuration of the I/O lines does not require the PIO Controller clock to be enabled.

However, when the clock is disabled, not all of the features of the PIO Controller are available. Note that the Input Change Interrupt and the read of the pin level require the clock to be validated.

After a hardware reset, the PIO clock is disabled by default (see Power Management Controller).

The user must configure the Power Management Controller before any access to the input line information.

### Interrupt Generation

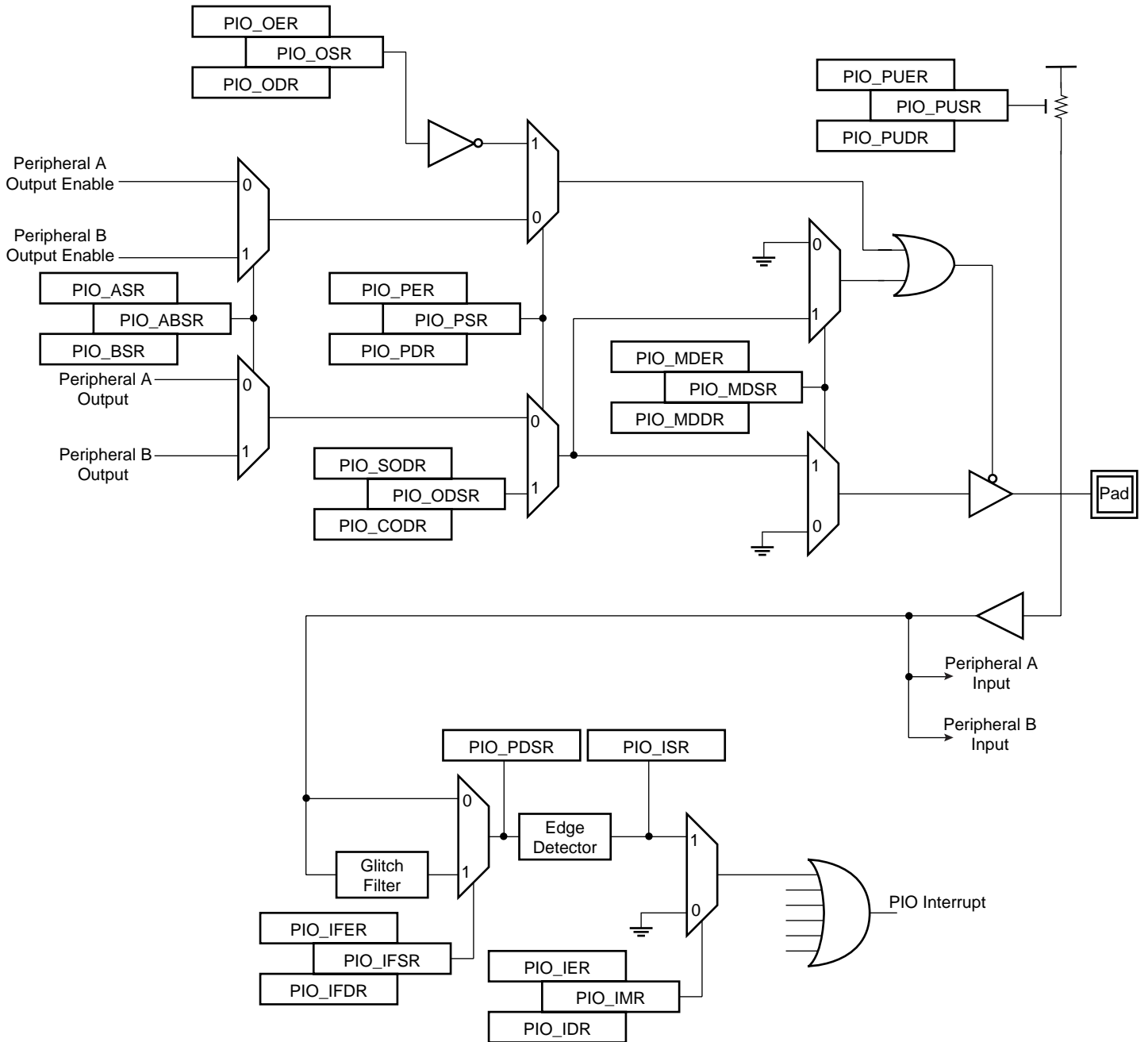
For interrupt handling, the PIO Controllers are considered as user peripherals. This means that the PIO Controller interrupt lines are connected among the interrupt sources 2 to 31. Refer to the PIO Controller peripheral identifier in the product description to identify the interrupt sources dedicated to the PIO Controllers.

The PIO Controller interrupt can be generated only if the PIO Controller clock is enabled.

# Functional Description

The PIO Controller features up to 32 fully-programmable I/O lines. Most of the control logic associated to each I/O is represented in Figure 149.

**Figure 149.** I/O Line Control Logic





## Pull-up Resistor Control

Each I/O line is designed with an embedded pull-up resistor. The value of this resistor is about 100 k $\Omega$  (see the product electrical characteristics for more details about this value). The pull-up resistor can be enabled or disabled by writing respectively PIO\_PUER (Pull-up Enable Register) and PIO\_PUDR (Pull-up Disable Register). Writing in these registers results in setting or clearing the corresponding bit in PIO\_PUSR (Pull-up Status Register). Reading a 1 in PIO\_PUSR means the pull-up is disabled and reading a 0 means the pull-up is enabled.

Control of the pull-up resistor is possible regardless of the configuration of the I/O line.

After reset, all of the pull-ups are enabled, i.e. PIO\_PUSR resets at the value 0x0.

## I/O Line or Peripheral Function Selection

When a pin is multiplexed with one or two peripheral functions, the selection is controlled with the registers PIO\_PER (PIO Enable Register) and PIO\_PDR (PIO Disable Register). The register PIO\_PSR (PIO Status Register) is the result of the set and clear registers and indicates whether the pin is controlled by the corresponding peripheral or by the PIO Controller. A value of 0 indicates that the pin is controlled by the corresponding on-chip peripheral selected in the PIO\_ABSR (AB Select Status Register). A value of 1 indicates the pin is controlled by the PIO controller.

If a pin is used as a general purpose I/O line (not multiplexed with an on-chip peripheral), PIO\_PER and PIO\_PDR have no effect and PIO\_PSR returns 1 for the corresponding bit.

After reset, most generally, the I/O lines are controlled by the PIO controller, i.e. PIO\_PSR resets at 1. However, in some events, it is important that PIO lines are controlled by the peripheral (as in the case of memory chip select lines that must be driven inactive after reset or for address lines that must be driven low for booting out of an external memory). Thus, the reset value of PIO\_PSR is defined at the product level, depending on the multiplexing of the device.

## Peripheral A or B Selection

The PIO Controller provides multiplexing of up to two peripheral functions on a single pin. The selection is performed by writing PIO\_ASR (A Select Register) and PIO\_BSR (Select B Register). PIO\_ABSR (AB Select Status Register) indicates which peripheral line is currently selected. For each pin, the corresponding bit at level 0 means peripheral A is selected whereas the corresponding bit at level 1 indicates that peripheral B is selected.

Note that multiplexing of peripheral lines A and B only affects the output line. The peripheral input lines are always connected to the pin input.

After reset, PIO\_ABSR is 0, thus indicating that all the PIO lines are configured on peripheral A. However, peripheral A generally does not drive the pin as the PIO Controller resets in I/O line mode.

Writing in PIO\_ASR and PIO\_BSR manages PIO\_ABSR regardless of the configuration of the pin. However, assignment of a pin to a peripheral function requires a write in the corresponding peripheral selection register (PIO\_ASR or PIO\_BSR) in addition to a write in PIO\_PDR.

## Output Control

When the I/O line is assigned to a peripheral function, i.e. the corresponding bit in PIO\_PSR is at 0, the drive of the I/O line is controlled by the peripheral. Peripheral A or B, depending on the value in PIO\_ABSR, determines whether the pin is driven or not.

When the I/O line is controlled by the PIO controller, the pin can be configured to be driven. This is done by writing PIO\_OER (Output Enable Register) and PIO\_PDR (Output Disable Register). The results of these write operations are detected in PIO\_OSR (Output Status Register). When a bit in this register is at 0, the corresponding I/O line is used as an input only. When the bit is at 1, the corresponding I/O line is driven by the PIO controller.

The level driven on an I/O line can be determined by writing in PIO\_SODR (Set Output Data Register) and PIO\_CODR (Clear Output Data Register). These write operations respectively set and clear PIO\_ODSR (Output Data Status Register), which represents the data driven on the I/O lines. Writing in PIO\_OER and PIO\_ODR manages PIO\_OSR whether the pin is configured to be controlled by the PIO controller or assigned to a peripheral function. This enables configuration of the I/O line prior to setting it to be managed by the PIO Controller.

Similarly, writing in PIO\_SODR and PIO\_CODR effects PIO\_ODSR. This is important as it defines the first level driven on the I/O line.

## Synchronous Data Output

Using the write operations in PIO\_SODR and PIO\_CODR can require that several instructions be executed in order to define values on several bits. Both clearing and setting I/O lines on an 8-bit port, for example, cannot be done at the same time, and thus might limit the application covered by the PIO Controller.

To avoid these inconveniences, the PIO Controller features a Synchronous Data Output to clear and set a number of I/O lines in a single write. This is performed by authorizing the writing of PIO\_ODSR (Output Data Status Register) from the register set PIO\_OWER (Output Write Enable Register), PIO\_OWDR (Output Write Disable Register) and PIO\_OWSR (Output Write Status Register). The value of PIO\_OWSR register is user-definable by writing in PIO\_OWER and PIO\_OWDR. It is used by the PIO Controller as a PIO\_ODSR write authorization mask. Authorizing the write of PIO\_ODSR on a user-definable number of bits is especially useful, as it guarantees that the unauthorized bit will not be changed when writing it and thus avoids the need of a time consuming read-modify-write operation.

After reset, the synchronous data output is disabled on all the I/O lines as PIO\_OWSR resets at 0x0.

## Multi Drive Control (Open Drain)

Each I/O can be independently programmed in Open Drain by using the Multi Drive feature. This feature permits several drivers to be connected on the I/O line which is driven low only by each device. An external pull-up resistor (or enabling of the internal one) is generally required to guarantee a high level on the line.

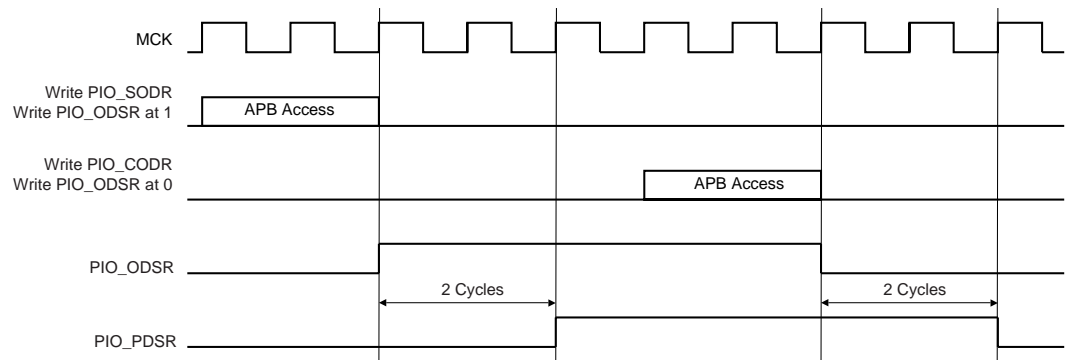
The Multi Drive feature is controlled by PIO\_MDER (Multi-driver Enable Register) and PIO\_MDDR (Multi-driver Disable Register). The Multi Drive can be selected whether the I/O line is controlled by the PIO controller or assigned to a peripheral function. PIO\_MDSR (Multi-driver Status Register) indicates the pins that are configured to support external drivers.

After reset, the Multi Drive feature is disabled on all pins, i.e. PIO\_MDSR resets at value 0x0.

## Output Line Timings

Figure 150 shows how the outputs are driven either by writing PIO\_SODR or PIO\_CODR, or by directly writing PIO\_ODSR. This last case is valid only if the corresponding bit in PIO\_OWSR is set. Figure 150 also shows when the feedback in PIO\_PDSR is available.

**Figure 150.** Output Line Timings



**Inputs**

The level on each I/O line can be read through PIO\_PDSR (Peripheral Data Status Register). This register indicates the level of the I/O lines regardless of their configuration, whether uniquely as an input or driven by the PIO controller or driven by a peripheral.

Reading the I/O line levels requires the clock of the PIO controller to be enabled, otherwise PIO\_PDSR reads the levels present on the I/O line at the time the clock was disabled.

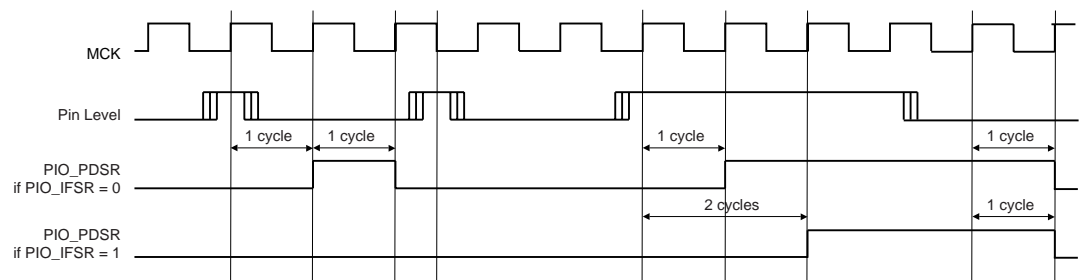
**Input Glitch Filtering**

Optional input glitch filters are independently programmable on each I/O line. When the glitch filter is enabled, a glitch with a duration of less than 1/2 Master Clock (MCK) cycle is automatically rejected, while a pulse with a duration of 1 Master Clock cycle or more is accepted. For pulse durations between 1/2 Master Clock cycle and 1 Master Clock cycle the pulse may or may not be taken into account, depending on the precise timing of its occurrence. Thus for a pulse to be visible it must exceed 1 Master Clock cycle, whereas for a glitch to be reliably filtered out, its duration must not exceed 1/2 Master Clock cycle. The filter introduces one Master Clock cycle latency if the pin level change occurs before a rising edge. However, this latency does not appear if the pin level change occurs before a falling edge. This is illustrated in Figure 151.

The glitch filters are controlled by the register set; PIO\_IFER (Input Filter Enable Register), PIO\_IFDR (Input Filter Disable Register) and PIO\_IFSR (Input Filter Status Register). Writing PIO\_IFER and PIO\_IFDR respectively sets and clears bits in PIO\_IFSR. This last register enables the glitch filter on the I/O lines.

When the glitch filter is enabled, it does not modify the behavior of the inputs on the peripherals. It acts only on the value read in PIO\_PDSR and on the input change interrupt detection. The glitch filters require that the PIO Controller clock is enabled.

**Figure 151.** Input Glitch Filter Timing



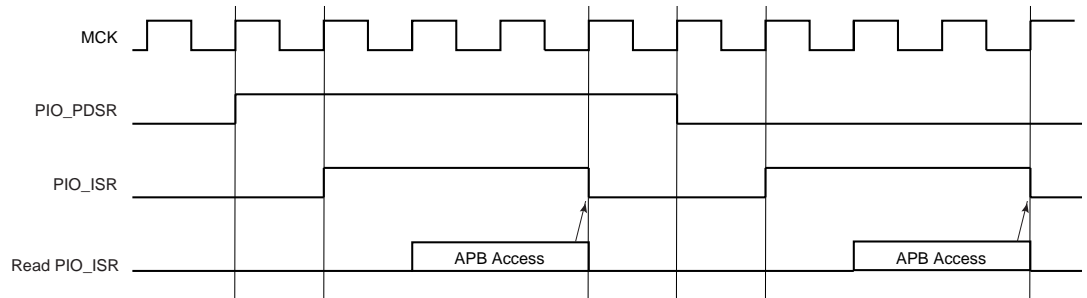
## Input Change Interrupt

The PIO Controller can be programmed to generate an interrupt when it detects an input change on an I/O line. The Input Change Interrupt is controlled by writing PIO\_IER (Interrupt Enable Register) and PIO\_IDR (Interrupt Disable Register), which respectively enable and disable the input change interrupt by setting and clearing the corresponding bit in PIO\_IMR (Interrupt Mask Register). As Input change detection is possible only by comparing two successive samplings of the input of the I/O line, the PIO Controller clock must be enabled. The Input Change Interrupt is available, regardless of the configuration of the I/O line, i.e. configured as an input only, controlled by the PIO Controller or assigned to a peripheral function.

When an input change is detected on an I/O line, the corresponding bit in PIO\_ISR (Interrupt Status Register) is set. If the corresponding bit in PIO\_IMR is set, the PIO Controller interrupt line is asserted. The interrupt signals of the thirty-two channels are ORed-wired together to generate a single interrupt signal to the Advanced Interrupt Controller.

When the software reads PIO\_ISR, all the interrupts are automatically cleared. This signifies that all the interrupts that are pending when PIO\_ISR is read must be handled.

**Figure 152.** Input Change Interrupt Timings



## I/O Lines Programming Example

The programming example shown in Table 66 below is used to define the following configuration.

- 4-bit output port on I/O lines 0 to 3, (should be written in a single write operation), open-drain, with pull-up resistor
- Four output signals on I/O lines 4 to 7 (to drive LEDs for example), driven high and low, no pull-up resistor
- Four input signals on I/O lines 8 to 11 (to read push-button states for example), with pull-up resistors, glitch filters and input change interrupts
- Four input signals on I/O line 12 to 15 to read an external device status (polled, thus no input change interrupt), no pull-up resistor, no glitch filter
- I/O lines 16 to 19 assigned to peripheral A functions with pull-up resistor
- I/O lines 20 to 23 assigned to peripheral B functions, no pull-up resistor
- I/O lines 24 to 27 assigned to peripheral A with Input Change Interrupt and pull-up resistor

**Table 66.** Programming Example

| Register | Value to be Written |
|----------|---------------------|
| PIO_PER  | 0x0000 FFFF         |
| PIO_PDR  | 0x0FFF 0000         |
| PIO_OER  | 0x0000 00FF         |
| PIO_ODR  | 0x0FFF FF00         |
| PIO_IFER | 0x0000 0F00         |
| PIO_IFDR | 0x0FFF F0FF         |
| PIO_SODR | 0x0000 0000         |
| PIO_CODR | 0x0FFF FFFF         |
| PIO_IER  | 0x0F00 0F00         |
| PIO_IDR  | 0x00FF F0FF         |
| PIO_MDER | 0x0000 000F         |
| PIO_MDDR | 0x0FFF FFF0         |
| PIO_PUDR | 0x00F0 00F0         |
| PIO_PUER | 0x0F0F FF0F         |
| PIO_ASR  | 0x0F0F 0000         |
| PIO_BSR  | 0x00F0 0000         |
| PIO_OWER | 0x0000 000F         |
| PIO_OWDR | 0x0FFF FFF0         |

## Parallel Input/Output Controller (PIO) User Interface

Each I/O line controlled by the PIO Controller is associated with a bit in each of the PIO Controller User Interface registers. Each register is 32 bits wide. If a parallel I/O line is not defined, writing to the corresponding bits has no effect. Undefined bits read zero. If the I/O line is not multiplexed with any peripheral, the I/O line is controlled by the PIO Controller and PIO\_PSR returns 1 systematically.

**Table 67.** PIO Register Mapping

| Offset | Register                                       | Name     | Access     | Reset Value |
|--------|--|----------|------------|-------------|
| 0x0000 | PIO Enable Register                            | PIO_PER  | Write-only | –           |
| 0x0004 | PIO Disable Register                           | PIO_PDR  | Write-only | –           |
| 0x0008 | PIO Status Register <sup>(1)</sup>             | PIO_PSR  | Read-only  | 0x0000 0000 |
| 0x000C | Reserved                                       |          |            |             |
| 0x0010 | PIO Output Enable Register                     | PIO_OER  | Write-only | –           |
| 0x0014 | PIO Output Disable Register                    | PIO_ODR  | Write-only | –           |
| 0x0018 | PIO Output Status Register                     | PIO_OSR  | Read-only  | 0x0000 0000 |
| 0x001C | Reserved                                       |          |            |             |
| 0x0020 | PIO Glitch Input Filter Enable Register        | PIO_IFER | Write-only | –           |
| 0x0024 | PIO Glitch Input Filter Disable Register       | PIO_IFDR | Write-only | –           |
| 0x0028 | PIO Glitch Input Filter Status Register        | PIO_IFSR | Read-only  | 0x0000 0000 |
| 0x002C | Reserved                                       |          |            |             |
| 0x0030 | PIO Set Output Data Register                   | PIO_SODR | Write-only | –           |
| 0x0034 | PIO Clear Output Data Register                 | PIO_CODR | Write-only | –           |
| 0x0038 | PIO Output Data Status Register <sup>(2)</sup> | PIO_ODSR | Read-only  | 0x0000 0000 |
| 0x003C | PIO Pin Data Status Register <sup>(3)</sup>    | PIO_PDSR | Read-only  |             |
| 0x0040 | PIO Interrupt Enable Register                  | PIO_IER  | Write-only | –           |
| 0x0044 | PIO Interrupt Disable Register                 | PIO_IDR  | Write-only | –           |
| 0x0048 | PIO Interrupt Mask Register                    | PIO_IMR  | Read-only  | 0x0000 0000 |
| 0x004C | PIO Interrupt Status Register <sup>(4)</sup>   | PIO_ISR  | Read-only  | 0x0000 0000 |
| 0x0050 | PIO Multi-driver Enable Register               | PIO_MDER | Write-only | –           |
| 0x0054 | PIO Multi-driver Disable Register              | PIO_MDDR | Write-only | –           |
| 0x0058 | PIO Multi-driver Status Register               | PIO_MDSR | Read-only  | 0x0000 0000 |
| 0x005C | Reserved                                       |          |            |             |
| 0x0060 | PIO Pull-up Disable Register                   | PIO_PUDR | Write-only | –           |
| 0x0064 | PIO Pull-up Enable Register                    | PIO_PUER | Write-only | –           |
| 0x0068 | PIO Pad Pull-up Status Register                | PIO_PUSR | Read-only  | 0x0000 0000 |
| 0x006C | Reserved                                       |          |            |             |

**Table 67.** PIO Register Mapping (Continued)

| Offset                 | Register  | Name     | Access     | Reset Value |
|------------------------|---|----------|------------|-------------|
| 0x0070                 | PIO Peripheral A Select Register <sup>(5)</sup> | PIO_ASR  | Write-only | –           |
| 0x0074                 | PIO Peripheral B Select Register <sup>(5)</sup> | PIO_BSR  | Write-only | –           |
| 0x0078                 | PIO AB Status Register <sup>(5)</sup>           | PIO_ABSR | Read-only  | 0x0000 0000 |
| 0x007C<br>to<br>0x009C | Reserved  |          |            |             |
| 0x00A0                 | PIO Output Write Enable                         | PIO_OWER | Write-only | –           |
| 0x00A4                 | PIO Output Write Disable                        | PIO_OWDR | Write-only | –           |
| 0x00A8                 | PIO Output Write Status Register                | PIO_OWSR | Read-only  | 0x0000 0000 |
| 0x00AC                 | Reserved  |          |            |             |

- Notes:
1. Reset value of PIO\_PSR depends on the product implementation.
  2. PIO\_ODSR is Read-only or Read/Write depending on PIO\_OWSR I/O lines.
  3. Reset value of PIO\_PDSR depends on the level of the I/O lines.
  4. PIO\_ISR is reset at 0x0. However, the first read of the register may read a different value as input changes may have occurred.
  5. Only this set of registers clears the status by writing 1 in the first register and sets the status by writing 1 in the second register.

## PIO Enable Register

**Name:** PIO\_PER

**Access Type:** Write-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: PIO Enable**

0 = No effect.

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

## PIO Disable Register

**Name:** PIO\_PDR

**Access Type:** Write-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: PIO Disable**

0 = No effect.

1 = Disables the PIO from controlling the corresponding pin (enables peripheral control of the pin).



## PIO Status Register

**Name:** PIO\_PSR

**Access Type:** Read-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: PIO Status**

0 = PIO is inactive on the corresponding I/O line (peripheral is active).

1 = PIO is active on the corresponding I/O line (peripheral is inactive).

## PIO Output Enable Register

**Name:** PIO\_OER

**Access Type:** Write-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Output Enable**

0 = No effect.

1 = Enables the output on the I/O line.

## PIO Output Disable Register

**Name:** PIO\_ODR

**Access Type:** Write-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Output Disable**

0 = No effect.

1 = Disables the output on the I/O line.

## PIO Output Status Register

**Name:** PIO\_OSR

**Access Type:** Read-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Output Status**

0 = The I/O line is a pure input.

1 = The I/O line is enabled in output.

## PIO Input Filter Enable Register

**Name:** PIO\_IFER

**Access Type:** Write-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Input Filter Enable**

0 = No effect.

1 = Enables the input glitch filter on the I/O line.

## PIO Input Filter Disable Register

**Name:** PIO\_IFDR

**Access Type:** Write-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Input Filter Disable**

0 = No effect.

1 = Disables the input glitch filter on the I/O line.

## PIO Input Filter Status Register

**Name:** PIO\_IFSR

**Access Type:** Read-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Input Filter Status**

0 = The input glitch filter is disabled on the I/O line.

1 = The input glitch filter is enabled on the I/O line.

## PIO Set Output Data Register

**Name:** PIO\_SODR

**Access Type:** Write-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Set Output Data**

0 = No effect.

1 = Sets the data to be driven on the I/O line.

## PIO Clear Output Data Register

**Name:** PIO\_CODR

**Access Type:** Write-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Set Output Data**

0 = No effect.

1 = Clears the data to be driven on the I/O line.

## PIO Output Data Status Register

**Name:** PIO\_ODSR

**Access Type:** Read-only or Read/Write

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Output Data Status**

0 = The data to be driven on the I/O line is 0.

1 = The data to be driven on the I/O line is 1.

## PIO Pin Data Status Register

**Name:** PIO\_PDSR

**Access Type:** Read-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Output Data Status**

0 = The I/O line is at level 0.

1 = The I/O line is at level 1.

## PIO Interrupt Enable Register

**Name:** PIO\_IER

**Access Type:** Write-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Input Change Interrupt Enable**

0 = No effect.

1 = Enables the Input Change Interrupt on the I/O line.

**PIO Interrupt Disable Register**

**Name:** PIO\_IDR

**Access Type:** Write-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

• **P0 - P31: Input Change Interrupt Disable**

0 = No effect.

1 = Disables the Input Change Interrupt on the I/O line.

**PIO Interrupt Mask Register**

**Name:** PIO\_IMR

**Access Type:** Read-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

• **P0 - P31: Input Change Interrupt Mask**

0 = Input Change Interrupt is disabled on the I/O line.

1 = Input Change Interrupt is enabled on the I/O line.

## PIO Interrupt Status Register

**Name:** PIO\_IMR

**Access Type:** Read-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Input Change Interrupt Mask**

0 = No Input Change has been detected on the I/O line since PIO\_ISR was last read or since reset.

1 = At least one Input Change has been detected on the I/O line since PIO\_ISR was last read or since reset.

## PIO Multi-driver Enable Register

**Name:** PIO\_MDER

**Access Type:** Write-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Multi Drive Enable**

0 = No effect.

1 = Enables Multi Drive on the I/O line.



## PIO Multi-driver Disable Register

**Name:** PIO\_MDDR

**Access Type:** Write-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Multi Drive Disable**

0 = No effect.

1 = Disables Multi Drive on the I/O line.

## PIO Multi-driver Status Register

**Name:** PIO\_MDSR

**Access Type:** Read-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Multi Drive Status**

0 = The Multi Drive is disabled on the I/O line. The pin is driven at high and low level.

1 = The Multi Drive is enabled on the I/O line. The pin is driven at low level only.



## PIO Pull Up Disable Register

Name: PIO\_PUDR

Access Type: Write-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Pull Up Disable**

0 = No effect.

1 = Disables the pull up resistor on the I/O line.

## PIO Pull Up Enable Register

Name: PIO\_PUER

Access Type: Write-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Pull Up Enable**

0 = No effect.

1 = Enables the pull up resistor on the I/O line.

**PIO Pad Pull Up Status Register**

Name: PIO\_PUSR

Access Type: Read-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

• **P0 - P31: Pull Up Status**

0 = Pull Up resistor is enabled on the I/O line.

1 = Pull Up resistor is disabled on the I/O line.

**PIO Peripheral A Select Register**

Name: PIO\_AS

Access Type: Write-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

• **P0 - P31: Peripheral A Select**

0 = No effect.

1 = Assigns the I/O line to the Peripheral A function.

## PIO Peripheral B Select Register

**Name:** PIO\_BSR

**Access Type:** Write-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Peripheral B Select**

0 = No effect.

1 = Assigns the I/O line to the peripheral B function.

## PIO Peripheral AB Status Register

**Name:** PIO\_ABSR

**Access Type:** Read-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Peripheral A B Status**

0 = The I/O line is assigned to the Peripheral A.

1 = The I/O line is assigned to the Peripheral B.

## PIO Output Write Enable Register

**Name:** PIO\_OWER

**Access Type:** Write-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Output Write Enable**

0 = No effect.

1 = Enables writing PIO\_ODSR for the I/O line.

## PIO Output Write Disable Register

**Name:** PIO\_OWDR

**Access Type:** Write-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Output Write Disable**

0 = No effect.

1 = Disables writing PIO\_ODSR for the I/O line.



## PIO Output Write Status Register

Name: PIO\_OWSR

Access Type: Read-only

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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  |

- **P0 - P31: Output Write Status**

0 = Writing PIO\_ODSR does not affect the I/O line.

1 = Writing PIO\_ODSR affects the I/O line.

## Serial Peripheral Interface (SPI)

### Overview

The Serial Peripheral Interface (SPI) circuit is a synchronous serial data link that provides communication with external devices in Master or Slave Mode. It also allows communication between processors if an external processor is connected to the system.

The Serial Peripheral Interface is a shift register that serially transmits data bits to other SPIs. During a data transfer, one SPI system acts as the master that controls the data flow, while the other system acts as the slave, having data shifted into and out of it by the master. Different CPUs can take turn being masters (Multiple Master Protocol versus Single Master Protocol where one CPU is always the master while all of the others are always slaves), and one master may simultaneously shift data into multiple slaves. However, only one slave may drive its output to write data back to the master at any given time.

A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

The SPI system consists of two data lines and two control lines:

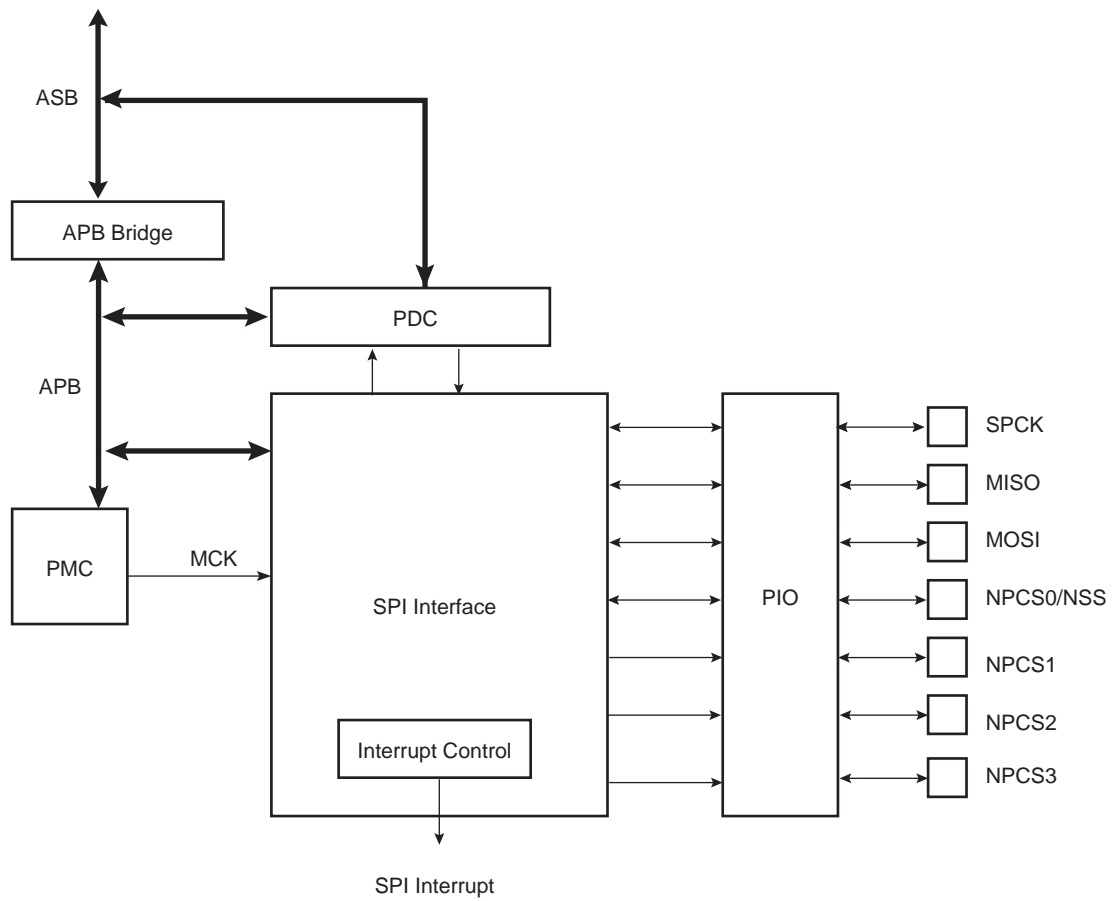
- Master Out Slave In (MOSI): This data line supplies the output data from the master shifted into the input(s) of the slave(s).
- Master In Slave Out (MISO): This data line supplies the output data from a slave to the input of the master. There may be no more than one slave transmitting data during any particular transfer.
- Serial Clock (SPCK): This control line is driven by the master and regulates the flow of the data bits. The master may transmit data at a variety of baud rates; the SPCK line cycles once for each bit that is transmitted.
- Slave Select (NSS): This control line allows slaves to be turned on and off by hardware.

The main features of the SPI are:

- Supports Communication with Serial External Devices
  - 4 Chip Selects with External Decoder Support Allow Communication with Up to 15 Peripherals
  - Serial Memories, such as DataFlash and 3-wire EEPROMs
  - Serial Peripherals, such as ADCs, DACs, LCD Controllers, CAN Controllers and Sensors
  - External Co-processors
- Master or Slave Serial Peripheral Bus Interface
  - 8- to 16-bit Programmable Data Length Per Chip Select
  - Programmable Phase and Polarity Per Chip Select
  - Programmable Transfer Delays Between Consecutive Transfers and Between Clock and Data Per Chip Select
  - Programmable Delay Between Consecutive Transfers
  - Selectable Mode Fault Detection
- Connection to PDC Channel Capabilities Optimizes Data Transfers
  - One Channel for the Receiver, One Channel for the Transmitter
  - Next Buffer Support

# Block Diagram

Figure 153. Block Diagram





## Application Block Diagram

Figure 154. Application Block Diagram: Single Master/Multiple Slave Implementation

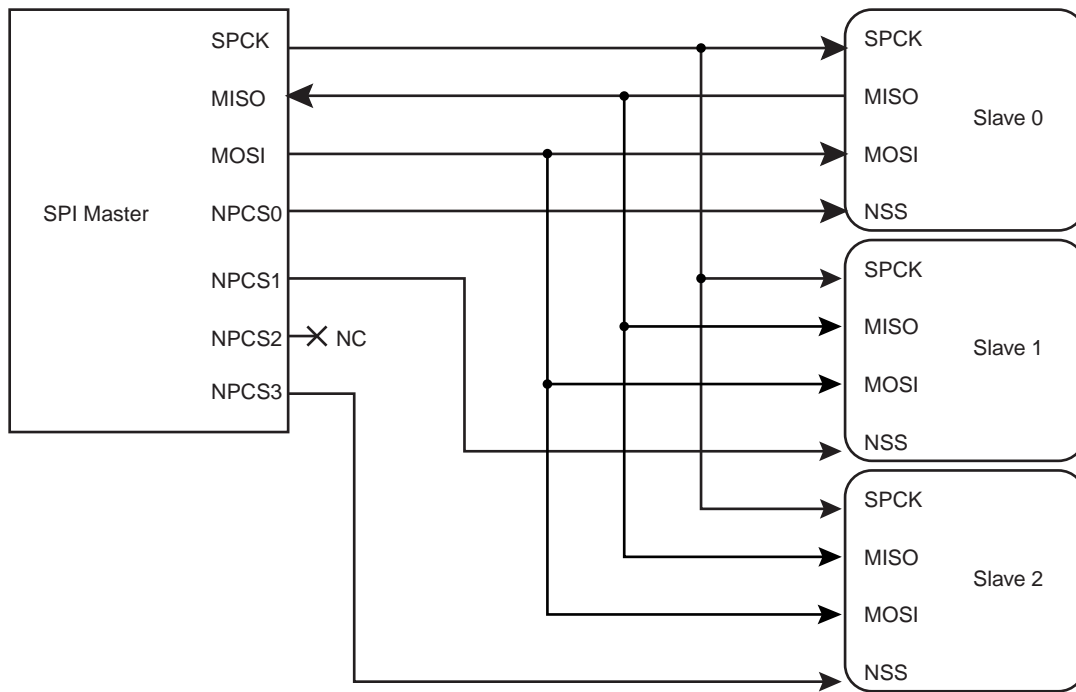


Table 68. Signal Description

| Pin Name    | Pin Description                     | Type   |        |
|-------------|-------------------------------------|--------|--------|
|             |                                     | Master | Slave  |
| MISO        | Master In Slave Out                 | Input  | Output |
| MOSI        | Master Out Slave In                 | Output | Input  |
| SPCK        | Serial Clock                        | Output | Input  |
| NPCS1-NPCS3 | Peripheral Chip Selects             | Output | Unused |
| NPCS0/NSS   | Peripheral Chip Select/Slave Select | Output | Input  |

## Product Dependencies

|                         |   |
|-------------------------|---|
| <b>I/O Lines</b>        | The pins used for interfacing the compliant external devices may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the SPI pins to their peripheral functions. |
| <b>Power Management</b> | The SPI may be clocked through the Power Management Controller (PMC), thus the programmer must first have to configure the PMC to enable the SPI clock.   |
| <b>Interrupt</b>        | The SPI interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling the SPI interrupt requires programming the AIC before configuring the SPI.                           |

## Functional Description

### Master Mode Operations

When configured in Master Mode, the Serial Peripheral Interface 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 core writes to the SPI\_TDR (Transmit Data Register).

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 SPI\_TDR, the SPI continues to transfer data. If the SPI\_RDR (Receive Data Register) has not been read before new data is received, the Overrun Error (OVRES) flag is set.

Note: As long as this flag is set, no data is loaded in the SPI\_RDR. The user has to read the status register to clear it.

The programmable 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 SPI\_CSR0 to SPI\_CSR3 (Chip Select Registers).

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

Figure 159 and Figure 160 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 in Section .

### Fixed Peripheral Select

This mode is used 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 SPI\_MR (Mode Register). The peripheral is defined by the PCS field in SPI\_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 SPI\_TDR 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 SPI\_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 SPI\_MR (Mode Register) selects one to four 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 SPI\_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 SPI\_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 SPI\_MR before writing new data in SPI\_TDR.

The value on the NPCS pins at the end of each transfer can be read in the SPI\_RDR (Receive Data Register).

By default, all NPCS signals are high (equal to one) before and after each transfer.

## Clock Generation and Transfer Delays

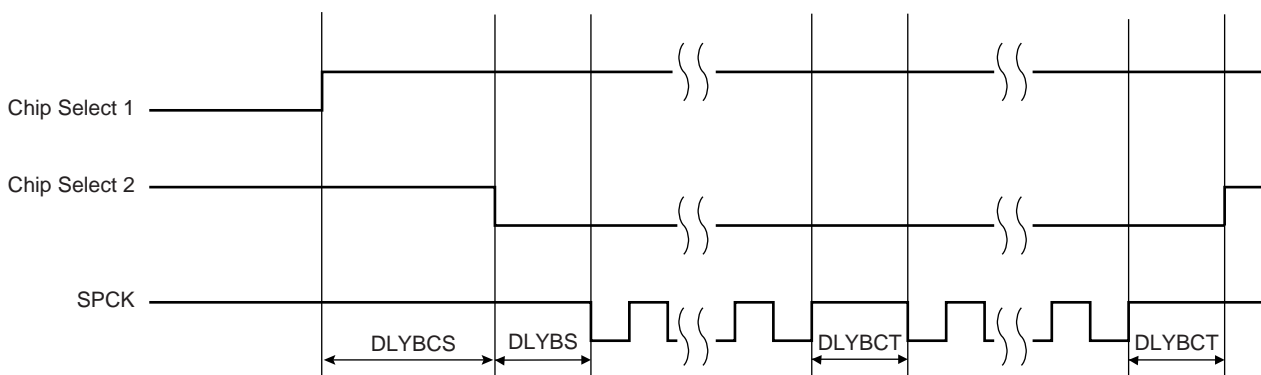
The SPI Baud rate clock is generated by dividing the Master Clock (MCK) or the Master Clock divided by 32 (if DIV32 is set in the Mode Register) by a value between 4 and 510. The divisor is defined in the SCBR field in each Chip Select Register. The transfer speed can thus be defined independently for each chip select signal.

Figure 155 shows a chip select transfer change and consecutive transfers on the same chip selects. Three delays can be programmed to modify the transfer waveforms:

- Delay between chip selects, programmable only once for all the chip selects by writing the field DLYBCS in the Mode Register. Allows insertion of a delay between release of one chip select and before assertion of a new one.
- Delay before SPCK, independently programmable for each chip select by writing the field DLYBS. Allows the start of SPCK to be delayed until after the chip select has been asserted.
- Delay between consecutive transfers, independently programmable for each chip select by writing the field DLYBCT. Allows insertion of a delay between two transfers occurring on the same chip select

These delays allow the SPI to be adapted to the interfaced peripherals and their speed and bus release time.

**Figure 155.** Programmable Delays



### **Mode Fault Detection**

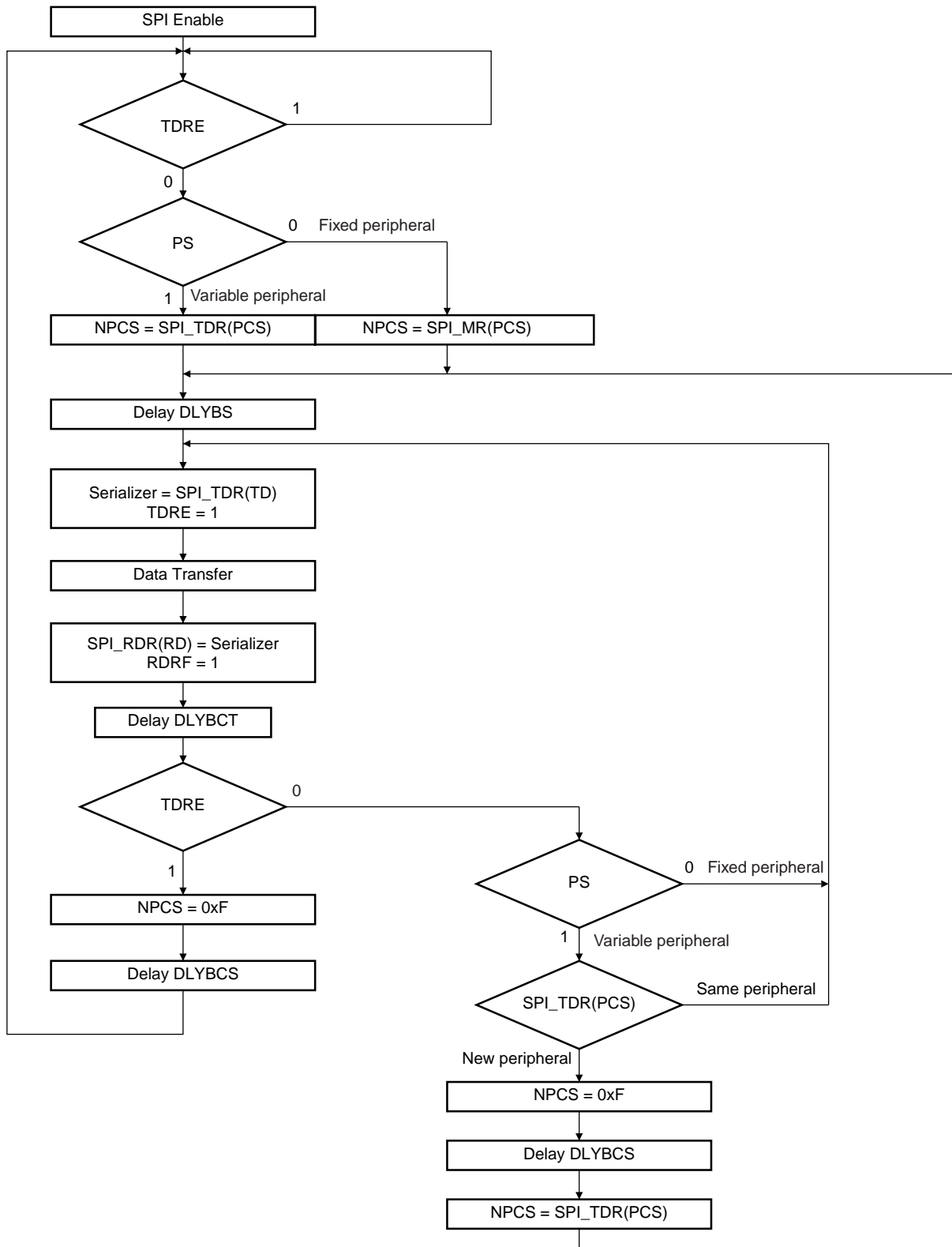
A mode fault is detected when the SPI is programmed in Master Mode and a low level is driven by an external master on the NPCS0/NSS signal.

When a mode fault is detected, the MODF bit in the SPI\_SR is set until the SPI\_SR is read and the SPI is disabled until re-enabled by bit SPIEN in the SPI\_CR (Control Register).

By default, Mode Fault Detection is enabled. It is disabled by setting the MODFDIS bit in the SPI Mode Register.

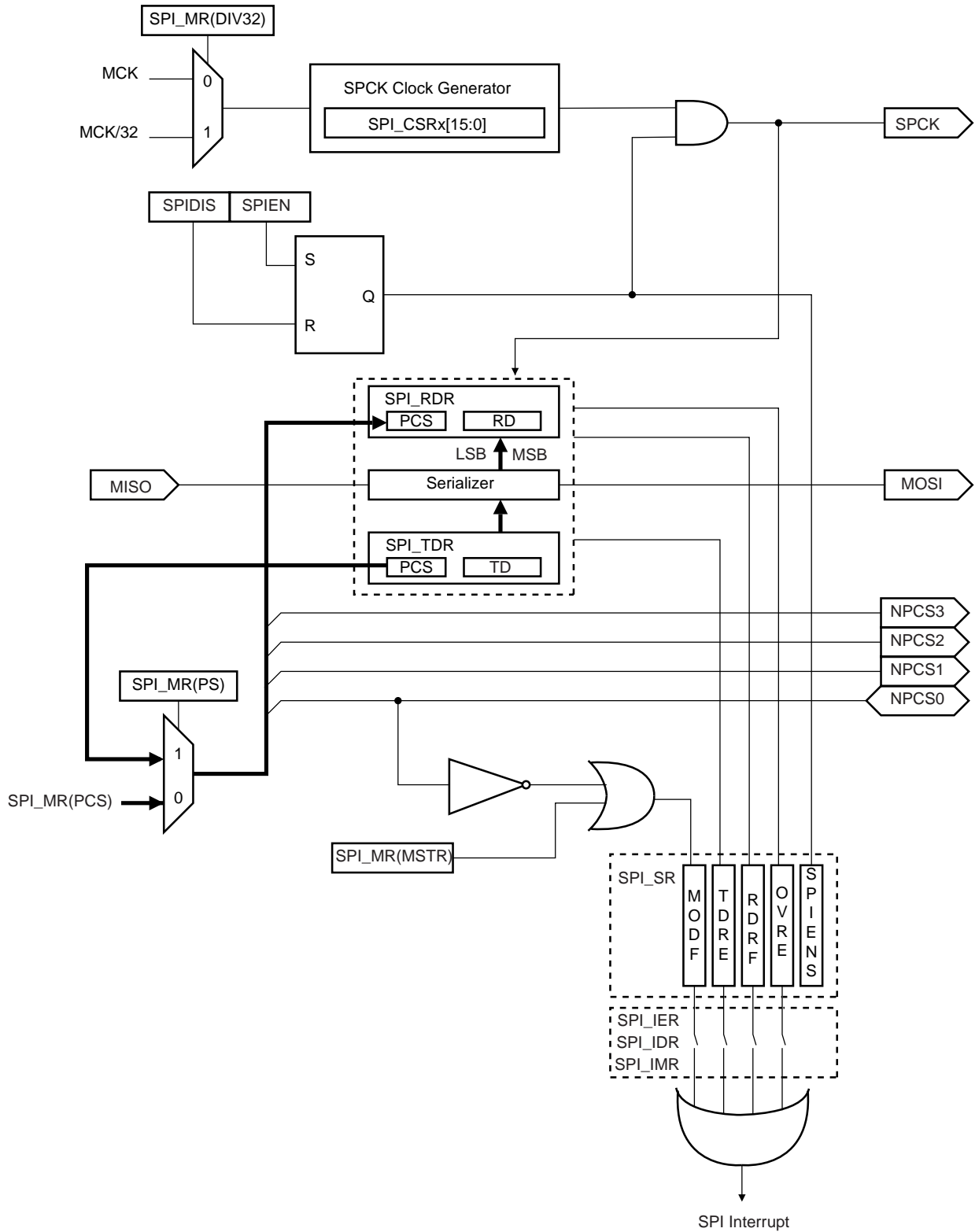
Master Mode Flow Diagram

Figure 156. Master Mode Flow Diagram



## Master Mode Block Diagram

Figure 157. Master Mode Block Diagram



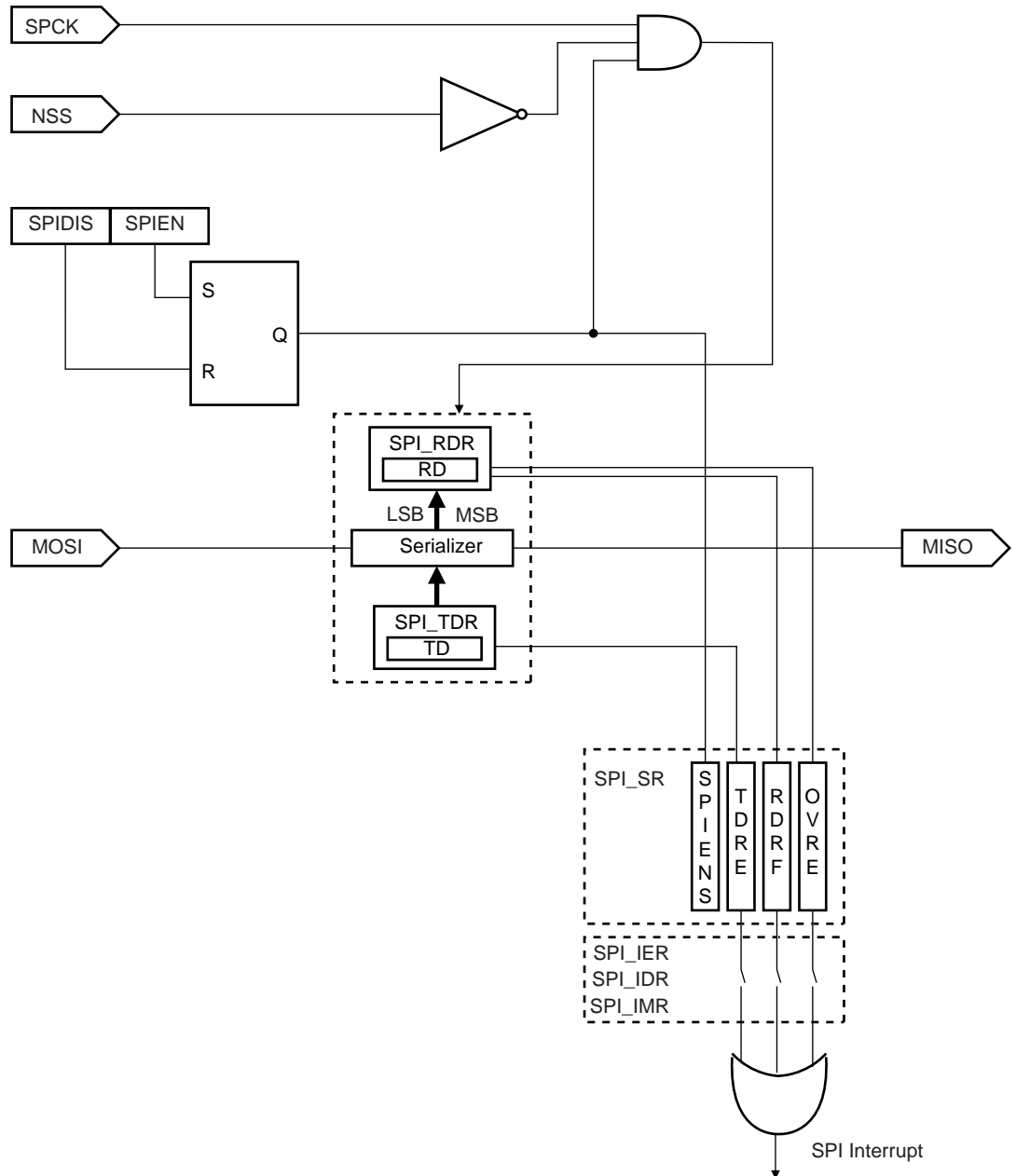
SPI Slave Mode

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 SPI\_CSR0 are used to define the transfer characteristics. The other Chip Select Registers are not used in Slave Mode.

In Slave Mode, the low and high pulse durations of the input clock on SPCK must be longer than two Master Clock periods.

Figure 158. Slave Mode Block Diagram



## Data Transfer

Four modes are used for data transfers. These modes correspond to combinations of a pair of parameters called clock polarity (CPOL) and clock phase (NCPHA) that determine the edges of the clock signal on which the data are driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations that are incompatible with one another. Thus a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are used and fixed in different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

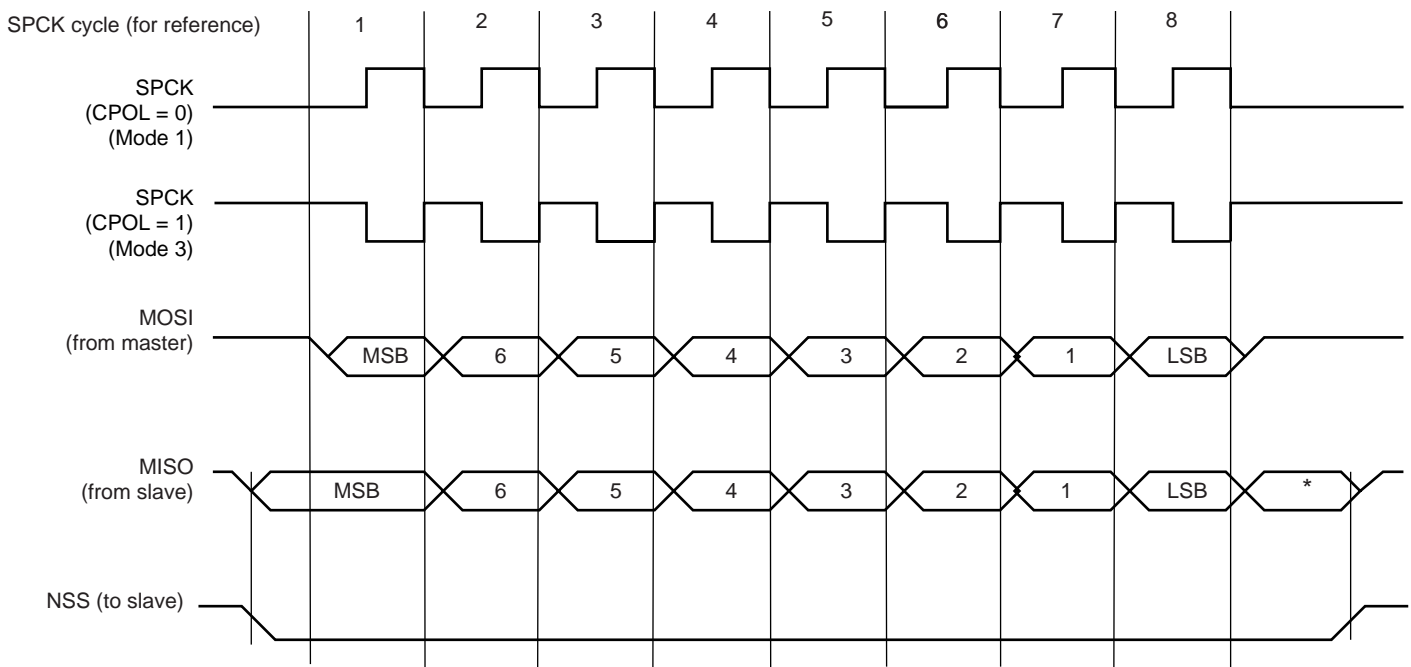
Table 69 shows the four modes and corresponding parameter settings.

**Table 69.** SPI Bus Protocol Mode

| SPI Mode | CPOL | NCPHA |
|----------|------|-------|
| 0        | 0    | 0     |
| 1        | 0    | 1     |
| 2        | 1    | 0     |
| 3        | 1    | 1     |

Figure 159 and Figure 160 show examples of data transfers.

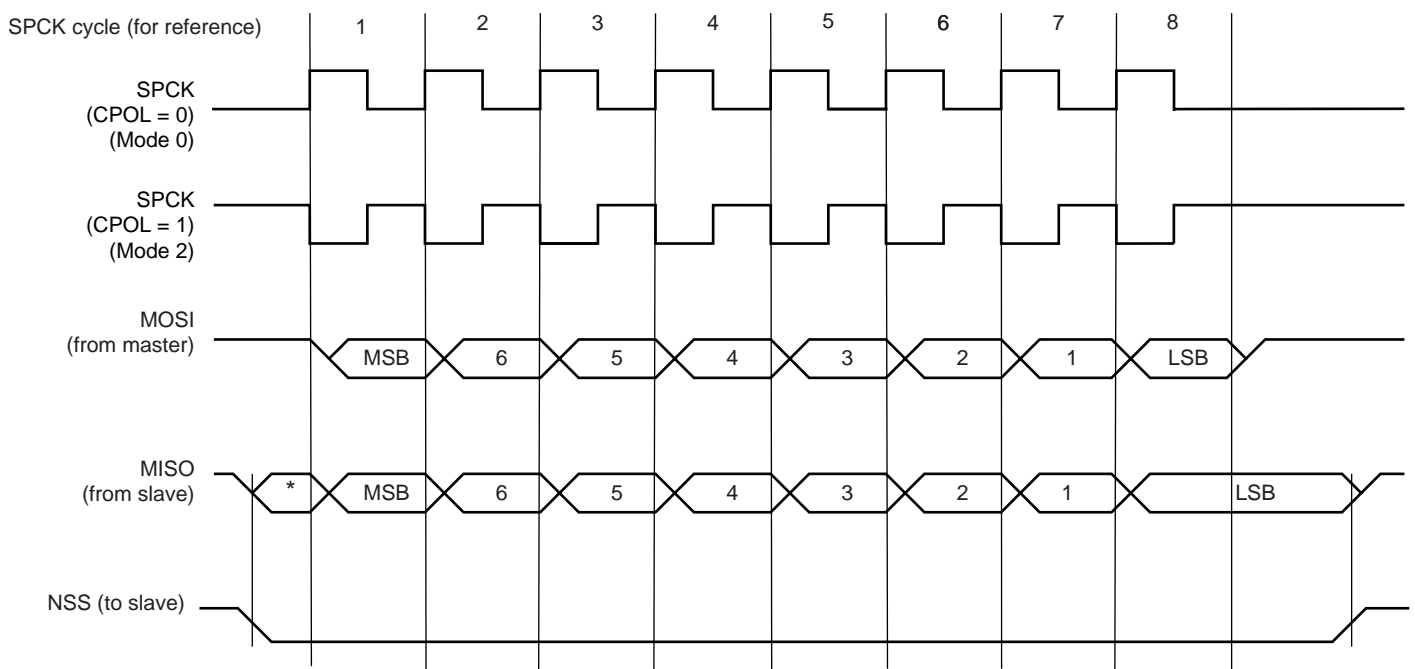
**Figure 159.** SPI Transfer Format (NCPHA = 1, 8 bits per transfer)



\* Not defined, but normally MSB of previous character received.



**Figure 160.** SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



\* Not defined but normally LSB of previous character transmitted.

## Serial Peripheral Interface (SPI) User Interface

**Table 70.** SPI Register Mapping

| Offset        | Register                   | Register Name | Access     | Reset      |
|---------------|----------------------------|---------------|------------|------------|
| 0x00          | Control Register           | SPI_CR        | Write-only | ---        |
| 0x04          | Mode Register              | SPI_MR        | Read/write | 0x0        |
| 0x08          | Receive Data Register      | SPI_RDR       | Read-only  | 0x0        |
| 0x0C          | Transmit Data Register     | SPI_TDR       | Write-only | ---        |
| 0x10          | Status Register            | SPI_SR        | Read-only  | 0x000000F0 |
| 0x14          | Interrupt Enable Register  | SPI_IER       | Write-only | ---        |
| 0x18          | Interrupt Disable Register | SPI_IDR       | Write-only | ---        |
| 0x1C          | Interrupt Mask Register    | SPI_IMR       | Read-only  | 0x0        |
| 0x20 - 0x2C   | Reserved                   |               |            |            |
| 0x30          | Chip Select Register 0     | SPI_CSR0      | Read/write | 0x0        |
| 0x34          | Chip Select Register 1     | SPI_CSR1      | Read/write | 0x0        |
| 0x38          | Chip Select Register 2     | SPI_CSR2      | Read/write | 0x0        |
| 0x3C          | Chip Select Register 3     | SPI_CSR3      | Read/write | 0x0        |
| 0x40 - 0xFF   | Reserved                   |               |            |            |
| 0x100 - 0x124 | Reserved for the PDC       |               |            |            |

**SPI Control Register**

**Name:** SPI\_CR

**Access Type:** Write-only

|       |    |    |    |    |    |        |       |
|-------|----|----|----|----|----|--------|-------|
| 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**

0 = No effect.

1 = Enables the SPI to transfer and receive data.

- **SPIDIS: SPI Disable**

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**

0 = No effect.

1 = Reset the SPI.

A software-triggered hardware reset of the SPI interface is performed.

## SPI Mode Register

**Name:** SPI\_MR

**Access Type:** Read/write

|        |    |    |         |       |        |    |      |
|--------|----|----|---------|-------|--------|----|------|
| 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    | –  | –  | MODFDIS | DIV32 | PCSDEC | PS | MSTR |

- **MSTR: Master/Slave Mode**

0 = SPI is in Slave mode.

1 = SPI is in Master mode.

- **PS: Peripheral Select**

0 = Fixed Peripheral Select.

1 = Variable Peripheral Select.

- **PCSDEC: Chip Select Decode**

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:

SPI\_CSR0 defines peripheral chip select signals 0 to 3.

SPI\_CSR1 defines peripheral chip select signals 4 to 7.

SPI\_CSR2 defines peripheral chip select signals 8 to 11.

SPI\_CSR3 defines peripheral chip select signals 12 to 15\*.

*\*Note: 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.*

- **DIV32: Clock Selection**

0 = The SPI operates at MCK.

1 = The SPI operates at MCK/32.

- **MODFDIS: Mode Fault Detection**

0 = Mode fault detection is enabled.

1 = Mode fault detection is disabled.

- **LLB: Local Loopback Enable**

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**

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**

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 MCK periods (or 192 MCK periods if DIV32 is set) will be inserted by default.

Otherwise, the following equation determines the delay:

If DIV32 is 0:

$$\text{Delay Between Chip Selects} = \text{DLYBCS} / \text{MCK}$$

If DIV32 is 1:

$$\text{Delay Between Chip Selects} = \text{DLYBCS} \times 32 / \text{MCK}$$

## SPI Receive Data Register

**Name:** SPI\_RDR

**Access Type:** Read-only

|    |    |    |    |     |    |    |    |
|----|----|----|----|-----|----|----|----|
| 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**

Data received by the SPI Interface is stored in this register right-justified. Unused bits read zero.

- **PCS: Peripheral Chip Select**

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

**Name:** SPI\_TDR

**Access Type:** Write-only

|    |    |    |    |     |    |    |    |
|----|----|----|----|-----|----|----|----|
| 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**

Data 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).

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

Name: SPI\_SR

Access Type: Read-only

|        |        |       |       |       |      |      |        |
|--------|--------|-------|-------|-------|------|------|--------|
| 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      |
| TXBUFE | RXBUFF | ENDTX | ENDRX | OVRES | MODF | TDRE | RDRF   |

- **RDRF: Receive Data Register Full**

0 = No data has been received since the last read of SPI\_RDR

1 = Data has been received and the received data has been transferred from the serializer to SPI\_RDR since the last read of SPI\_RDR.

- **TDRE: Transmit Data Register Empty**

0 = Data has been written to SPI\_TDR and not yet transferred to the serializer.

1 = The last data written in the Transmit Data Register has been transferred to 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**

0 = No Mode Fault has been detected since the last read of SPI\_SR.

1 = A Mode Fault occurred since the last read of the SPI\_SR.

- **OVRES: Overrun Error Status**

0 = No overrun has been detected since the last read of SPI\_SR.

1 = An overrun has occurred since the last read of SPI\_SR.

An overrun occurs when SPI\_RDR is loaded at least twice from the serializer since the last read of the SPI\_RDR.

- **ENDRX: End of RX buffer**

0 = The Receive Counter Register has not reached 0 since the last write in SPI\_RCR or SPI\_RNCR.

1 = The Receive Counter Register has reached 0 since the last write in SPI\_RCR or SPI\_RNCR.

- **ENDTX: End of TX buffer**

0 = The Transmit Counter Register has not reached 0 since the last write in SPI\_TCR or SPI\_TNCR.

1 = The Transmit Counter Register has reached 0 since the last write in SPI\_TCR or SPI\_TNCR.

- **RXBUFF: RX Buffer Full**

0 = SPI\_RCR or SPI\_RNCR have a value other than 0.

1 = Both SPI\_RCR and SPI\_RNCR have a value of 0.

- **TXBUFE: TX Buffer Empty**

0 = SPI\_TCR or SPI\_TNCR have a value other than 0.

1 = Both SPI\_TCR and SPI\_TNCR have a value of 0.

- **SPIENS: SPI Enable Status**

0 = SPI is disabled.

1 = SPI is enabled.

## SPI Interrupt Enable Register

**Name:** SPI\_IER

**Access Type:** Write-only

|        |        |       |       |       |      |      |      |
|--------|--------|-------|-------|-------|------|------|------|
| 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    |
| TXBUFE | RXBUFF | ENDTX | ENDRX | OVRES | MODF | TDRE | RDRF |

- **RDRF: Receive Data Register Full Interrupt Enable**
- **TDRE: SPI Transmit Data Register Empty Interrupt Enable**
- **MODF: Mode Fault Error Interrupt Enable**
- **OVRES: Overrun Error Interrupt Enable**
- **ENDRX: End of Receive Buffer Interrupt Enable**
- **ENDTX: End of Transmit Buffer Interrupt Enable**
- **RXBUFF: Receive Buffer Full Interrupt Enable**
- **TXBUFE: Transmit Buffer Empty Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.



**SPI Interrupt Disable Register**

**Name:** SPI\_IDR

**Access Type:** Write-only

|        |        |       |       |       |      |      |      |
|--------|--------|-------|-------|-------|------|------|------|
| 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    |
| TXBUFE | RXBUFF | ENDTX | ENDRX | OVRES | MODF | TDRE | RDRF |

- **RDRF: Receive Data Register Full Interrupt Disable**
- **TDRE: SPI Transmit Data Register Empty Interrupt Disable**
- **MODF: Mode Fault Error Interrupt Disable**
- **OVRES: Overrun Error Interrupt Disable**
- **ENDRX: End of Receive Buffer Interrupt Disable**
- **ENDTX: End of Transmit Buffer Interrupt Disable**
- **RXBUFF: Receive Buffer Full Interrupt Disable**
- **TXBUFE: Transmit Buffer Empty Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

## SPI Interrupt Mask Register

Name: SPI\_IMR

Access Type: Read-only

|        |        |       |       |       |      |      |      |
|--------|--------|-------|-------|-------|------|------|------|
| 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    |
| TXBUFE | RXBUFF | ENDTX | ENDRX | OVRES | MODF | TDRE | RDRF |

- **RDRF: Receive Data Register Full Interrupt Mask**
- **TDRE: SPI Transmit Data Register Empty Interrupt Mask**
- **MODF: Mode Fault Error Interrupt Mask**
- **OVRES: Overrun Error Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask**
- **ENDTX: End of Transmit Buffer Interrupt Mask**
- **RXBUFF: Receive Buffer Full Interrupt Mask**
- **TXBUFE: Transmit Buffer Empty Interrupt Mask**

0 = The corresponding interrupt is not enabled.

1 = The corresponding interrupt is enabled.

## SPI Chip Select Register

**Name:** SPI\_CSR0... SPI\_CSR3

**Access Type:** Read/write

|        |    |    |    |    |    |       |      |
|--------|----|----|----|----|----|-------|------|
| 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**

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 the required clock/data relationship between master and slave devices.

- **NCPHA: Clock Phase**

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 the required 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 |
|-----------|-------------------|
| 0000      | 8                 |
| 0001      | 9                 |
| 0010      | 10                |
| 0011      | 11                |
| 0100      | 12                |
| 0101      | 13                |
| 0110      | 14                |
| 0111      | 15                |
| 1000      | 16                |
| 1001      | Reserved          |
| 1010      | Reserved          |
| 1011      | Reserved          |
| 1100      | Reserved          |
| 1101      | Reserved          |
| 1110      | Reserved          |
| 1111      | Reserved          |

- **SCBR: Serial Clock Baud Rate**

In Master Mode, the SPI Interface uses a modulus counter to derive the SPCK baud rate from the Master Clock MCK. 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:

If DIV32 is 0:

$$\text{SPCK Baudrate} = \text{MCK} / (2 \times \text{SCBR})$$

If DIV32 is 1:

$$\text{SPCK Baudrate} = \text{MCK} / (64 \times \text{SCBR})$$

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**

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 equations determine the delay:

If DIV32 is 0:

$$\text{Delay Before SPCK} = \text{DLYBS} / \text{MCK}$$

If DIV32 is 1:

$$\text{Delay Before SPCK} = 32 \times \text{DLYBS} / \text{MCK}$$

- **DLYBCT: Delay Between Consecutive Transfers**

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 minimum delay of four MCK cycles are inserted (or 128 MCK cycles when DIV32 is set) between two consecutive characters.

Otherwise, the following equation determines the delay:

If DIV32 is 0:

$$\text{Delay Between Consecutive Transfers} = 32 \times \text{DLYBCT} / \text{MCK}$$

If DIV32 is 1:

$$\text{Delay Between Consecutive Transfers} = 1024 \times \text{DLYBCT} / \text{MCK}$$

## Two-wire Interface (TWI)

### Overview

The Two-wire Interface (TWI) interconnects components on a unique two-wire bus, made up of one clock line and one data line with speeds of up to 400 Kbits per second, based on a byte-oriented transfer format. It can be used with any Atmel two-wire bus serial EEPROM. The TWI is programmable as a master with sequential or single-byte access.

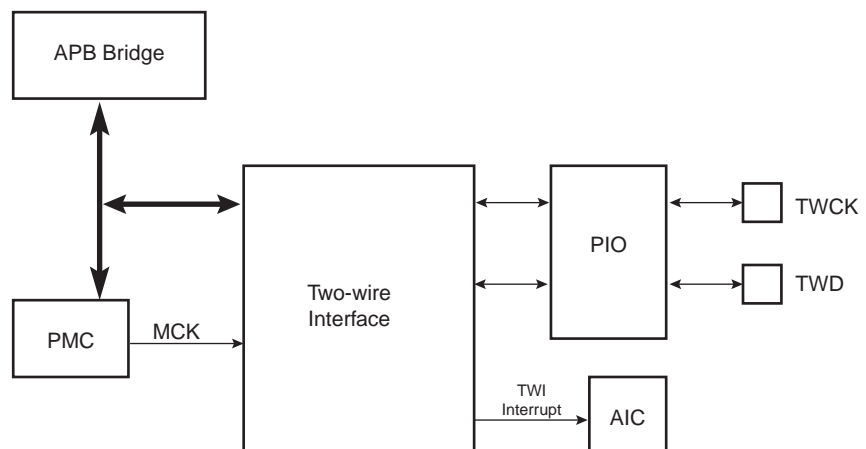
A configurable baud rate generator permits the output data rate to be adapted to a wide range of core clock frequencies.

The main features of the TWI are:

- Compatibility with standard two-wire serial memory
- One, two or three bytes for slave address
- Sequential read/write operations

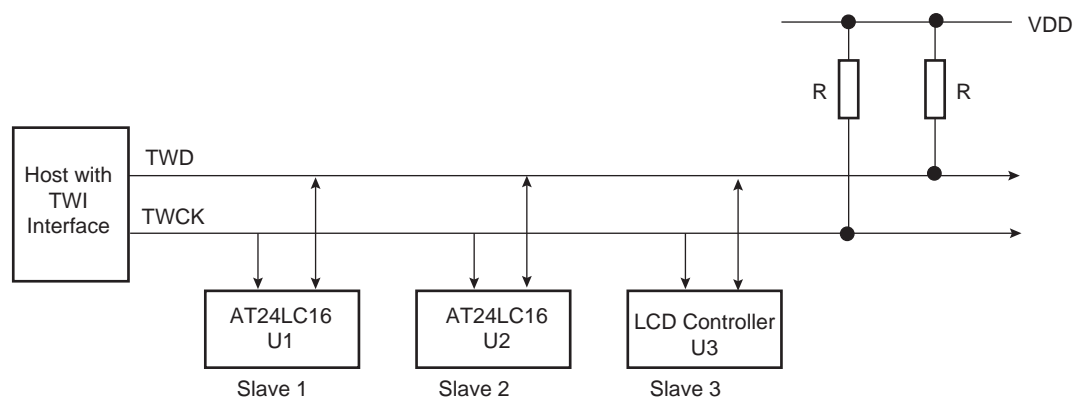
### Block Diagram

Figure 161. Block Diagram



### Application Block Diagram

Figure 162. Application Block Diagram



**Table 71.** I/O Lines Description

| Pin Name | Pin Description       | Type         |
|----------|-----------------------|--------------|
| TWD      | Two-wire Serial Data  | Input/Output |
| TWCK     | Two-wire Serial Clock | Input/Output |

## Product Dependencies

### I/O Lines

Both TWD and TWCK are bi-directional lines, connected to a positive supply voltage via a current source or pull-up resistor (see Figure 162 on page 381). When the bus is free, both lines are high. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function.

TWD and TWCK pins may be multiplexed with PIO lines. To enable the TWI, the programmer must perform the following steps:

- Program the PIO controller to:
  - Dedicate TWD and TWCK as peripheral lines.
  - Define TWD and TWCK as open-drain.

### Power Management

- Enable the peripheral clock.

The TWI interface may be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the TWI clock.

### Interrupt

The TWI interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). In order to handle interrupts, the AIC must be programmed before configuring the TWI.

## Functional Description

### Transfer Format

The data put on the TWD line must be eight bits long. Data is transferred MSB first; each byte must be followed by an acknowledgement. The number of bytes per transfer is unlimited (see Figure 164 on page 383).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 163 on page 382).

- A high-to-low transition on the TWD line while TWCK is high defines the START condition.
- A low-to-high transition on the TWD line while TWCK is high defines a STOP condition.

**Figure 163.** START and STOP Conditions

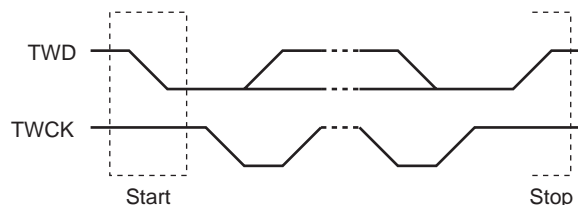
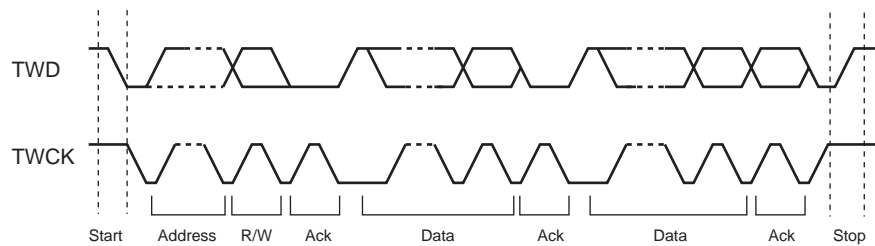


Figure 164. Transfer Format



## Modes of Operation

The TWI has two modes of operations:

- Master transmitter mode
- Master receiver mode

The TWI Control Register (TWI\_CR) allows configuration of the interface in Master Mode. In this mode, it generates the clock according to the value programmed in the Clock Waveform Generator Register (TWI\_CWGR). This register defines the TWCK signal completely, enabling the interface to be adapted to a wide range of clocks.

## Transmitting Data

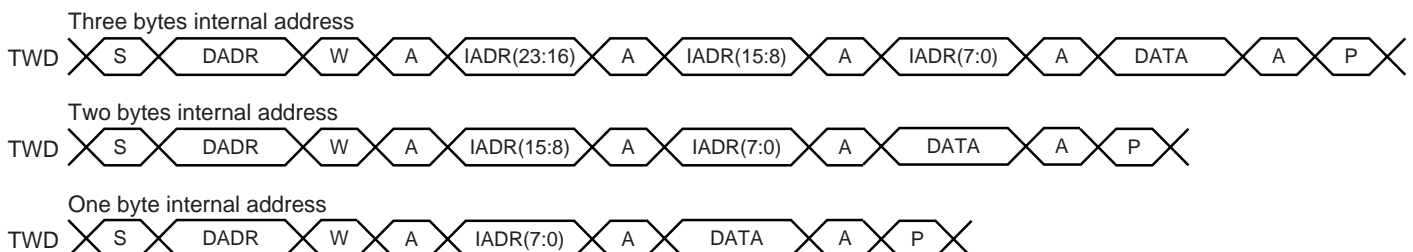
After the master initiates a Start condition, it sends a 7-bit slave address, configured in the Master Mode register (DADR in TWI\_MMR), to notify the slave device. The bit following the slave address indicates the transfer direction (write or read). If this bit is 0, it indicates a write operation (transmit operation). If the bit is 1, it indicates a request for data read (receive operation).

The TWI transfers require the slave to acknowledge each received byte. During the acknowledge clock pulse, the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. The master polls the data line during this clock pulse and sets the **NAK** bit in the status register if the slave does not acknowledge the byte. As with the other status bits, an interrupt can be generated if enabled in the interrupt enable register (TWI\_IER). After writing in the transmit-holding register (TWI\_THR), setting the START bit in the control register starts the transmission. The data is shifted in the internal shifter and when an acknowledge is detected, the TXRDY bit is set until a new write in the TWI\_THR (see Figure 166 on page 384). The master generates a stop condition to end the transfer.

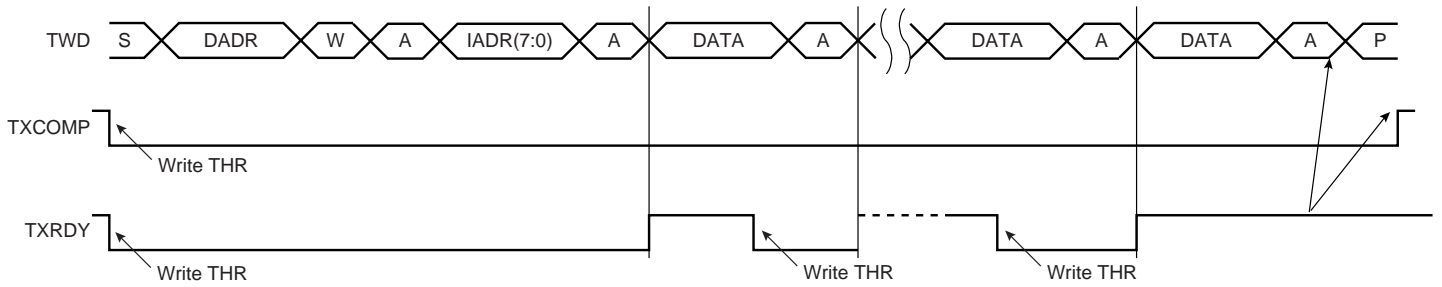
The read sequence begins by setting the START bit. When the RXRDY bit is set in the status register, a character has been received in the receive-holding register (TWI\_RHR). The RXRDY bit is reset when reading the TWI\_RHR.

The TWI interface performs various transfer formats (7-bit slave address, 10-bit slave address). The three internal address bytes are configurable through the Master Mode register (TWI\_MMR). If the slave device supports only a 7-bit address, the **IADRSZ** must be set to 0. For slave address higher than seven bits, the user must configure the address size (**IADRSZ**) and set the other slave address bits in the internal address register (TWI\_IADR).

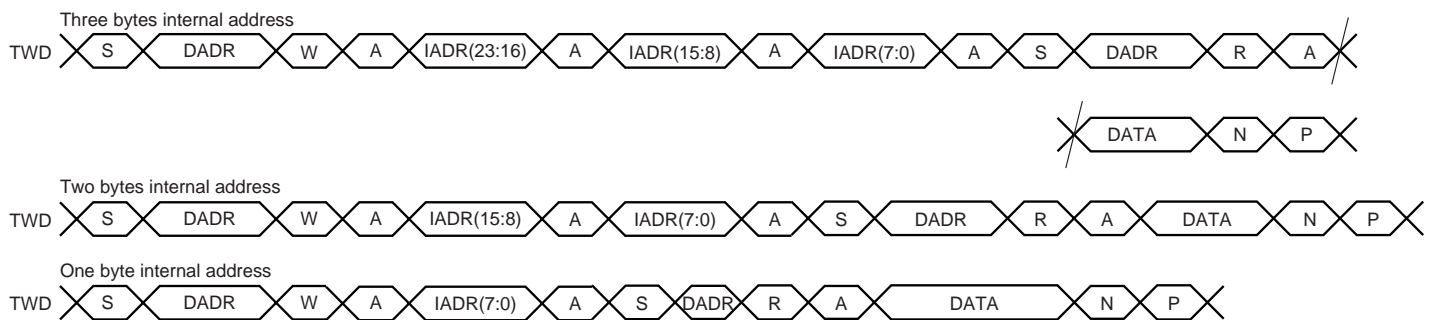
Figure 165. Master Write with One, Two or Three Bytes Internal Address and One Data Byte



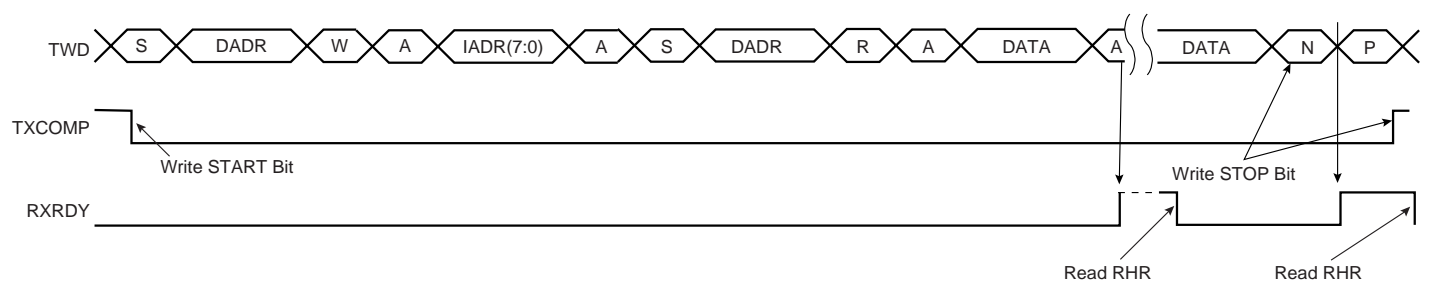
**Figure 166. Master Write with One Byte Internal Address and Multiple Data Bytes**



**Figure 167. Master Read with One, Two or Three Bytes Internal Address and One Data Byte**



**Figure 168. Master Read with One Byte Internal Address and Multiple Data Bytes**

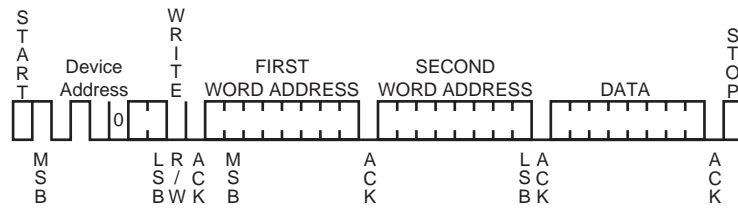


- S = Start
- P = Stop
- W = Write/read
- A = Acknowledge
- DADR= Device Address
- IADR = Internal Address

Figure 169 shows a byte write to an Atmel AT24LC512 EEPROM. This demonstrates the use of internal addresses to access the device.



Figure 169. Internal Address Usage



**Read/Write Flowcharts**

The following flowcharts shown in Figure 170 on page 386 and in Figure 171 on page 387 give examples for read and write operations in Master Mode. A polling or interrupt method can be used to check the status bits. The interrupt method requires that the interrupt enable register (TWI\_IER) be configured first.

**Figure 170.** TWI Write in Master Mode

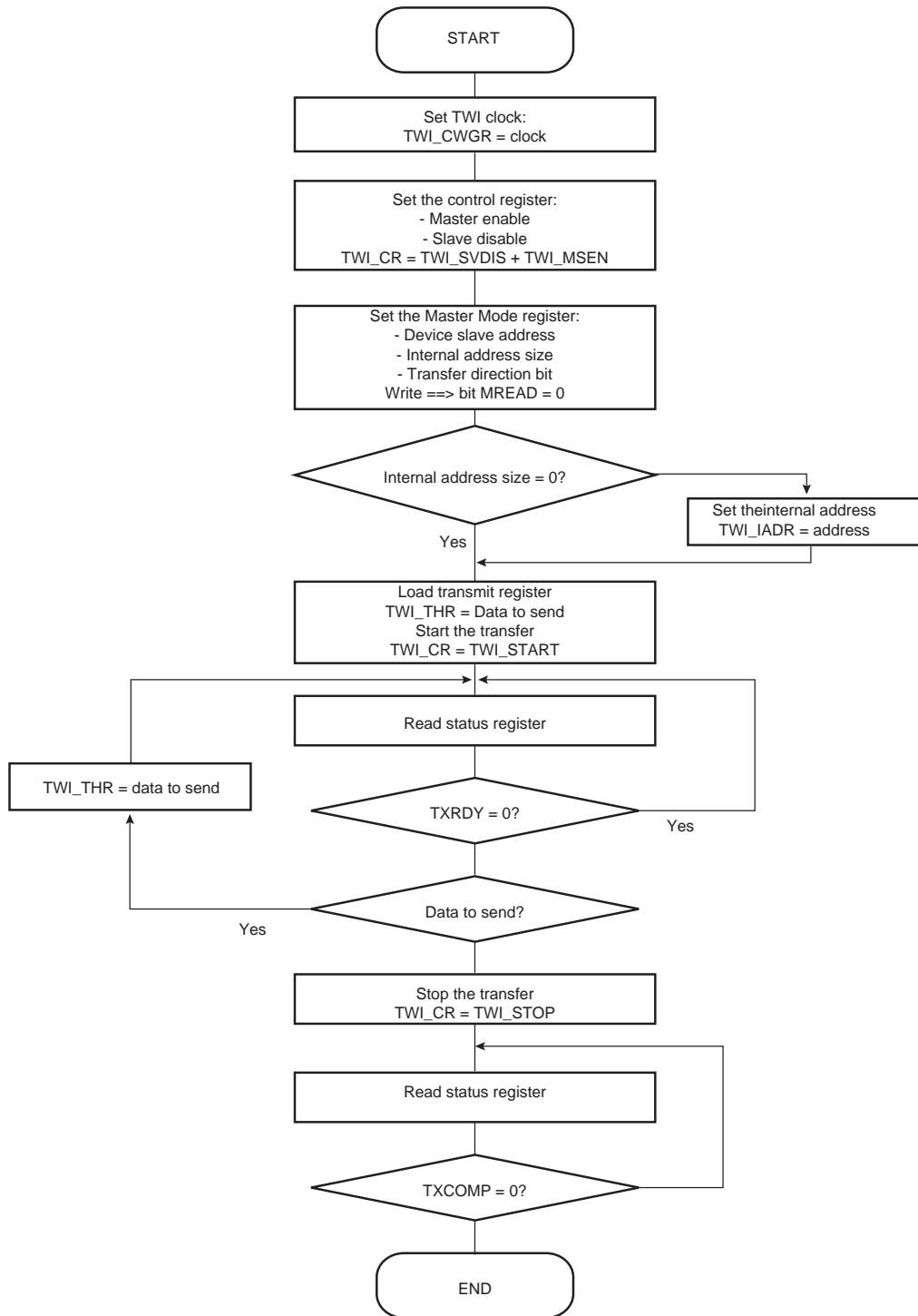
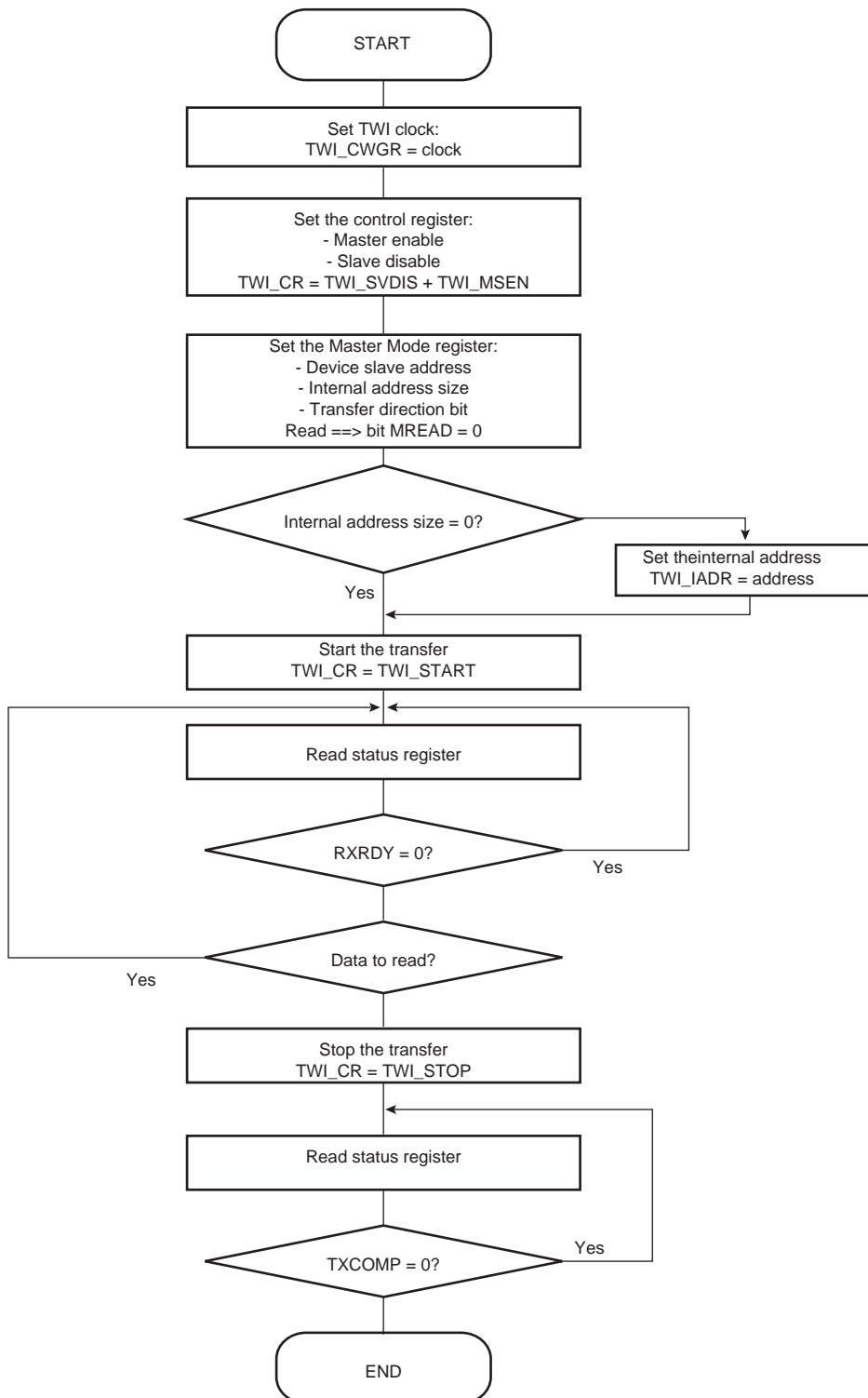


Figure 171. TWI Read in Master Mode



## Two-wire Interface (TWI) User Interface

**Table 72.** TWI Register Mapping

| Offset | Register                          | Name     | Access     | Reset Value |
|--------|-----------------------------------|----------|------------|-------------|
| 0x0000 | Control Register                  | TWI_CR   | Write-only | N/A         |
| 0x0004 | Master Mode Register              | TWI_MMR  | Read/write | 0x0000      |
| 0x0008 | Reserved                          |          |            |             |
| 0x000C | Internal Address Register         | TWI_IADR | Read/write | 0x0000      |
| 0x0010 | Clock Waveform Generator Register | TWI_CWGR | Read/write | 0x0000      |
| 0x0020 | Status Register                   | TWI_SR   | Read-only  | 0x0008      |
| 0x0024 | Interrupt Enable Register         | TWI_IER  | Write-only | N/A         |
| 0x0028 | Interrupt Disable Register        | TWI_IDR  | Write-only | N/A         |
| 0x002C | Interrupt Mask Register           | TWI_IMR  | Read-only  | 0x0000      |
| 0x0030 | Receive Holding Register          | TWI_RHR  | Read-only  | 0x0000      |
| 0x0034 | Transmit Holding Register         | TWI_THR  | Read/write | 0x0000      |

**TWI Control Register**

**Register Name:** TWI\_CR

**Access Type:** Write-only

|       |    |    |    |       |      |      |       |
|-------|----|----|----|-------|------|------|-------|
| 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 | –  | –  | –  | MSDIS | MSEN | STOP | START |

• **START: Send a START Condition**

0 = No effect.

1 = A frame beginning with a START bit is transmitted according to the features defined in the mode register.

This action is necessary when the TWI peripheral wants to read data from a slave. When configured in Master Mode with a write operation, a frame is sent with the mode register as soon as the user writes a character in the holding register.

• **STOP: Send a STOP Condition**

0 = No effect.

1 = STOP Condition is sent just after completing the current byte transmission in master read or write mode.

In single data byte master read or write, the START and STOP must both be set.

In multiple data bytes master read or write, the STOP must be set before ACK/NACK bit transmission.

In master read mode, if a NACK bit is received, the STOP is automatically performed.

In multiple data write operation, when both THR and shift register are empty, a STOP condition is automatically sent.

• **MSEN: TWI Master Transfer Enabled**

0 = No effect.

1 = If MSDIS = 0, the master data transfer is enabled.

• **MSDIS: TWI Master Transfer Disabled**

0 = No effect.

1 = The master data transfer is disabled, all pending data is transmitted. The shifter and holding characters (if it contains data) are transmitted in case of write operation. In read operation, the character being transferred must be completely received before disabling.

• **SWRST: Software Reset**

0 = No effect.

1 = Equivalent to a system reset.

## TWI Master Mode Register

Register Name: TWI\_MMR

Address Type: Read/write

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

- **IADRSZ: Internal Device Address Size**

| IADRSZ[9:8] |   |                                    |
|-------------|---|------------------------------------|
| 0           | 0 | No internal device address         |
| 0           | 1 | One-byte internal device address   |
| 1           | 0 | Two-byte internal device address   |
| 1           | 1 | Three-byte internal device address |

- **MREAD: Master Read Direction**

0 = Master write direction.

1 = Master read direction.

- **DADR: Device Address**

The device address is used in Master Mode to access slave devices in read or write mode.

### TWI Internal Address Register

Register Name: TWI\_IADR

Access Type: Read/write

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

- **IADR: Internal Address**

0, 1, 2 or 3 bytes depending on IADRSZ.

### TWI Clock Waveform Generator Register

Register Name: TWI\_CWGR

Access Type: Read/write

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

- **CLDIV: Clock Low Divider**

The TWCK low period is defined as follows:

$$T_{low} = ((CLDIV \times 2^{CKDIV}) + 3) \times T_{MCK}$$

- **CHDIV: Clock High Divider**

The TWCK high period is defined as follows:

$$T_{high} = ((CHDIV \times 2^{CKDIV}) + 3) \times T_{MCK}$$

- **CKDIV: Clock Divider**

The CKDIV is used to increase both TWCK high and low periods.

## TWI Status Register

**Register Name:** TWI\_SR

**Access Type:** Read-only

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

- **TXCOMP: Transmission Completed**

0 = In master, during the length of the current frame. In slave, from START received to STOP received.

1 = When both holding and shifter registers are empty and STOP condition has been sent (in Master) or received (in Slave), or when MSEN is set (enable TWI).

- **RXRDY: Receive Holding Register Ready**

0 = No character has been received since the last TWI\_RHR read operation.

1 = A byte has been received in the TWI\_RHR since the last read.

- **TXRDY: Transmit Holding Register Ready**

0 = The transmit holding register has not been transferred into shift register. Set to 0 when writing into TWI\_THR register.

1 = As soon as data byte is transferred from TWI\_THR to internal shifter or if a NACK error is detected, TXRDY is set at the same time as TXCOMP and NACK. TXRDY is also set when MSEN is set (enable TWI).

- **OVRE: Overrun Error**

0 = TWI\_RHR has not been loaded while RXRDY was set

1 = TWI\_RHR has been loaded while RXRDY was set. Reset by read in TWI\_SR when TXCOMP is set.

- **UNRE: Underrun Error**

0 = No underrun error

1 = No valid data in TWI\_THR (TXRDY set) while trying to load the data shifter. This action automatically generated a STOP bit in Master Mode. Reset by read in TWI\_SR when TXCOMP is set.

- **NACK: Not Acknowledged**

0 = Each data byte has been correctly received by the far-end side TWI slave component.

1 = A data byte has not been acknowledged by the slave component. Set at the same time as TXCOMP. Reset after read.



**TWI Interrupt Enable Register**

Register Name: TWI\_IER

Access Type: Write-only

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

- **TXCOMP: Transmission Completed**
- **RXRDY: Receive Holding Register Ready**
- **TXRDY: Transmit Holding Register Ready**
- **OVRE: Overrun Error**
- **UNRE: Underrun Error**
- **NACK: Not Acknowledge**

0 = No effect.

1 = Enables the corresponding interrupt.

## TWI Interrupt Disable Register

Register Name: TWI\_IDR

Access Type: Write-only

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

- **TXCOMP:** Transmission Completed
- **RXRDY:** Receive Holding Register Ready
- **TXRDY:** Transmit Holding Register Ready
- **OVRE:** Overrun Error
- **UNRE:** Underrun Error
- **NACK:** Not Acknowledge

0 = No effect.

1 = Disables the corresponding interrupt.

**TWI Interrupt Mask Register**

Register Name: TWI\_IMR

Access Type: Read-only

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

- **TXCOMP: Transmission Completed**
- **RXRDY: Receive Holding Register Ready**
- **TXRDY: Transmit Holding Register Ready**
- **OVRE: Overrun Error**
- **UNRE: Underrun Error**
- **NACK: Not Acknowledge**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.



## TWI Receive Holding Register

Register Name: TWI\_RHR

Access Type: Read-only

|        |    |    |    |    |    |    |    |
|--------|----|----|----|----|----|----|----|
| 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  |
| RXDATA |    |    |    |    |    |    |    |

- RXDATA: Master or Slave Receive Holding Data

## TWI Transmit Holding Register

Register Name: TWI\_THR

Access Type: Read/write

|        |    |    |    |    |    |    |    |
|--------|----|----|----|----|----|----|----|
| 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  |
| TXDATA |    |    |    |    |    |    |    |

- TXDATA: Master or Slave Transmit Holding Data

# Universal Synchronous Asynchronous Receiver Transceiver (USART)

## Overview

The Universal Synchronous Asynchronous Receiver Transceiver (USART) provides one full duplex universal synchronous asynchronous serial link. Data frame format is widely programmable (data length, parity, number of stop bits) to support a maximum of standards. The receiver implements parity error, framing error and overrun error detection. The receiver time-out enables handling variable-length frames and the transmitter timeguard facilitates communications with slow remote devices. Multi-drop communications are also supported through address bit handling in reception and transmission.

The USART features three test modes: remote loopback, local loopback and automatic echo.

The USART supports specific operating modes providing interfaces on RS485 busses, with ISO7816 T = 0 or T = 1 smart card slots, infrared transceivers and connection to modem ports. The hardware handshaking feature enables an out-of-band flow control by automatic management of the pins RTS and CTS.

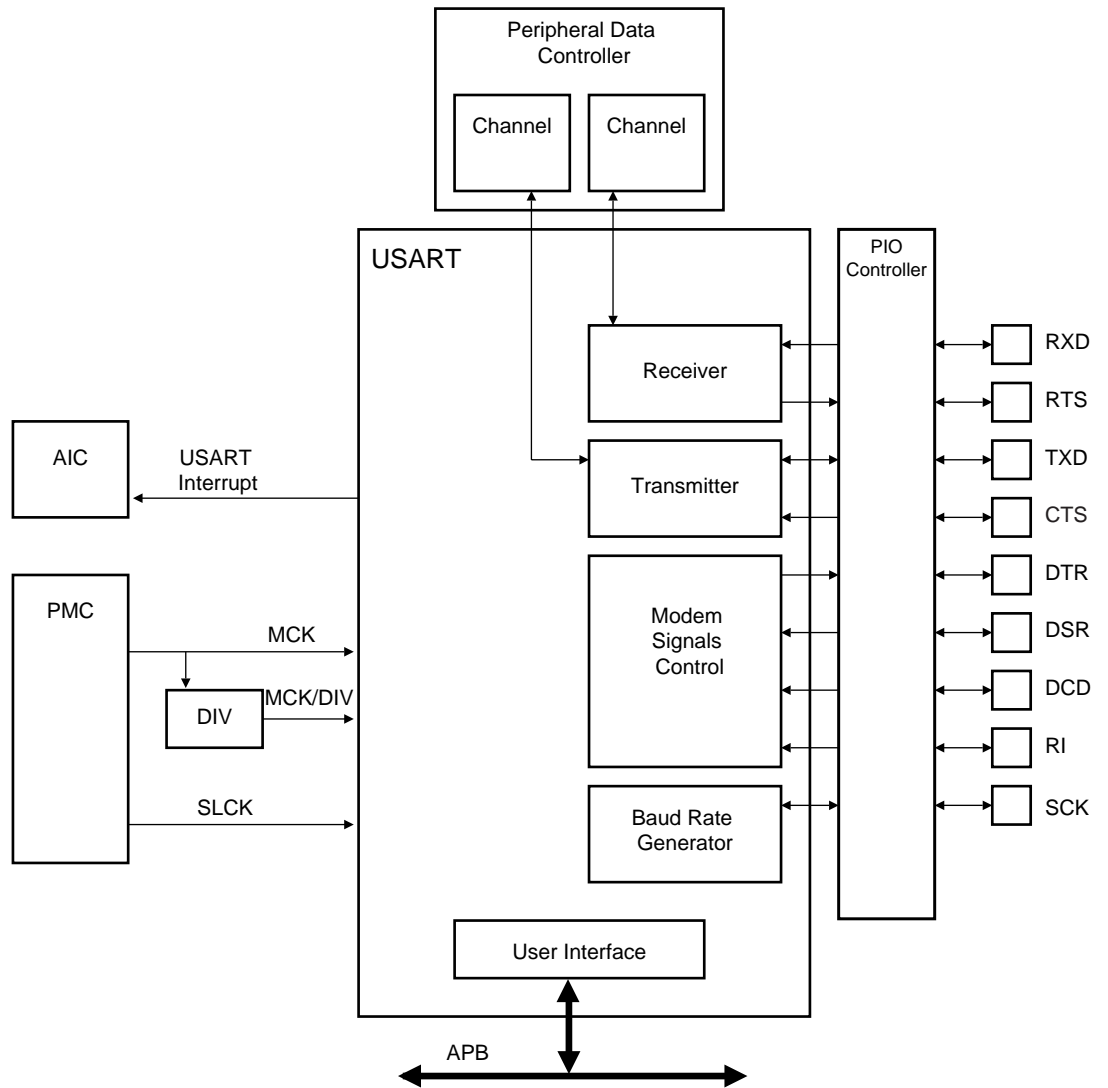
The USART supports the connection to the Peripheral Data Controller, which enables data transfers to the transmitter and from the receiver. The PDC provides chained buffer management without any intervention of the processor.

Important features of the USART are:

- Programmable Baud Rate Generator
- 5- to 9-bit Full-duplex Synchronous or Asynchronous Serial Communications
  - 1, 1.5 or 2 Stop Bits in Asynchronous Mode or 1 or 2 Stop Bits in Synchronous Mode
  - Parity Generation and Error Detection
  - Framing Error Detection, Overrun Error Detection
  - MSB- or LSB-first
  - Optional Break Generation and Detection
  - By 8 or by-16 Over-sampling Receiver Frequency
  - Optional Hardware Handshaking RTS-CTS
  - Optional Modem Signal Management DTR-DSR-DCD-RI
  - Receiver Time-out and Transmitter Timeguard
  - Optional Multi-Drop Mode with Address Generation and Detection
- RS485 with driver control signal
- ISO7816, T = 0 or T = 1 Protocols for Interfacing with Smart Cards
  - NACK Handling, Error Counter with Repetition and Iteration Limit
- IrDA Modulation and Demodulation
  - Communication at up to 115.2 Kbps
- Test Modes
  - Remote Loopback, Local Loopback, Automatic Echo
- Supports Connection of Two Peripheral Data Controller Channels (PDC)
  - Offer Buffer Transfer without Processor Intervention

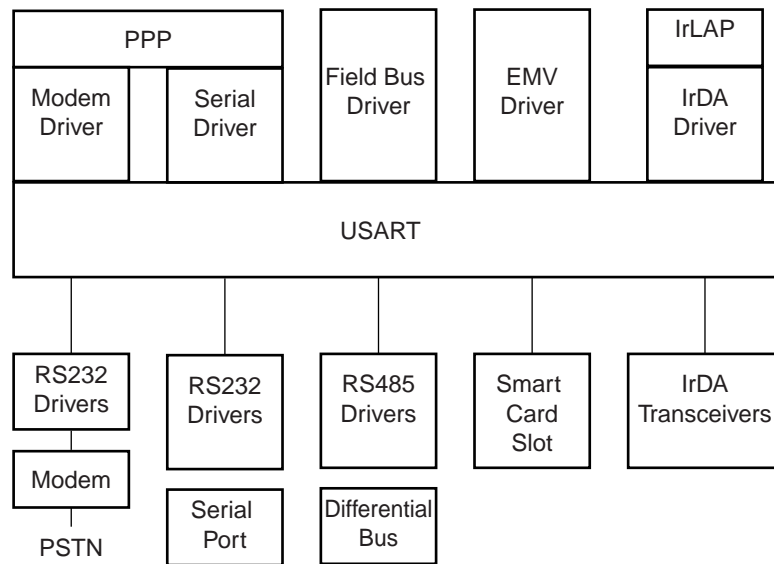
# Block Diagram

Figure 172. USART Block Diagram



## Application Block Diagram

Figure 173. Application Block Diagram



## I/O Lines Description

Table 73. I/O Line Description

| Name | Description          | Type   | Active Level |
|------|----------------------|--------|--------------|
| SCK  | Serial Clock         | I/O    |              |
| TXD  | Transmit Serial Data | I/O    |              |
| RXD  | Receive Serial Data  | Input  |              |
| RI   | Ring Indicator       | Input  | Low          |
| DSR  | Data Set Ready       | Input  | Low          |
| DCD  | Data Carrier Detect  | Input  | Low          |
| DTR  | Data Terminal Ready  | Output | Low          |
| CTS  | Clear to Send        | Input  | Low          |
| RTS  | Request to Send      | Output | Low          |

## Product Dependencies

### I/O Lines

The pins used for interfacing the USART may be multiplexed with the PIO lines. The programmer must first program the PIO controller to assign the desired USART pins to their peripheral function. If I/O lines of the USART are not used by the application, they can be used for other purposes by the PIO Controller.

All the pins of the modems may or may not be implemented on the USART within a product. Frequently, only the USART1 is fully equipped with all the modem signals. For the other

USARTs of the product not equipped with the corresponding pin, the associated control bits and statuses have no effect on the behavior of the USART.

## Power Management

The USART is not continuously clocked. The programmer must first enable the USART Clock in the Power Management Controller (PMC) before using the USART. However, if the application does not require USART operations, the USART clock can be stopped when not needed and be restarted later. In this case, the USART will resume its operations where it left off.

Configuring the USART does not require the USART clock to be enabled.

## Interrupt

The USART interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the USART interrupt requires the AIC to be programmed first. Note that it is not recommended to use the USART interrupt line in edge sensitive mode.

## Functional Description

The USART is capable of managing several types of serial synchronous or asynchronous communications.

It supports the following communication modes.

- 5- to 9-bit full-duplex asynchronous serial communication:
  - MSB- or LSB-first
  - 1, 1.5 or 2 stop bits
  - Parity even, odd, marked, space or none
  - By-8 or by-16 over-sampling receiver frequency
  - Optional hardware handshaking
  - Optional modem signals management
  - Optional break management
  - Optional multi-drop serial communication
- High-speed 5- to 9-bit full-duplex synchronous serial communication:
  - MSB- or LSB-first
  - 1 or 2 stop bits
  - Parity even, odd, marked, space or none
  - by 8 or by-16 over-sampling frequency
  - Optional Hardware handshaking
  - Optional Modem signals management
  - Optional Break management
  - Optional Multi-Drop serial communication
- RS485 with driver control signal
- ISO7816, T0 or T1 protocols for interfacing with smart cards
  - NACK handling, error counter with repetition and iteration limit
- InfraRed IrDA Modulation and Demodulation
- Test modes
  - remote loopback, local loopback, automatic echo

## Baud Rate Generator

The Baud Rate Generator provides the bit period clock named the Baud Rate Clock to both the receiver and the transmitter.

The Baud Rate Generator clock source can be selected by setting the USCLKS field in the Mode Register (US\_MR) between:

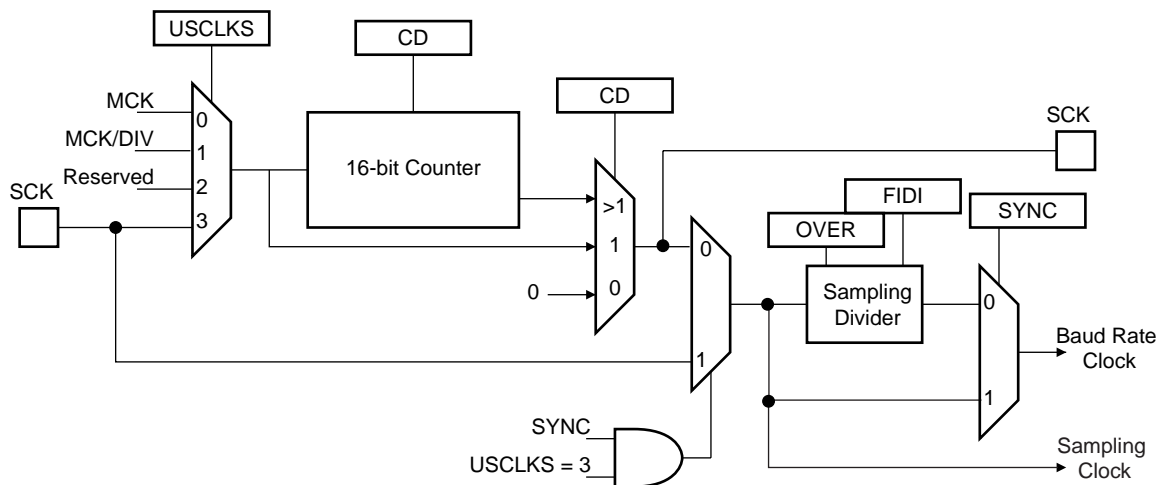


- the Master Clock MCK
- A division of the Master Clock, the divider being product dependent, but generally set to 8
- the external clock, available on the SCK pin

The Baud Rate Generator is based upon a 16-bit divider, which is programmed with the CD field of the Baud Rate Generator Register (US\_BRGR). If CD is programmed at 0, the Baud Rate Generator does not generate any clock. If CD is programmed at 1, the divider is bypassed and becomes inactive.

If the external SCK clock is selected, the duration of the low and high levels of the signal provided on the SCK pin must be longer than a Master Clock (MCK) period. The frequency of the signal provided on SCK must be at least 4.5 times lower than MCK.

Figure 174. Baud Rate Generator



**Baud Rate in Asynchronous Mode**

If the USART is programmed to operate in asynchronous mode, the selected clock is first divided by CD, which is field programmed in the Baud Rate Generator Register (US\_BRGR). The resulting clock is provided to the receiver as a sampling clock and then divided by 16 or 8, depending on the programming of the OVER bit in US\_MR.

If OVER is set to 1, the receiver sampling is 8 times higher than the baud rate clock. If OVER is cleared, the sampling is performed at 16 times the baud rate clock.

The following formula performs the calculation of the Baud Rate.

$$Baudrate = \frac{SelectedClock}{(8(2 - Over)CD)}$$

This gives a maximum baud rate of MCK divided by 8, assuming that MCK is the highest possible clock and that OVER is programmed at 1.

*Baud Rate Calculation Example*

Table 74 shows calculations of CD to obtain a baud rate at 38400 bauds for different source clock frequencies. This table also shows the actual resulting baud rate and the error.

**Table 74.** Baud Rate Example (OVER = 0)

| Source Clock | Expected Baud Rate | Calculation Result | CD  | Actual Baud Rate | Error |
|--------------|--------------------|--------------------|-----|------------------|-------|
| MHz          | Bit/s              |                    |     | Bit/s            |       |
| 3 686 400    | 38 400             | 6.00               | 6   | 38 400.00        | 0.00% |
| 4 915 200    | 38 400             | 8.00               | 8   | 38 400.00        | 0.00% |
| 5 000 000    | 38 400             | 8.14               | 8   | 39 062.50        | 1.70% |
| 7 372 800    | 38 400             | 12.00              | 12  | 38 400.00        | 0.00% |
| 8 000 000    | 38 400             | 13.02              | 13  | 38 461.54        | 0.16% |
| 12 000 000   | 38 400             | 19.53              | 20  | 37 500.00        | 2.40% |
| 12 288 000   | 38 400             | 20.00              | 20  | 38 400.00        | 0.00% |
| 14 318 180   | 38 400             | 23.30              | 23  | 38 908.10        | 1.31% |
| 14 745 600   | 38 400             | 24.00              | 24  | 38 400.00        | 0.00% |
| 18 432 000   | 38 400             | 30.00              | 30  | 38 400.00        | 0.00% |
| 24 000 000   | 38 400             | 39.06              | 39  | 38 461.54        | 0.16% |
| 24 576 000   | 38 400             | 40.00              | 40  | 38 400.00        | 0.00% |
| 25 000 000   | 38 400             | 40.69              | 40  | 38 109.76        | 0.76% |
| 32 000 000   | 38 400             | 52.08              | 52  | 38 461.54        | 0.16% |
| 32 768 000   | 38 400             | 53.33              | 53  | 38 641.51        | 0.63% |
| 33 000 000   | 38 400             | 53.71              | 54  | 38 194.44        | 0.54% |
| 40 000 000   | 38 400             | 65.10              | 65  | 38 461.54        | 0.16% |
| 50 000 000   | 38 400             | 81.38              | 81  | 38 580.25        | 0.47% |
| 60 000 000   | 38 400             | 97.66              | 98  | 38 265.31        | 0.35% |
| 70 000 000   | 38 400             | 113.93             | 114 | 38 377.19        | 0.06% |

The baud rate is calculated with the following formula:

$$BaudRate = MCK / CD \times 16$$

The baud rate error is calculated with the following formula. It is not recommended to work with an error higher than 5%.

$$Error = 1 - \left( \frac{ExpectedBaudRate}{ActualBaudRate} \right)$$

**Baud Rate in Synchronous Mode**

If the USART is programmed to operate in synchronous mode, the selected clock is simply divided by the field CD in US\_BRGR.

$$BaudRate = \frac{SelectedClock}{CD}$$

In synchronous mode, if the external clock is selected (USCLKS = 3), the clock is provided directly by the signal on the USART SCK pin. No division is active. The value written in US\_BRGR has no effect. The external clock frequency must be at least 4.5 times lower than the system clock.

When either the external clock SCK or the internal clock divided (MCK/DIV) is selected, the value programmed in CD must be even if the user has to ensure a 50:50 mark/space ratio on the SCK pin. If the internal clock MCK is selected, the Baud Rate Generator ensures a 50:50 duty cycle on the SCK pin, even if the value programmed in CD is odd.

### Baud Rate in ISO 7816 Mode

The ISO7816 specification defines the bit rate with the following formula:

$$B = \frac{D_i}{F_i} \times f$$

where:

- B is the bit rate
- Di is the bit-rate adjustment factor
- Fi is the clock frequency division factor
- f is the ISO7816 clock frequency (Hz)

Di is a binary value encoded on a 4-bit field, named DI, as represented in Table 75.

**Table 75.** Binary and Decimal Values for D

|              |      |      |      |      |      |      |      |      |
|--------------|------|------|------|------|------|------|------|------|
| DI field     | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 1000 | 1001 |
| Di (decimal) | 1    | 2    | 4    | 8    | 16   | 32   | 12   | 20   |

Fi is a binary value encoded on a 4-bit field, named FI, as represented in Table 76.

**Table 76.** Binary and Decimal Values for F

|              |      |      |      |      |      |      |      |      |      |      |      |      |
|--------------|------|------|------|------|------|------|------|------|------|------|------|------|
| FI field     | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 1001 | 1010 | 1011 | 1100 | 1101 |
| Fi (decimal) | 372  | 372  | 558  | 744  | 1116 | 1488 | 1860 | 512  | 768  | 1024 | 1536 | 2048 |

Table 77 shows the resulting Fi/Di Ratio, which is the ratio between the ISO7816 clock and the baud rate clock..

**Table 77.** Possible Values for the Fi/Di Ratio

|       |       |       |       |       |      |       |       |      |       |      |       |
|-------|-------|-------|-------|-------|------|-------|-------|------|-------|------|-------|
| Fi/Di | 372   | 558   | 774   | 1116  | 1488 | 1806  | 512   | 768  | 1024  | 1536 | 2048  |
| 1     | 372   | 558   | 744   | 1116  | 1488 | 1860  | 512   | 768  | 1024  | 1536 | 2048  |
| 2     | 186   | 279   | 372   | 558   | 744  | 930   | 256   | 384  | 512   | 768  | 1024  |
| 4     | 93    | 139.5 | 186   | 279   | 372  | 465   | 128   | 192  | 256   | 384  | 512   |
| 8     | 46.5  | 69.75 | 93    | 139.5 | 186  | 232.5 | 64    | 96   | 128   | 192  | 256   |
| 16    | 23.25 | 34.87 | 46.5  | 69.75 | 93   | 116.2 | 32    | 48   | 64    | 96   | 128   |
| 32    | 11.62 | 17.43 | 23.25 | 34.87 | 46.5 | 58.13 | 16    | 24   | 32    | 48   | 64    |
| 12    | 31    | 46.5  | 62    | 93    | 124  | 155   | 42.66 | 64   | 85.33 | 128  | 170.6 |
| 20    | 18.6  | 27.9  | 37.2  | 55.8  | 74.4 | 93    | 25.6  | 38.4 | 51.2  | 76.8 | 102.4 |

If the USART is configured in ISO7816 Mode, the clock selected by the USCLKS field in the Mode Register (US\_MR) is first divided by the value programmed in the field CD in the Baud

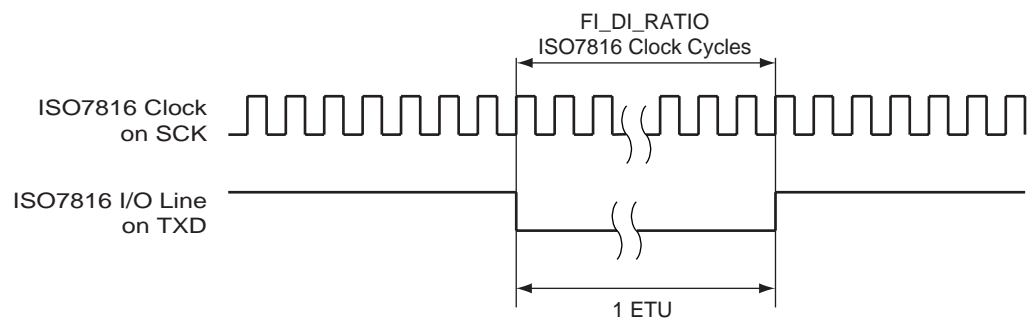
Rate Generator Register (US\_BRGR). The resulting clock can be provided to the SCK pin to feed the smart card clock inputs. This means that the CLKO bit can be set in US\_MR.

This clock is then divided by the value programmed in the FI\_DI\_RATIO field in the FI\_DI\_Ratio register (US\_FIDI). This is performed by the Sampling Divider, which performs a division by up to 2047 in ISO7816 Mode. The non-integer values of the Fi/Di Ratio are not supported and the user must program the FI\_DI\_RATIO field to a value as close as possible to the expected value.

The FI\_DI\_RATIO field resets to the value 0x174 (372 in decimal) and is the most common divider between the ISO7816 clock and the bit rate ( $F_i = 372$ ,  $D_i = 1$ ).

Figure 175 shows the relation between the Elementary Time Unit, corresponding to a bit time, and the ISO 7816 clock.

**Figure 175.** Elementary Time Unit (ETU)



## Receiver and Transmitter Control

After reset, the receiver is disabled. The user must enable the receiver by setting the RXEN bit in the Control Register (US\_CR). However, the receiver registers can be programmed before the receiver clock is enabled.

After reset, the transmitter is disabled. The user must enable it by setting the TXEN bit in the Control Register (US\_CR). However, the transmitter registers can be programmed before being enabled.

The Receiver and the Transmitter can be enabled together or independently.

At any time, the software can perform a reset on the receiver or the transmitter of the USART by setting the corresponding bit, RSTRX and RSTTX respectively, in the Control Register (US\_CR). The reset commands have the same effect as a hardware reset on the corresponding logic. Regardless of what the receiver or the transmitter is performing, the communication is immediately stopped.

The user can also independently disable the receiver or the transmitter by setting RXDIS and TXDIS respectively in US\_CR. If the receiver is disabled during a character reception, the USART waits until the end of reception of the current character, then the reception is stopped. If the transmitter is disabled while it is operating, the USART waits the end of transmission of both the current character and character being stored in the Transmit Holding Register (US\_THR). If a time guard is programmed, it is handled normally.

## Synchronous and Asynchronous Modes

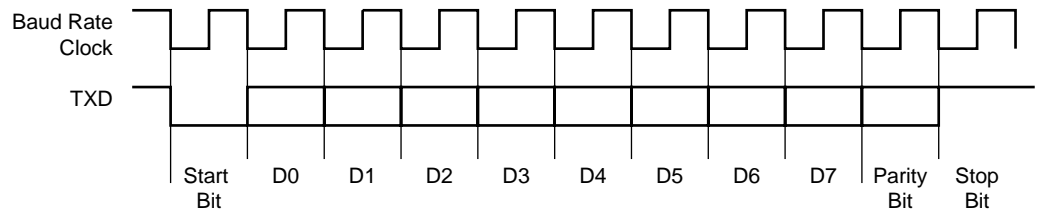
### Transmitter Operations

The transmitter performs the same in both synchronous and asynchronous operating modes (SYNC = 0 or SYNC = 1). One start bit, up to 9 data bits, one optional parity bit and up to two stop bits are successively shifted out on the TXD pin at each falling edge of the programmed serial clock.

The number of data bits is selected by the CHRL field and the MODE9 bit in the Mode Register (US\_MR). Nine bits are selected by setting the MODE 9 bit regardless of the CHRL field. The parity bit is set according to the PAR field in US\_MR. The even, odd, space, marked or none parity bit can be configured. The MSBF field in US\_MR configures which data bit is sent first. If written at 1, the most significant bit is sent first. At 0, the less significant bit is sent first. The number of stop bits is selected by the NBSTOP field in US\_MR. The 1.5 stop bit is supported in asynchronous mode only.

**Figure 176.** Character Transmit

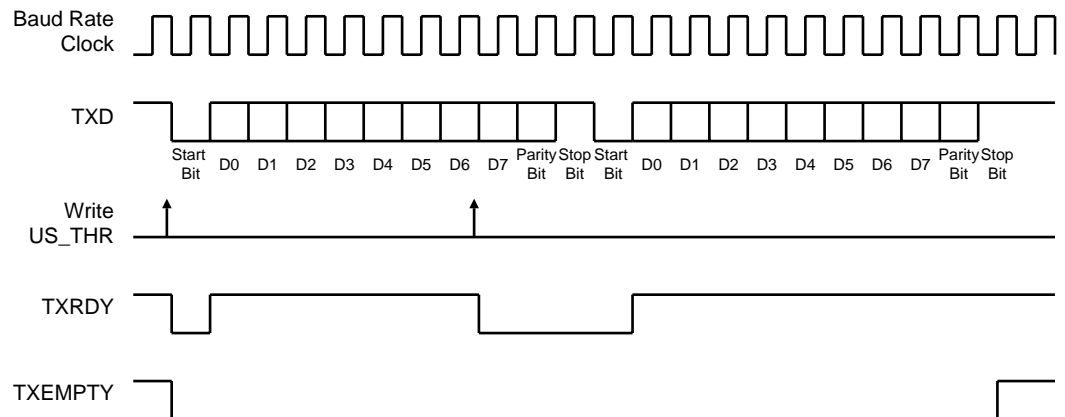
Example: 8-bit, Parity Enabled One Stop



The characters are sent by writing in the Transmit Holding Register (US\_THR). The transmitter reports two status bits in the Channel Status Register (US\_CSR): TXRDY (Transmitter Ready), which indicates that US\_THR is empty and TXEMPTY, which indicates that all the characters written in US\_THR have been processed. When the current character processing is completed, the last character written in US\_THR is transferred into the Shift Register of the transmitter and US\_THR becomes empty, thus TXRDY raises.

Both TXRDY and TXEMPTY bits are low since the transmitter is disabled. Writing a character in US\_THR while TXRDY is active has no effect and the written character is lost.

**Figure 177.** Transmitter Status



## Asynchronous Receiver

If the USART is programmed in asynchronous operating mode ( $SYNC = 0$ ), the receiver oversamples the RXD input line. The oversampling is either 16 or 8 times the Baud Rate clock, depending on the OVER bit in the Mode Register (US\_MR).

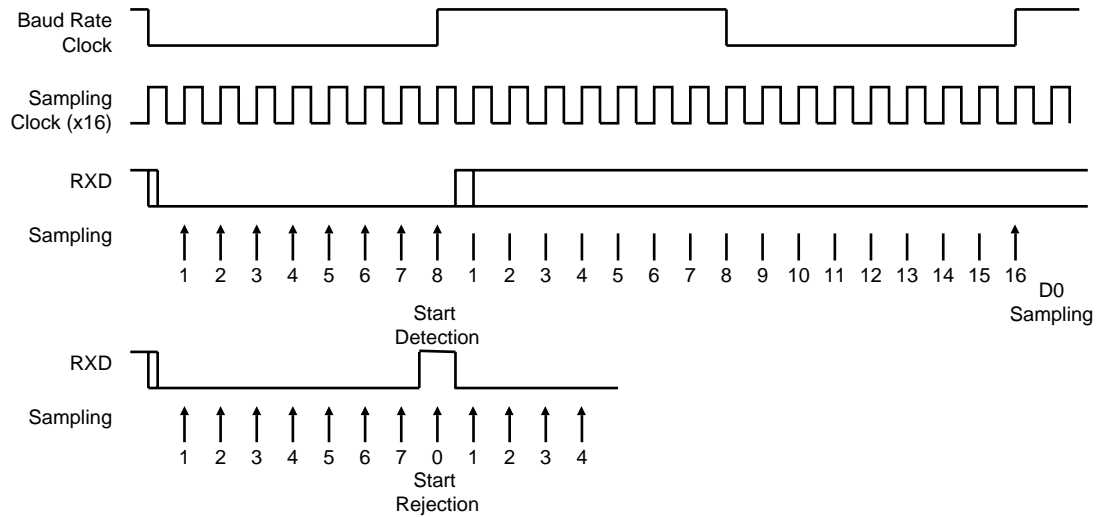
The receiver samples the RXD line. If the line is sampled during one half of a bit time at 0, a start bit is detected and data, parity and stop bits are successively sampled on the bit rate clock.

If the oversampling is 16, (OVER at 0), a start is detected at the eighth sample at 0. Then, data bits, parity bit and stop bit are sampled on each 16 sampling clock cycle. If the oversampling is 8 (OVER at 1), a start bit is detected at the fourth sample at 0. Then, data bits, parity bit and stop bit are sampled on each 8 sampling clock cycle.

The number of data bits, first bit sent and parity mode are selected by the same fields and bits as the transmitter, i.e. respectively CHRL, MODE9, MSBF and PAR. The number of stop bits has no effect on the receiver as it considers only one stop bit, regardless of the field NBSTOP, so that resynchronization between the receiver and the transmitter can occur. Moreover, as soon as the stop bit is sampled, the receiver starts looking for a new start bit so that resynchronization can also be accomplished when the transmitter is operating with one stop bit.

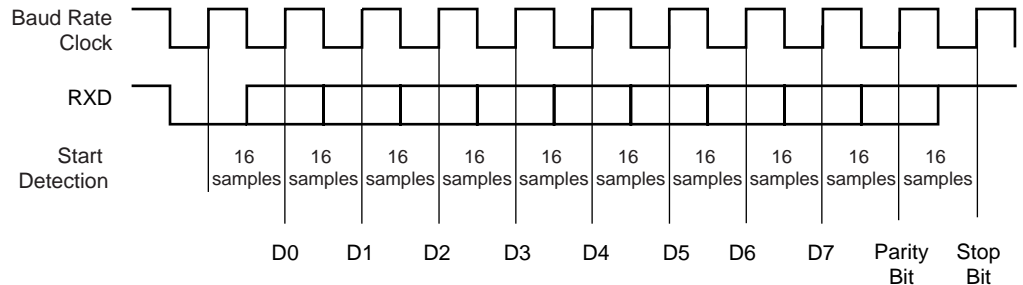
Figure 178 and Figure 179 illustrate start detection and character reception when USART operates in asynchronous mode.

**Figure 178.** Asynchronous Start Detection



**Figure 179.** Asynchronous Character Reception

Example: 8-bit, Parity Enabled



## Synchronous Receiver

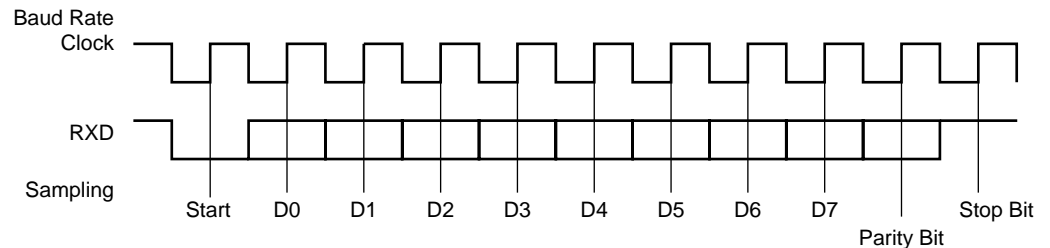
In synchronous mode (SYNC = 1), the receiver samples the RXD signal on each rising edge of the Baud Rate Clock. If a low level is detected, it is considered as a start. All data bits, the parity bit and the stop bits are sampled and the receiver waits for the next start bit. Synchronous mode operations provide a high speed transfer capability.

Configuration fields and bits are the same as in asynchronous mode.

Figure 180 illustrates a character reception in synchronous mode.

**Figure 180.** Synchronous Mode Character Reception

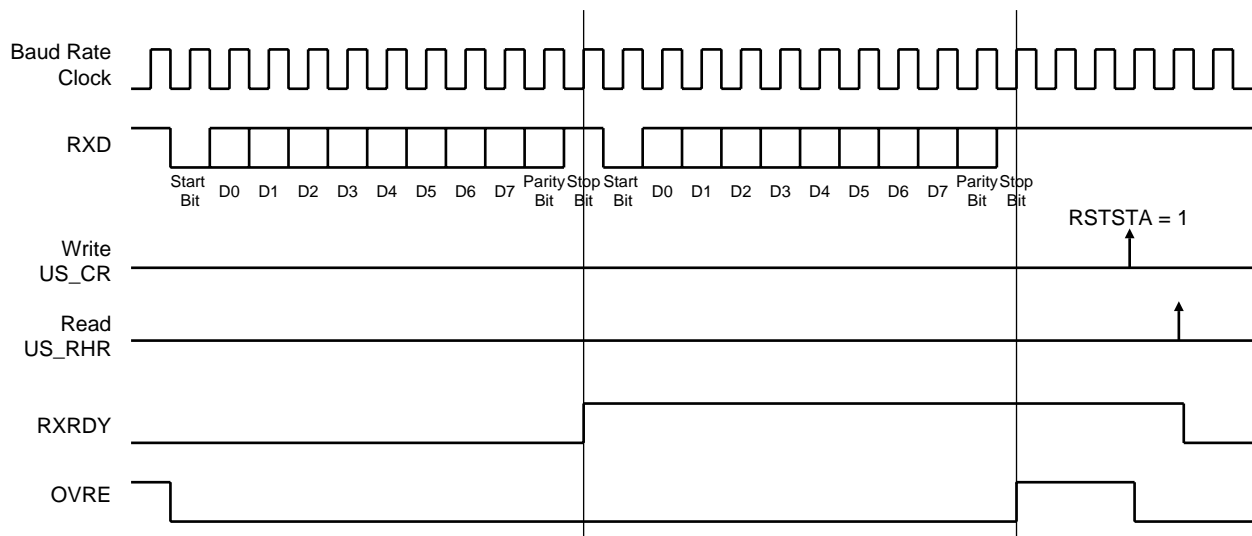
Example: 8-bit, Parity Enabled 1 Stop



## Receiver Operations

When a character reception is completed, it is transferred to the Receive Holding Register (US\_RHR) and the RXRDY bit in the Status Register (US\_CSR) rises. If a character is completed while the RXRDY is set, the OVRE (Overrun Error) bit is set. The last character is transferred into US\_RHR and overwrites the previous one. The OVRE bit is cleared by writing the Control Register (US\_CR) with the RSTSTA (Reset Status) bit at 1.

**Figure 181.** Receiver Status



## Parity

The USART supports five parity modes selected by programming the PAR field in the Mode Register (US\_MR). The PAR field also enables the Multidrop mode, which is discussed in a separate paragraph. Even and odd parity bit generation and error detection are supported.

If even parity is selected, the parity generator of the transmitter drives the parity bit at 1 if a number of 1s in the character data bit is even, and at 0 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If the odd parity is selected, the parity generator of the

transmitter drives the parity bit at 0 if a number of 1s in the character data bit is even, and at 1 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If the mark parity is used, the parity generator of the transmitter drives the parity bit at 1 for all characters. The receiver parity checker reports an error if the parity bit is sampled at 0. If the space parity is used, the parity generator of the transmitter drives the parity bit at 0 for all characters. The receiver parity checker reports an error if the parity bit is sampled at 1. If parity is disabled, the transmitter does not generate any parity bit and the receiver does not report any parity error.

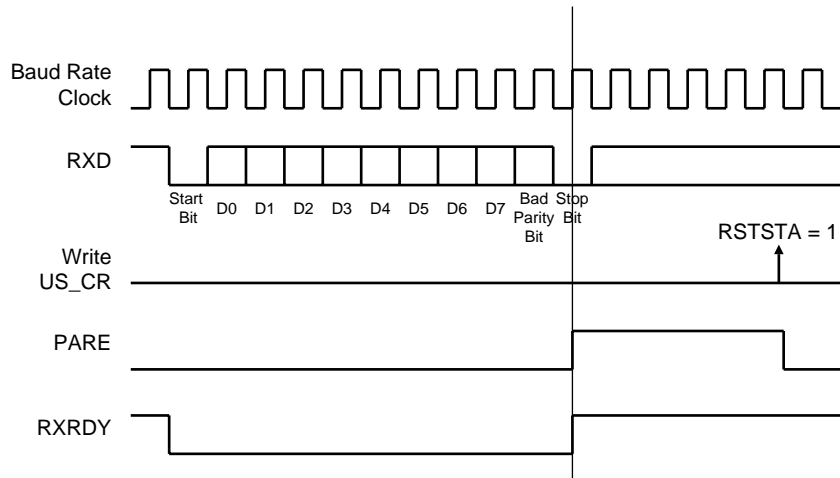
Table 78 shows an example of the parity bit for the character 0x41 (character ASCII “A”) depending on the configuration of the USART. Because there are two bits at 1, 1 bit is added when a parity is odd, or 0 is added when a parity is even. I

**Table 78.** Parity Bit Examples

| Character | Hexa | Binary    | Parity Bit | ParityMode |
|-----------|------|-----------|------------|------------|
| A         | 0x41 | 0100 0001 | 1          | Odd        |
| A         | 0x41 | 0100 0001 | 0          | Even       |
| A         | 0x41 | 0100 0001 | 1          | Mark       |
| A         | 0x41 | 0100 0001 | 0          | Space      |
| A         | 0x41 | 0100 0001 | None       | None       |

When the receiver detects a parity error, it sets the PARE (Parity Error) bit in the Channel Status Register (US\_CSR). The PARE bit can be cleared by writing the Control Register (US\_CR) with the RSTSTA bit at 1. Figure 182 illustrates the parity bit status setting and clearing.

**Figure 182.** Parity Error



### Multi-drop Mode

If the PAR field in the Mode Register (US\_MR) is programmed to the value 0x3, the USART runs in Multi-drop mode. This mode differentiates the data characters and the address characters. Data is transmitted with the parity bit at 0 and addresses are transmitted with the parity bit at 1.

If the USART is configured in multi-drop mode, the receiver sets the PARE parity error bit when the parity bit is high and the transmitter is able to send a character with the parity bit high when the Control Register is written with the SENDA bit at 1.



To handle parity error, the PARE bit is cleared when the Control Register is written with the bit RSTSTA at 1.

The transmitter sends an address byte (parity bit set) when SENDA is written to US\_CR. In this case, the next byte written to US\_THR is transmitted as an address. Any character written in US\_THR without having written the command SENDA is transmitted normally with the parity at 0.

## Transmitter Timeguard

The timeguard feature enables the USART interface with slow remote devices.

The timeguard function enables the transmitter to insert an idle state on the TXD line between two characters. This idle state actually acts as a long stop bit.

The duration of the idle state is programmed in the TG field of the Transmitter Timeguard Register (US\_TTGR). When this field is programmed at zero no timeguard is generated. Otherwise, the transmitter holds a high level on TXD after each transmitted byte during the number of bit periods programmed in TG in addition to the number of stop bits.

As illustrated in Figure 183, the behavior of TXRDY and TXEMPTY status bits is modified by the programming of a timeguard. TXRDY rises only when the start bit of the next character is sent, and thus remains at 0 during the timeguard transmission if a character has been written in US\_THR. TXEMPTY remains low until the timeguard transmission is completed as the timeguard is part of the current character being transmitted.

**Figure 183.** Timeguard Operations

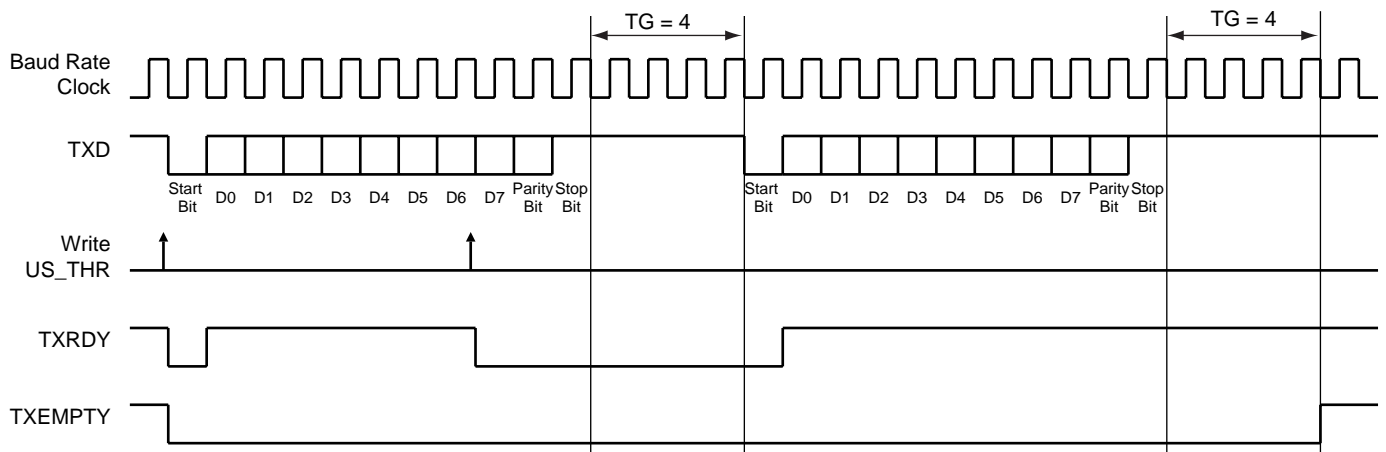


Table 79 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the Baud Rate.

**Table 79.** Maximum Timeguard Length Depending on Baud Rate

| Baud Rate | Bit time | Timeguard |
|-----------|----------|-----------|
| bit/sec   | µs       | ms        |
| 1 200     | 833      | 212.50    |
| 9 600     | 104      | 26.56     |
| 14400     | 69.4     | 17.71     |
| 19200     | 52.1     | 13.28     |
| 28800     | 34.7     | 8.85      |
| 33400     | 29.9     | 7.63      |

**Table 79.** Maximum Timeguard Length Depending on Baud Rate (Continued)

| Baud Rate | Bit time | Timeguard |
|-----------|----------|-----------|
| 56000     | 17.9     | 4.55      |
| 57600     | 17.4     | 4.43      |
| 115200    | 8.7      | 2.21      |

### Receiver Time-out

The Receiver Time-out provides support in handling variable-length frames. This feature detects an idle condition on the RXD line. When a time-out is detected, the bit TIMEOUT in the Channel Status Register (US\_CSR) rises and can generate an interrupt, thus indicating to the driver an end of frame.

The time-out delay period (during which the receiver waits for a new character) is programmed in the TO field of the Receiver Time-out Register (US\_RTOR). If the TO field is programmed at 0, the Receiver Time-out is disabled and no time-out is detected. The TIMEOUT bit in US\_CSR remains at 0. Otherwise, the receiver loads a 16-bit counter with the value programmed in TO. This counter is decremented at each bit period and reloaded each time a new character is received. If the counter reaches 0, the TIMEOUT bit in the Status Register rises.

The user can either:

- Obtain an interrupt when a time-out is detected after having received at least one character. This is performed by writing the Control Register (US\_CR) with the STTTO (Start Time-out) bit at 1.
- Obtain a periodic interrupt while no character is received. This is performed by writing US\_CR with the RETTO (Reload and Start Time-out) bit at 1.

If STTTO is performed, the counter clock is stopped until a first character is received. The idle state on RXD before the start of the frame does not provide a time out. This prevents having to obtain a periodic interrupt and enables a wait of the end of frame when the idle state on RXD is detected.

If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

Figure 184 shows the block diagram of the Receiver Time out feature.

**Figure 184.** Receiver Time-out Block Diagram

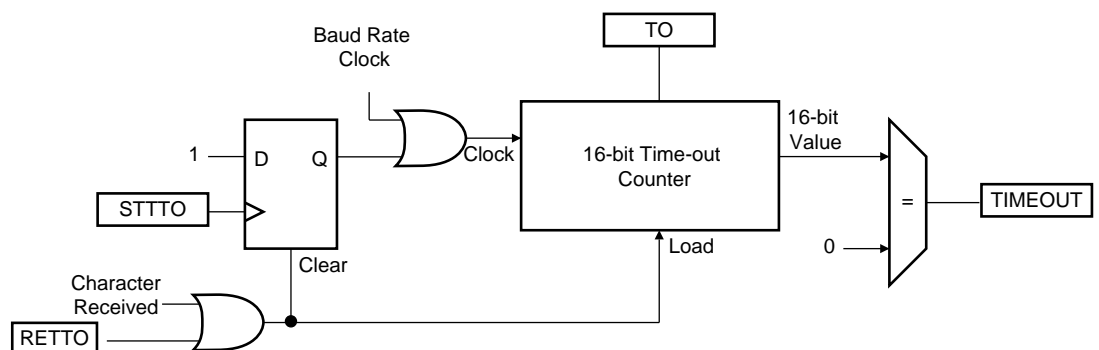


Table 80 gives the maximum time-out period for some standard baud rates.

**Table 80.** Maximum Time-out Period

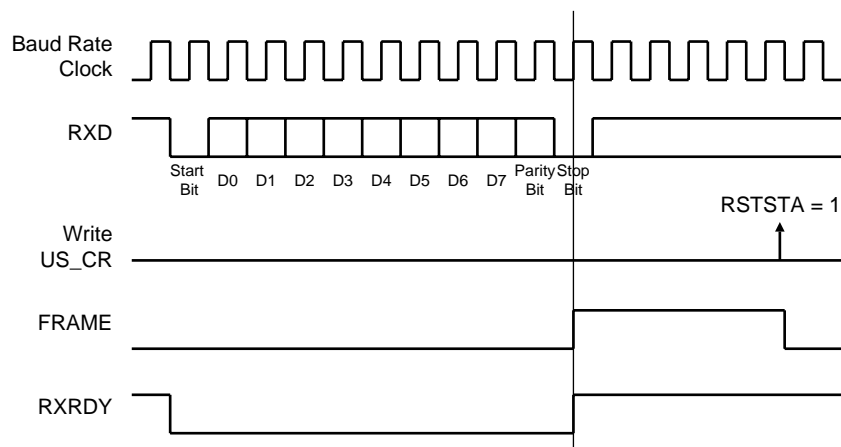
| Baud Rate | Bit Time | Time -out |
|-----------|----------|-----------|
| bit/sec   | µs       | ms        |
| 600       | 1 667    | 109 225   |
| 1 200     | 833      | 54 613    |
| 2 400     | 417      | 27 306    |
| 4 800     | 208      | 13 653    |
| 9 600     | 104      | 6 827     |
| 14400     | 69       | 4 551     |
| 19200     | 52       | 3 413     |
| 28800     | 35       | 2 276     |
| 33400     | 30       | 1 962     |
| 56000     | 18       | 1 170     |
| 57600     | 17       | 1 138     |
| 200000    | 5        | 328       |

## Framing Error

The receiver is capable of detecting framing errors. A framing error happens when the stop bit of a received character is detected at level 0. This can occur if the receiver and the transmitter are fully desynchronized.

A framing error is reported on the FRAME bit of the Channel Status Register (US\_CSR). The FRAME bit is asserted in the middle of the stop bit as soon as the framing error is detected. It is cleared by writing the Control Register (US\_CR) with the RSTSTA bit at 1.

**Figure 185.** Framing Error Status



## Transmit Break

The user can request the transmitter to generate a break condition on the TXD line. A break condition drives the TXD line low during at least one complete character. It appears the same as a 0x00 character sent with the parity and the stop bits at 0. However, the transmitter holds the TXD line at least during one character until the user requests the break condition to be removed.

A break is transmitted by writing the Control Register (US\_CR) with the STTBK bit at 1. This can be performed at any time, either while the transmitter is empty (no character in either the Shift Register or in US\_THR) or when a character is being transmitted. If a break is requested while a character is being shifted out, the character is first completed before the TXD line is held low.

Once STTBK command is requested further STTBK commands are ignored until the end of the break is completed.

The break condition is removed by writing US\_CR with the STPBK bit at 1. If the STPBK is requested before the end of the minimum break duration (one character, including start, data, parity and stop bits), the transmitter ensures that the break condition completes.

The transmitter considers the break as though it is a character, i.e. the STTBK and STPBK commands are taken into account only if the TXRDY bit in US\_CSR is at 1 and the start of the break condition clears the TXRDY and TXEMPTY bits as if a character is processed.

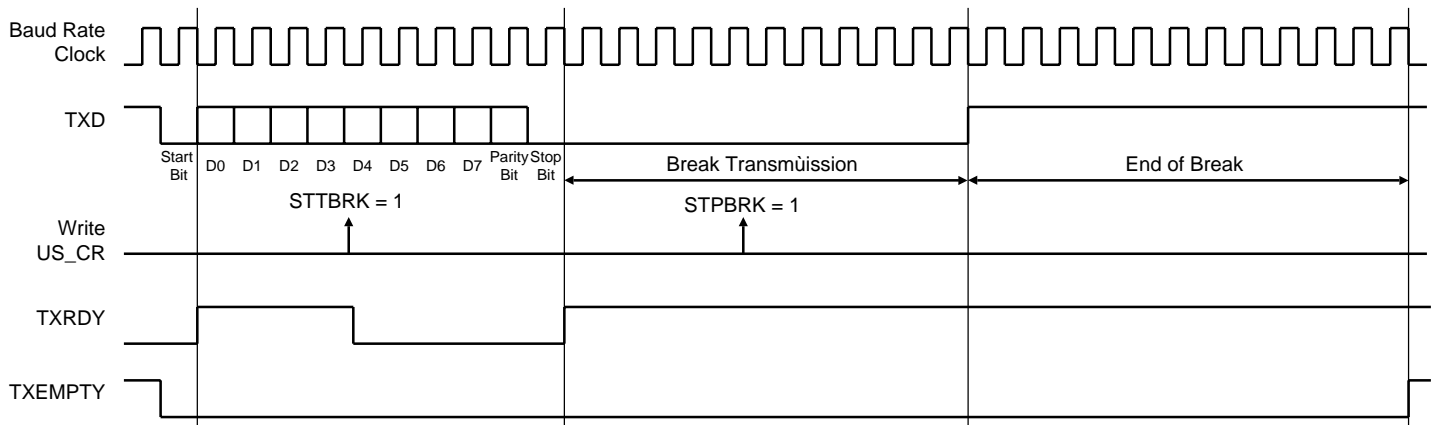
Writing US\_CR with the both STTBK and STPBK bits at 1 can lead to an unpredictable result. All STPBK commands requested without a previous STTBK command are ignored. A byte written into the Transmit Holding Register while a break is pending, but not started, is ignored.

After the break condition, the transmitter returns the TXD line to 1 for a minimum of 12 times. Thus, the transmitter ensures that the remote receiver detects correctly the end of break and the start of the next character. If the timeguard is programmed with a value higher than 12, the TXD line is held high for the timeguard period.

After holding the TXD line for this period, the transmitter resumes normal operations.

Figure 186 illustrates the effect of both the Start Break (STTBK) and Stop Break (STP BRK) commands on the TXD line.

**Figure 186. Break Transmission**



**Receive Break**

The receiver detects a break condition when all data, parity and stop bits are low. This corresponds to detecting a framing error with data at 0x00, but FRAME remains low.

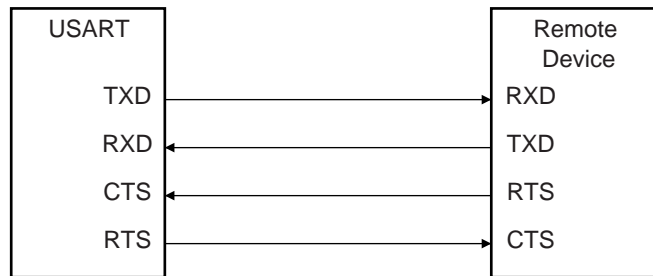
When the low stop bit is detected, the receiver asserts the RXBRK bit in US\_CSR. This bit may be cleared by writing the Control Register (US\_CR) with the bit RSTSTA at 1.

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 one sample at high level in synchronous operating mode. The end of break detection also asserts the RXBRK bit.

## Hardware Handshaking

The USART features a hardware handshaking out-of-band flow control. The RTS and CTS pins are used to connect with the remote device, as shown in Figure 187.

**Figure 187.** Connection with a Remote Device for Hardware Handshaking



Setting the USART to operate with hardware handshaking is performed by writing the USART\_MODE field in the Mode Register (US\_MR) to the value 0x2.

The USART behavior when hardware handshaking is enabled is the same as the behavior in standard synchronous or asynchronous mode, except that the receiver drives the RTS pin as described below and the level on the CTS pin modifies the behavior of the transmitter as described below. Using this mode requires using the PDC channel for reception. The transmitter can handle hardware handshaking in any case.

Figure 188 shows how the receiver operates if hardware handshaking is enabled. The RTS pin is driven high if the receiver is disabled and if the status RXBUFF (Receive Buffer Full) coming from the PDC channel is high. Normally, the remote device does not start transmitting while its CTS pin (driven by RTS) is high. As soon as the Receiver is enabled, the RTS falls, indicating to the remote device that it can start transmitting. Defining a new buffer to the PDC clears the status bit RXBUFF and, as a result, asserts the pin RTS low.

**Figure 188.** Receiver Behavior when Operating with Hardware Handshaking

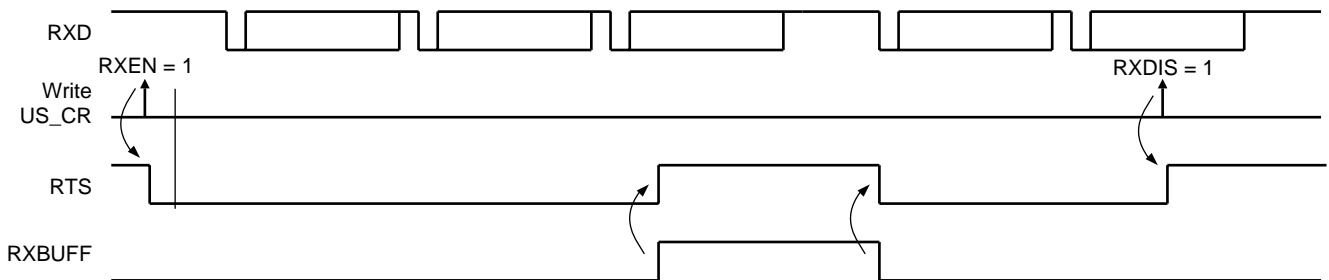
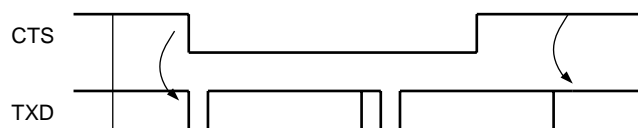


Figure 189 shows how the transmitter operates if hardware handshaking is enabled. The CTS pin disables the transmitter. If a character is being processing, the transmitter is disabled only after the completion of the current character and transmission of the next character happens as soon as the pin CTS falls.

**Figure 189.** Transmitter Behavior when Operating with Hardware Handshaking



## ISO7816 Mode

The USART features an ISO7816-compatible operating mode. This mode permits interfacing with smart cards and Security Access Modules (SAM) communicating through an ISO7816 link. Both T = 0 and T = 1 protocols defined by the ISO7816 specification are supported.

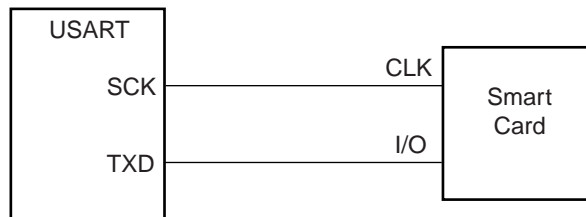
Setting the USART in ISO7816 mode is performed by writing the USART\_MODE field in the Mode Register (US\_MR) to the value 0x4 for protocol T = 0 and to the value 0x5 for protocol T = 1.

## ISO7816 Mode overview

The ISO7816 is a half duplex communication on only one bidirectional line. The baud rate is determined by a division of the clock provided to the remote device (see “Baud Rate Generator” on page 400).

The USART connects to a smart card, as shown in Figure 190. The TXD line becomes bidirectional and the Baud Rate Generator feeds the ISO7816 clock on the SCK pin. As the TXD pin becomes bidirectional, its output remains driven by the output of the transmitter but only when the transmitter is active while its input is directed to the input of the receiver. The USART is considered as the master of the communication as it generates the clock.

**Figure 190.** Connection of a Smart Card to the USART



When operating in ISO7816, either in T = 0 or T = 1 modes, the character format is fixed. The configuration is 8 data bits, even parity and 1 or 2 stop bits, regardless of the values programmed in the CHRL, MODE9, PAR and CHMODE fields. MSBF can be used to transmit LSB or MSB first.

The USART cannot operate concurrently in both receiver and transmitter modes as the communication is unidirectional at a time. It has to be configured according to the required mode by enabling or disabling either the receiver or the transmitter as desired. Enabling both the receiver and the transmitter at the same time in ISO7816 mode may lead to unpredictable results.

The ISO7816 specification defines an inverse transmission format. Data bits of the character must be transmitted on the I/O line at their negative value. The USART does not support this format and the user has to perform an exclusive OR on the data before writing it in the Transmit Holding Register (US\_THR) or after reading it in the Receive Holding Register (US\_RHR).

## Protocol T = 0

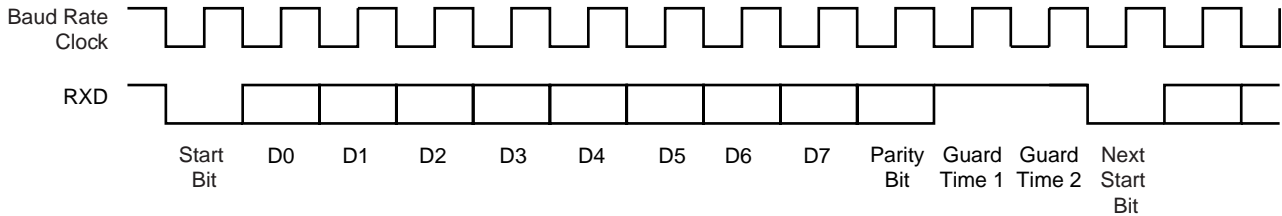
In T = 0 protocol, a character is made up of one start bit, eight data bits, one parity bit and one guard time, which lasts two bit times. The transmitter shifts out the bits and does not drive the I/O line during the guard time.

If no parity error is detected, the I/O line remains at 1 during the guard time and the transmitter can continue with the transmission of the next character, as shown in Figure 191.

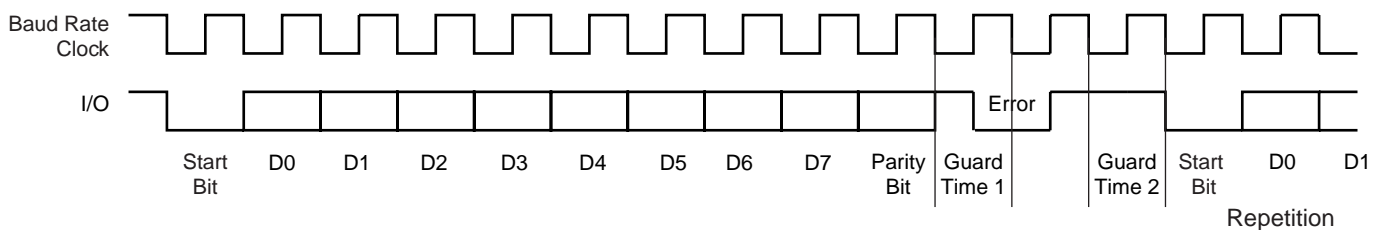
If a parity error is detected by the receiver, it drives the I/O line at 0 during the guard time, as shown in Figure 192. This error bit is also named NACK, for Non Acknowledge. In this case, the character lasts 1 bit time more, as the guard time length is the same and is added to the error bit time which lasts 1 bit time.

When the USART is the receiver and it detects an error, it does not load the erroneous character in the Receive Holding Register (US\_RHR). It appropriately sets the PARE bit in the Status Register (US\_SR) so that the software can handle the error.

**Figure 191.** T = 0 Protocol without Parity Error



**Figure 192.** T = 0 Protocol with Parity Error



**Receive Error Counter**

The USART receiver also records the total number of errors. This can be read in the Number of Error (US\_NER) register. The NB\_ERRORS field can record up to 255 errors. Reading US\_NER automatically clears the NB\_ERRORS field.

**Receive NACK Inhibit**

The USART can also be configured to inhibit an error. This can be achieved by setting the INACK bit in the Mode Register (US\_MR). If INACK is at 1, no error signal is driven on the I/O line even if a parity bit is detected, but the INACK bit is set in the Status Register (US\_SR). The INACK bit can be cleared by writing the Control Register (US\_CR) with the RSTNACK bit at 1.

Moreover, if INACK is set, the erroneous received character is stored in the Receive Holding Register, as if no error occurred. However, the RXRDY bit does not raise.

**Transmit Character Repetition**

When the USART is transmitting a character and gets a NACK, it can automatically repeat the character before moving on to the next one. Repetition is enabled by writing the MAX\_ITERATION field in the Mode Register (US\_MR) at a value higher than 0. Each character can be transmitted up to eight times; the first transmission plus seven repetitions.

If MAX\_ITERATION does not equal zero, the USART repeats the character as many times as the value loaded in MAX\_ITERATION.

When the USART repetition number reaches MAX\_ITERATION, the ITERATION bit is set in the Channel Status Register (US\_CSR). If the repetition of the character is acknowledged by the receiver, the repetitions are stopped and the iteration counter is cleared.

The ITERATION bit in US\_CSR can be cleared by writing the Control Register with the RSIT bit at 1.

**Disable Successive Receive NACK**

The receiver can limit the number of successive NACKs sent back to the remote transmitter. This is programmed by setting the bit DSNACK in the Mode Register (US\_MR). The maximum number of NACK transmitted is programmed in the MAX\_ITERATION field. As soon as

MAX\_ITERATION is reached, the character is considered as correct, an acknowledge is sent on the line and the ITERATION bit in the Channel Status Register is set.

**Protocol T = 1**

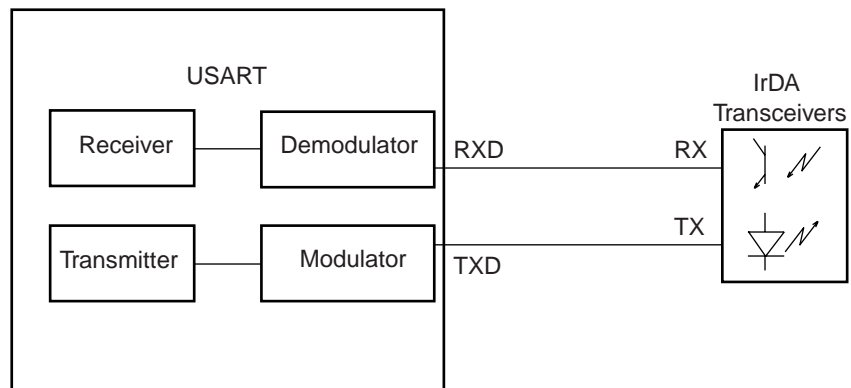
When operating in ISO7816 protocol T = 1, the transmission is similar to an asynchronous format with only one stop bit. The parity is generated when transmitting and checked when receiving. Parity error detection sets the PARE bit in the Channel Status Register (US\_CSR).

**IrDA Mode**

The USART features an IrDA mode supplying half-duplex point-to-point wireless communication. It embeds the modulator and demodulator which allows a glueless connection to the infrared transceivers, as shown in Figure 193. The modulator and demodulator are compliant with the IrDA specification version 1.1 and support data transfer speeds ranging from 2,4 Kbps to 115,2 Kbps.

The USART IrDA mode is enabled by setting the USART\_MODE field in the Mode Register (US\_MR) to the value 0x8. The IrDA Filter Register (US\_IF) allows configuring the demodulator filter. The USART transmitter and receiver operate in a normal asynchronous mode and all parameters are accessible. Note that the modulator and the demodulator are activated.

**Figure 193.** Connection to IrDA Transceivers



The receiver and the transmitter must be enabled or disabled according to the direction of the transmission to be managed.

**IrDA Modulation**

For baud rates up to and including 115.2 Kbits/sec, the RZI modulation scheme is used. "0" is represented by a light pulse of 3/16th of a bit time. Some examples of signal pulse duration are shown in Table 81..

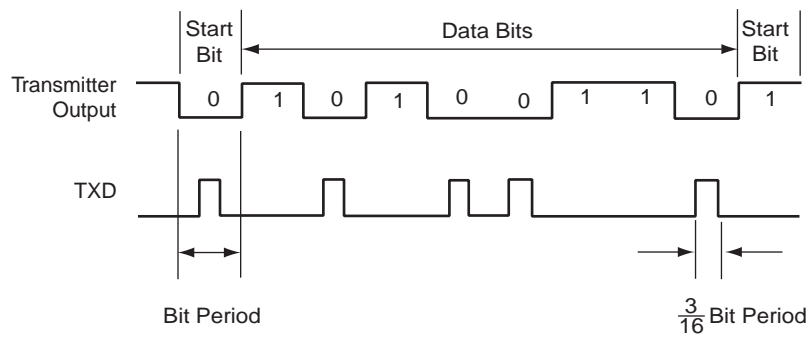
**Table 81.** IrDA Pulse Duration

| Baud Rate  | Pulse Duration (3/16) |
|------------|-----------------------|
| 2.4 Kb/s   | 78.13 μs              |
| 9.6 Kb/s   | 19.53 μs              |
| 19.2 Kb/s  | 9.77 μs               |
| 38.4 Kb/s  | 4.88 μs               |
| 57.6 Kb/s  | 3.26 μs               |
| 115.2 Kb/s | 1.63 μs               |

Figure 194 shows an example of character transmission.



Figure 194. IrDA Modulation



IrDA Baud Rate

Table 82 gives some examples of CD values, baud rate error and pulse duration. Note that the requirement on the maximum acceptable error of +/- 1.87% must be met.

Table 82. IrDA Baud Rate Error

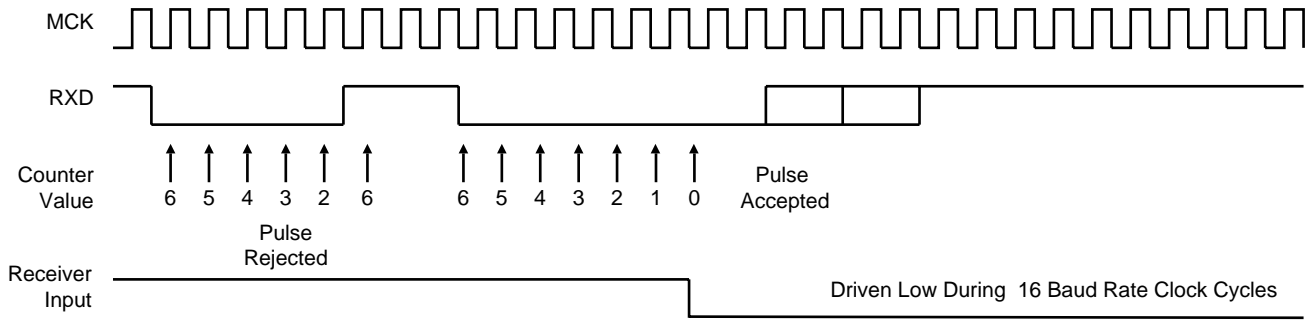
| Peripheral Clock | Baud rate | CD  | Baud rate Error | Pulse time |
|------------------|-----------|-----|-----------------|------------|
| 3 686 400        | 115 200   | 2   | 0.00%           | 1.63       |
| 20 000 000       | 115 200   | 11  | 1.38%           | 1.63       |
| 32 768 000       | 115 200   | 18  | 1.25%           | 1.63       |
| 40 000 000       | 115 200   | 22  | 1.38%           | 1.63       |
| 3 686 400        | 57 600    | 4   | 0.00%           | 3.26       |
| 20 000 000       | 57 600    | 22  | 1.38%           | 3.26       |
| 32 768 000       | 57 600    | 36  | 1.25%           | 3.26       |
| 40 000 000       | 57 600    | 43  | 0.93%           | 3.26       |
| 3 686 400        | 38 400    | 6   | 0.00%           | 4.88       |
| 20 000 000       | 38 400    | 33  | 1.38%           | 4.88       |
| 32 768 000       | 38 400    | 53  | 0.63%           | 4.88       |
| 40 000 000       | 38 400    | 65  | 0.16%           | 4.88       |
| 3 686 400        | 19 200    | 12  | 0.00%           | 9.77       |
| 20 000 000       | 19 200    | 65  | 0.16%           | 9.77       |
| 32 768 000       | 19 200    | 107 | 0.31%           | 9.77       |
| 40 000 000       | 19 200    | 130 | 0.16%           | 9.77       |
| 3 686 400        | 9 600     | 24  | 0.00%           | 19.53      |
| 20 000 000       | 9 600     | 130 | 0.16%           | 19.53      |
| 32 768 000       | 9 600     | 213 | 0.16%           | 19.53      |
| 40 000 000       | 9 600     | 260 | 0.16%           | 19.53      |
| 3 686 400        | 2 400     | 96  | 0.00%           | 78.13      |
| 20 000 000       | 2 400     | 521 | 0.03%           | 78.13      |
| 32 768 000       | 2 400     | 853 | 0.04%           | 78.13      |

### IrDA Demodulator

The demodulator is based on the IrDA Receive filter comprised of an 8-bit down counter which is loaded with the value programmed in US\_IF. When a falling edge is detected on the RXD pin, the Filter Counter starts counting down at the Master Clock (MCK) speed. If a rising edge is detected on the RXD pin, the counter stops and is reloaded with US\_IF. If no rising edge is detected when the counter reaches 0, the input of the receiver is driven low during one bit time.

Figure 195 illustrates the operations of the IrDA demodulator.

**Figure 195.** IrDA Demodulator Operations

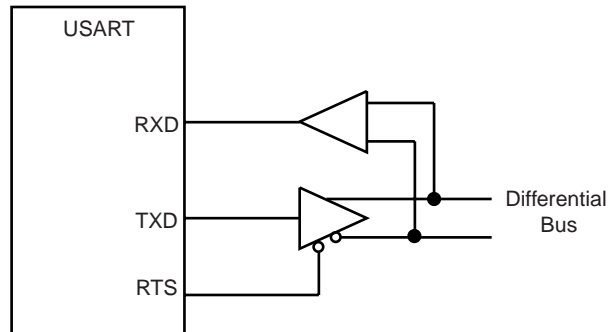


As the IrDA mode uses the same logic as the ISO7816, note that the FI\_DI\_RATIO field in US\_FIDI must be set to a value higher than 0 in order to assure IrDA communications operate correctly.

**RS485 Mode**

The USART features the RS485 mode to enable line driver control. While operating in RS485 mode, the USART behaves as though in asynchronous or synchronous mode and configuration of all the parameters are possible. The difference is that the RTS pin is driven low when the transmitter is operating. The behavior of the RTS pin is controlled by the TXEMPTY bit. A typical connection of the USART to a RS485 bus is shown in Figure 196.

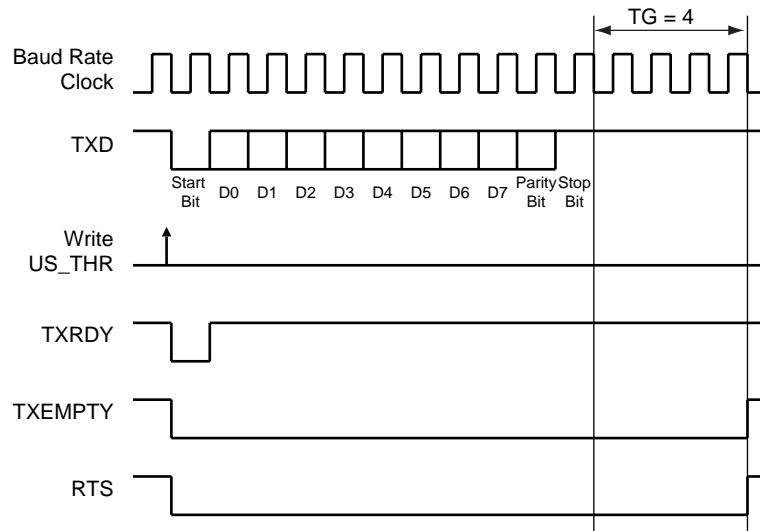
**Figure 196.** Typical Connection to a RS485 bus.



The USART is set in RS485 mode by programming the USART\_MODE field in the Mode Register (US\_MR) to the value 0x1.

The RTS pin is at a level inverse of the TXEMPTY bit. Significantly, the RTS pin remains low when a timeguard is programmed so that the line can remain driven after the last character completion. Figure 197 gives an example of the RTS waveform during a character transmission when the timeguard is enabled.

**Figure 197.** Example of RTS Drive with Timeguard



## Modem Mode

The USART features modem mode, which enables control of the signals: DTR (Data Terminal Ready), DSR (Data Set Ready), RTS (Request to Send), CTS (Clear to Send), DCD (Data Carrier Detect) and RI (Ring Indicator). While operating in modem mode, the USART behaves as a DTE (Data Terminal Equipment) as it drives DTR and RTS and can detect level change on DSR, DCD, CTS and RI.

Setting the USART in modem mode is performed by writing the USART\_MODE field in the Mode Register (US\_MR) to the value 0x3. While operating in modem mode the USART behaves as though in asynchronous mode and all the parameter configurations are available.

Table 83 gives the correspondence of the USART signals with modem connection standards.

**Table 83.** Circuit References

| USART pin | V24 | CCITT | Direction              |
|-----------|-----|-------|------------------------|
| TXD       | 2   | 103   | From terminal to modem |
| RTS       | 4   | 105   | From terminal to modem |
| DTR       | 20  | 108.2 | From terminal to modem |
| RXD       | 3   | 104   | From modem to terminal |
| CTS       | 5   | 106   | From terminal to modem |
| DSR       | 6   | 107   | From terminal to modem |
| DCD       | 8   | 109   | From terminal to modem |
| RI        | 22  | 125   | From terminal to modem |

The control of the RTS and DTR output pins is performed by writing the Control Register (US\_CR) with the RTSDIS, RTSEN, DTRDIS and DTREN bits respectively at 1. The disable command forces the corresponding pin to its inactive level, i.e. high. The enable commands force the corresponding pin to its active level, i.e. low.

The level changes are detected on the RI, DSR, DCD and CTS pins. If an input change is detected, the RIIC, DSRIC, DCDIC and CTSIC bits in the Channel Status Register (US\_CSR) are set respectively and can trigger an interrupt. The status is automatically cleared when US\_CSR is read. Furthermore, the CTS automatically disables the transmitter when it is detected at its inactive state. If a character is being transmitted when the CTS rises, the character transmission is completed before the transmitter is actually disabled.

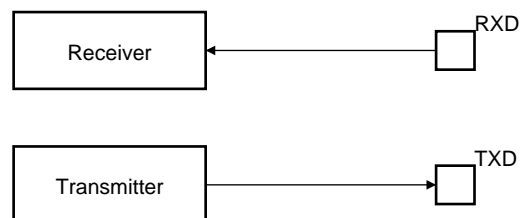
## Test Modes

The USART can be programmed to operate in three different test modes. The internal loopback capability allows on-board diagnostics. In the loopback mode the USART interface pins are disconnected or not and reconfigured for loopback internally or externally.

## Normal Mode

As a reminder, the normal mode simply connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

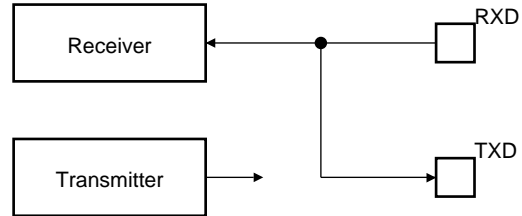
**Figure 198.** Normal Mode Configuration



**Automatic Echo**

Automatic echo mode allows bit-by-bit retransmission. When a bit is received on the RXD pin, it is sent to the TXD pin, as shown in Figure 199. Programming the transmitter has no effect on the TXD pin. The RXD pin is still connected to the receiver input, thus the receiver remains active.

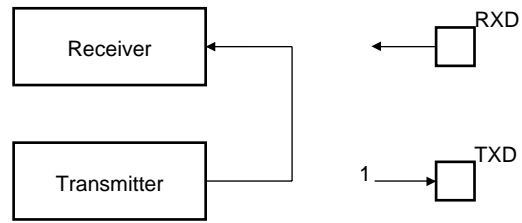
**Figure 199.** Automatic Echo



**Local Loopback**

The local loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in Figure 200. The TXD and RXD pins are not used. The RXD pin has no effect on the receiver and the TXD pin is continuously driven high, as in idle state.

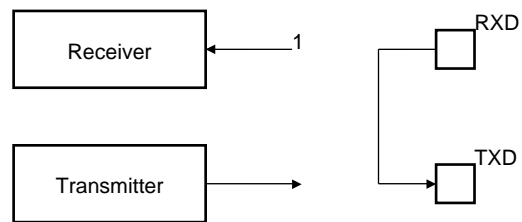
**Figure 200.** Local Loopback



**Remote Loopback**

Remote loopback mode directly connects the RXD pin to the TXD pin, as shown in Figure 201. The transmitter and the receiver are disabled and have no effect. This mode allows bit-by-bit retransmission.

**Figure 201.** Remote Loopback



## USART User Interface

**Table 84.** USART Memory Map

| Offset               | Register                       | Name    | Access     | Reset State |
|----------------------|--------------------------------|---------|------------|-------------|
| 0x0000               | Control Register               | US_CR   | Write-only | –           |
| 0x0004               | Mode Register                  | US_MR   | Read/Write | –           |
| 0x0008               | Interrupt Enable Register      | US_IER  | Write-only | –           |
| 0x000C               | Interrupt Disable Register     | US_IDR  | Write-only | –           |
| 0x0010               | Interrupt Mask Register        | US_IMR  | Read-only  | 0           |
| 0x0014               | Channel Status Register        | US_CSR  | Read-only  | –           |
| 0x0018               | Receiver Holding Register      | US_RHR  | Read-only  | 0           |
| 0x001C               | Transmitter Holding Register   | US_THR  | Write-only | –           |
| 0x0020               | Baud Rate Generator Register   | US_BRGR | Read/Write | 0           |
| 0x0024               | Receiver Time-out Register     | US_RTOR | Read/Write | 0           |
| 0x0028               | Transmitter Timeguard Register | US_TTGR | Read/Write | 0           |
| 0x2C<br>to<br>0x3C   | Reserved                       | –       | –          | –           |
| 0x0040               | FI DI Ratio Register           | US_FIDI | Read/Write | 0x174       |
| 0x0044               | Number of Errors Register      | US_NER  | Read-only  | –           |
| 0x0048               | Reserved                       | –       | –          | –           |
| 0x004C               | IrDA Filter Register           | US_IF   | Read/Write | 0           |
| 0x5C<br>to<br>0xFC   | Reserved                       | –       | –          | –           |
| 0x100<br>to<br>0x128 | Reserved for PDC Registers     | –       | –          | –           |

## USART Control Register

**Name:** US\_CR

**Access Type:** Write-only

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

- **RSTRX: Reset Receiver**

0 = No effect.

1 = Resets the receiver.

- **RSTTX: Reset Transmitter**

0 = No effect.

1 = Resets the transmitter.

- **RXEN: Receiver Enable**

0 = No effect.

1 = Enables the receiver, if RXDIS is 0.

- **RXDIS: Receiver Disable**

0 = No effect.

1 = Disables the receiver.

- **TXEN: Transmitter Enable**

0 = No effect.

1 = Enables the transmitter if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0 = No effect.

1 = Disables the transmitter.

- **RSTSTA: Reset Status Bits**

0 = No effect.

1 = Resets the status bits PARE, FRAME, OVRE and RXBRK in the US\_CSR.

- **STTBRK: Start Break**

0 = No effect.

1 = Starts transmission of a break after the characters present in US\_THR and the Transmit Shift Register have been transmitted. No effect if a break is already being transmitted.

- **STPBRK: Stop Break**

0 = No effect.

1 = Stops transmission of the break after a minimum of one character length and transmits a high level during 12-bit periods. No effect if no break is being transmitted.

- **STTTO: Start Time-out**

0 = No effect

1 = Starts waiting for a character before clocking the time-out counter.

- **SENDA: Send Address**

0 = No effect.

1 = In Multi-drop Mode only, the next character written to the US\_THR is sent with the address bit set.

- **RSTIT: Reset Iterations**

0 = No effect.

1 = Resets ITERATION in US\_CSR. No effect if the ISO7816 is not enabled.

- **RSTNACK: Reset Non Acknowledge**

0 = No effect

1 = Resets NACK in US\_CSR.

- **RETTO: Rearm Time-out**

0 = No effect

1 = Restart Time-out

- **DTREN: Data Terminal Ready Enable**

0 = No effect.

1 = Drives the pin DTR at 0.

- **DTRDIS: Data Terminal Ready Disable**

0 = No effect.

1 = Drives the pin DTR to 1.

- **RTSEN: Request to Send Enable**

0 = No effect.

1 = Drives the pin RTS to 0.

- **RTSDIS: Request to Send Disable**

0 = No effect.

1 = Drives the pin RTS to 1.



## USART Mode Register

Name: US\_MR

Access Type: Read/Write

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

### • USART\_MODE

| USART_MODE |   |   |   | Mode of the USART       |
|------------|---|---|---|-------------------------|
| 0          | 0 | 0 | 0 | Normal                  |
| 0          | 0 | 0 | 1 | RS485                   |
| 0          | 0 | 1 | 0 | Hardware Handshaking    |
| 0          | 0 | 1 | 1 | Modem                   |
| 0          | 1 | 0 | 0 | ISO7816 Protocol: T = 0 |
| 0          | 1 | 0 | 1 | Reserved                |
| 0          | 1 | 1 | 0 | ISO7816 Protocol: T = 1 |
| 0          | 1 | 1 | 1 | Reserved                |
| 1          | 0 | 0 | 0 | IrDA                    |
| 1          | 1 | x | x | Reserved                |

### • USCLKS: Clock Selection

| USCLKS |   | Selected Clock |
|--------|---|----------------|
| 0      | 0 | MCK            |
| 0      | 1 | MCK / DIV      |
| 1      | 0 | Reserved       |
| 1      | 1 | SCK            |

### • CHRL: Character Length.

| CHRL |   | Character Length |
|------|---|------------------|
| 0    | 0 | 5 bits           |
| 0    | 1 | 6 bits           |
| 1    | 0 | 7 bits           |
| 1    | 1 | 8 bits           |

- **SYNC: Synchronous Mode Select**

0 = USART operates in Asynchronous Mode.

1 = USART operates in Synchronous Mode

- **PAR: Parity Type**

| PAR |   |   | Parity Type                |
|-----|---|---|----------------------------|
| 0   | 0 | 0 | Even parity                |
| 0   | 0 | 1 | Odd parity                 |
| 0   | 1 | 0 | Parity forced to 0 (Space) |
| 0   | 1 | 1 | Parity forced to 1 (Mark)  |
| 1   | 0 | x | No parity                  |
| 1   | 1 | x | Multi-drop mode            |

- **NBSTOP: Number of Stop Bits**

| NBSTOP |   | Asynchronous (SYNC = 0) | Synchronous (SYNC = 1) |
|--------|---|-------------------------|------------------------|
| 0      | 0 | 1 stop bit              | 1 stop bit             |
| 0      | 1 | 1.5 stop bits           | Reserved               |
| 1      | 0 | 2 stop bits             | 2 stop bits            |
| 1      | 1 | Reserved                | Reserved               |

- **CHMODE: Channel Mode**

| CHMODE |   | Mode Description  |
|--------|---|---|
| 0      | 0 | Normal Mode   |
| 0      | 1 | Automatic Echo. Receiver input is connected to the TXD pin.             |
| 1      | 0 | Local Loopback. Transmitter output is connected to the Receiver Input.. |
| 1      | 1 | Remote Loopback. RXD pin is internally connected to the TXD pin.        |

- **MSBF: Bit Order**

0 = Least Significant Bit is sent/received first.

1 = Most Significant Bit is sent/received first.

- **MODE9: 9-bit Character Length**

0 = CHRL defines character length.

1 = 9-bit character length.

- **CKLO: Clock Output Select**

0 = The USART does not drive the SCK pin.

1 = The USART drives the SCK pin if USCLKS does not select the external clock SCK.

- **OVER: Oversampling Mode**

0 = 16x Oversampling.

1 = 8x Oversampling.

- **INACK: Inhibit Non Acknowledge**

0 = The NACK is generated.

1 = The NACK is not generated.

- **DSNACK: Disable Successive NACK**

0 = NACK is sent on the ISO line as soon as a parity error occurs in the received character (unless INACK is set).

1 = Successive parity errors are counted up to the value specified in the MAX\_ITERATION field. These parity errors generate a NACK on the ISO line. As soon as this value is reached, no additional NACK is sent on the ISO line. The flag ITERATION is asserted.

- **MAX\_ITERATION**

Defines the maximum number of iterations in mode ISO7816, protocol T = 0.

- **FILTER: Infrared Receive Line Filter**

0 = The USART does not filter the receive line.

1 = The USART filters the receive line using a three-sample filter (1/16-bit clock) (2 over 3 majority).

## USART Interrupt Enable Register

Name: US\_IER

Access Type: Write-only

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

- **RXRDY: RXRDY Interrupt Enable**
- **TXRDY: TXRDY Interrupt Enable**
- **RXBRK: Receiver Break Interrupt Enable**
- **ENDRX: End of Receive Transfer Interrupt Enable**
- **ENDTX: End of Transmit Interrupt Enable**
- **OVRE: Overrun Error Interrupt Enable**
- **FRAME: Framing Error Interrupt Enable**
- **PARE: Parity Error Interrupt Enable**
- **TIMEOUT: Time-out Interrupt Enable**
- **TXEMPTY: TXEMPTY Interrupt Enable**
- **ITERATION: Iteration Interrupt Enable**
- **TXBUFE: Buffer Empty Interrupt Enable**
- **RXBUFF: Buffer Full Interrupt Enable**
- **NACK: Non Acknowledge Interrupt Enable**
- **RIIC: Ring Indicator Input Change Enable**
- **DSRIC: Data Set Ready Input Change Enable**
- **DCDIC: Data Carrier Detect Input Change Interrupt Enable**
- **CTSIC: Clear to Send Input Change Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

**USART Interrupt Disable Register**

**Name:** US\_IDR

**Access Type:** Write-only

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

- **RXRDY: RXRDY Interrupt Disable**
- **TXRDY: TXRDY Interrupt Disable**
- **RXBRK: Receiver Break Interrupt Disable**
- **ENDRX: End of Receive Transfer Interrupt Disable**
- **ENDTX: End of Transmit Interrupt Disable**
- **OVRE: Overrun Error Interrupt Disable**
- **FRAME: Framing Error Interrupt Disable**
- **PARE: Parity Error Interrupt Disable**
- **TIMEOUT: Time-out Interrupt Disable**
- **TXEMPTY: TXEMPTY Interrupt Disable**
- **ITERATION: Iteration Interrupt Disable**
- **TXBUFE: Buffer Empty Interrupt Disable**
- **RXBUFF: Buffer Full Interrupt Disable**
- **NACK: Non Acknowledge Interrupt Disable**
- **RIIC: Ring Indicator Input Change Disable**
- **DSRIC: Data Set Ready Input Change Disable**
- **DCDIC: Data Carrier Detect Input Change Interrupt Disable**
- **CTSIC: Clear to Send Input Change Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

## USART Interrupt Mask Register

Name: US\_IMR

Access Type: Read-only

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

- **RXRDY: RXRDY Interrupt Mask**
- **TXRDY: TXRDY Interrupt Mask**
- **RXBRK: Receiver Break Interrupt Mask**
- **ENDRX: End of Receive Transfer Interrupt Mask**
- **ENDTX: End of Transmit Interrupt Mask**
- **OVRE: Overrun Error Interrupt Mask**
- **FRAME: Framing Error Interrupt Mask**
- **PARE: Parity Error Interrupt Mask**
- **TIMEOUT: Time-out Interrupt Mask**
- **TXEMPTY: TXEMPTY Interrupt Mask**
- **ITERATION: Iteration Interrupt Mask**
- **TXBUFE: Buffer Empty Interrupt Mask**
- **RXBUFF: Buffer Full Interrupt Mask**
- **NACK: Non Acknowledge Interrupt Mask**
- **RIIC: Ring Indicator Input Change Mask**
- **DSRIC: Data Set Ready Input Change Mask**
- **DCDIC: Data Carrier Detect Input Change Interrupt Mask**
- **CTSIC: Clear to Send Input Change Interrupt Mask**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

## USART Channel Status Register

**Name:** US\_CSR

**Access Type:** Read-only

|      |       |      |        |        |           |         |         |
|------|-------|------|--------|--------|-----------|---------|---------|
| 31   | 30    | 29   | 28     | 27     | 26        | 25      | 24      |
| –    | –     | –    | –      | –      | –         | –       | –       |
| 23   | 22    | 21   | 20     | 19     | 18        | 17      | 16      |
| CTS  | DCD   | DSR  | RI     | CTSIC  | DCDIC     | DSRIC   | RIIC    |
| 15   | 14    | 13   | 12     | 11     | 10        | 9       | 8       |
| –    | –     | NACK | RXBUFF | TXBUFE | ITERATION | TXEMPTY | TIMEOUT |
| 7    | 6     | 5    | 4      | 3      | 2         | 1       | 0       |
| PARE | FRAME | OVRE | ENDTX  | ENDRX  | RXBRK     | TXRDY   | RXRDY   |

- **RXRDY: Receiver Ready**

0 = No complete character has been received since the last read of 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 US\_RHR has not yet been read.

- **TXRDY: Transmitter Ready**

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 = There is no character in the US\_THR.

- **RXBRK: Break Received/End of Break**

0 = No Break received or End of Break detected since the last RSTSTA.

1 = Break Received or End of Break detected since the last RSTSTA.

- **ENDRX: End of Receiver Transfer**

0 = The End of Transfer signal from the Receive PDC channel is inactive.

1 = The End of Transfer signal from the Receive PDC channel is active.

- **ENDTX: End of Transmitter Transfer**

0 = The End of Transfer signal from the Transmit PDC channel is inactive.

1 = The End of Transfer signal from the Transmit PDC channel is active.

- **OVRE: Overrun Error**

0 = No overrun error has occurred since since the last RSTSTA.

1 = At least one overrun error has occurred since the last RSTSTA.

- **FRAME: Framing Error**

0 = No stop bit has been detected low since the last RSTSTA.

1 = At least one stop bit has been detected low since the last RSTSTA.

- **PARE: Parity Error**

0 = No parity error has been detected since the last RSTSTA.

1 = At least one parity error has been detected since the last RSTSTA.

- **TIMEOUT: Receiver Time-out**

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**

0 = There are characters in either US\_THR or the Transmit Shift Register, or the transmitter is disabled.

1 = There is at least one character in either US\_THR or the Transmit Shift Register.

- **ITERATION: Max number of Repetitions Reached**

0 = Maximum number of repetitions has not been reached since the last RSIT.

1 = Maximum number of repetitions has been reached since the last RSIT.

- **TXBUFE: Transmission Buffer Empty**

0 = The signal Buffer Empty from the Transmit PDC channel is inactive.

1 = The signal Buffer Empty from the Transmit PDC channel is active.

- **RXBUFF: Reception Buffer Full**

0 = The signal Buffer Full from the Receive PDC channel is inactive.

1 = The signal Buffer Full from the Receive PDC channel is active.

- **NACK: Non Acknowledge**

0 = No Non Acknowledge has not been detected since the last RSTNACK.

1 = At least one Non Acknowledge has been detected since the last RSTNACK.

- **RIIC: Ring Indicator Input Change Flag**

0 = No input change has been detected on the RI pin since the last read of US\_CSR.

1 = At least one input change has been detected on the RI pin since the last read of US\_CSR.

- **DSRIC: Data Set Ready Input Change Flag**

0 = No input change has been detected on the DSR pin since the last read of US\_CSR.

1 = At least one input change has been detected on the DSR pin since the last read of US\_CSR.

- **DCDIC: Data Carrier Detect Input Change Flag**

0 = No input change has been detected on the DCD pin since the last read of US\_CSR.

1 = At least one input change has been detected on the DCD pin since the last read of US\_CSR.

- **CTSIC: Clear to Send Input Change Flag**

0 = No input change has been detected on the CTS pin since the last read of US\_CSR.

1 = At least one input change has been detected on the CTS pin since the last read of US\_CSR.

- **RI: Image of RI Input**

0 = RI is at 0.

1 = RI is at 1.

- **DSR: Image of DSR Input**

0 = DSR is at 0

1 = DSR is at 1.

- **DCD: Image of DCD Input**

0 = DCD is at 0.

1 = DCD is at 1.

- **CTS: Image of CTS Input**

0 = CTS is at 0.

1 = CTS is at 1.



**USART Receive Holding Register**

**Name:** US\_RHR  
**Access Type:** Read-only

|       |    |    |    |    |    |    |       |
|-------|----|----|----|----|----|----|-------|
| 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.

**USART Transmit Holding Register**

**Name:** US\_THR  
**Access Type:** Write-only

|       |    |    |    |    |    |    |       |
|-------|----|----|----|----|----|----|-------|
| 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.



## USART Baud Rate Generator Register

Name: US\_BRGR

Access Type: Read/Write

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 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 Divider**

| CD         | USART_MODE ≠ ISO7816             |                                 |                                | USART_MODE = ISO7816                      |
|------------|----------------------------------|---------------------------------|--------------------------------|---|
|            | SYNC = 0                         |                                 | SYNC = 1                       |   |
|            | OVER = 0                         | OVER = 1                        |                                |   |
| 0          | Baud Rate Clock Disabled         |                                 |                                |   |
| 1 to 65535 | Baud Rate = Selected Clock/16/CD | Baud Rate = Selected Clock/8/CD | Baud Rate = Selected Clock /CD | Baud Rate = Selected Clock/CD/FI_DI_RATIO |

**USART Receiver Time-out Register**

Name: US\_RTOR

Access Type: Read/Write

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

• **TO: Time-out Value**

0: The Receiver Time-out is disabled.

1 - 65535: The Receiver Time-out is enabled and the Time-out delay is TO x Bit Period.

**USART Transmitter Timeguard Register**

Name: US\_TTGR

Access Type: Read/Write

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 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: Timeguard Value**

0: The Transmitter Timeguard is disabled.

1 - 255: The Transmitter timeguard is enabled and the timeguard delay is TG x Bit Period.

## USART FI DI RATIO Register

**Name:** US\_FIDI  
**Access Type:** Read/Write  
**Reset Value:** 0x174

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

- **FI\_DI\_RATIO: FI Over DI Ratio Value**

0: If ISO7816 mode is selected, the Baud Rate Generator generates no signal.

1-2047: If ISO7816 mode is selected, the Baud Rate is the clock provided on SCK divided by FI\_DI\_RATIO.

**USART Number of Errors Register**

**Name:** US\_NER

**Access Type:** Read-only

|           |    |    |    |    |    |    |    |
|-----------|----|----|----|----|----|----|----|
| 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  |
| NB_ERRORS |    |    |    |    |    |    |    |

- **NB\_ERRORS: Number of Errors**

Total number of errors that occurred during an ISO7816 transfer. This register automatically clears when read.

## USART IrDA FILTER Register

Name: US\_IF

Access Type: Read/Write

|             |    |    |    |    |    |    |    |
|-------------|----|----|----|----|----|----|----|
| 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  |
| IRDA_FILTER |    |    |    |    |    |    |    |

- **IRDA\_FILTER: IrDA Filter**

Sets the filter of the IrDA demodulator.

## Serial Synchronous Controller (SSC)

### Overview

The Atmel Synchronous Serial Controller (SSC) provides a synchronous communication link with external devices. It supports many serial synchronous communication protocols generally used in audio and telecom applications such as I2S, Short Frame Sync, Long Frame Sync, etc.

The SSC contains an independent receiver and transmitter and a common clock divider. The receiver and the transmitter each interface with three signals: the TD/RD signal for data, the TK/RK signal for the clock and the TF/RF signal for the Frame Sync. Transfers contain up to 16 data of up to 32 bits. they can be programmed to start automatically or on different events detected on the Frame Sync signal.

The SSC's high-level of programmability and its two dedicated PDC channels of up to 32 bits permit a continuous high bit rate data transfer without processor intervention.

Featuring connection to two PDC channels, the SSC permits interfacing with low processor overhead to the following:

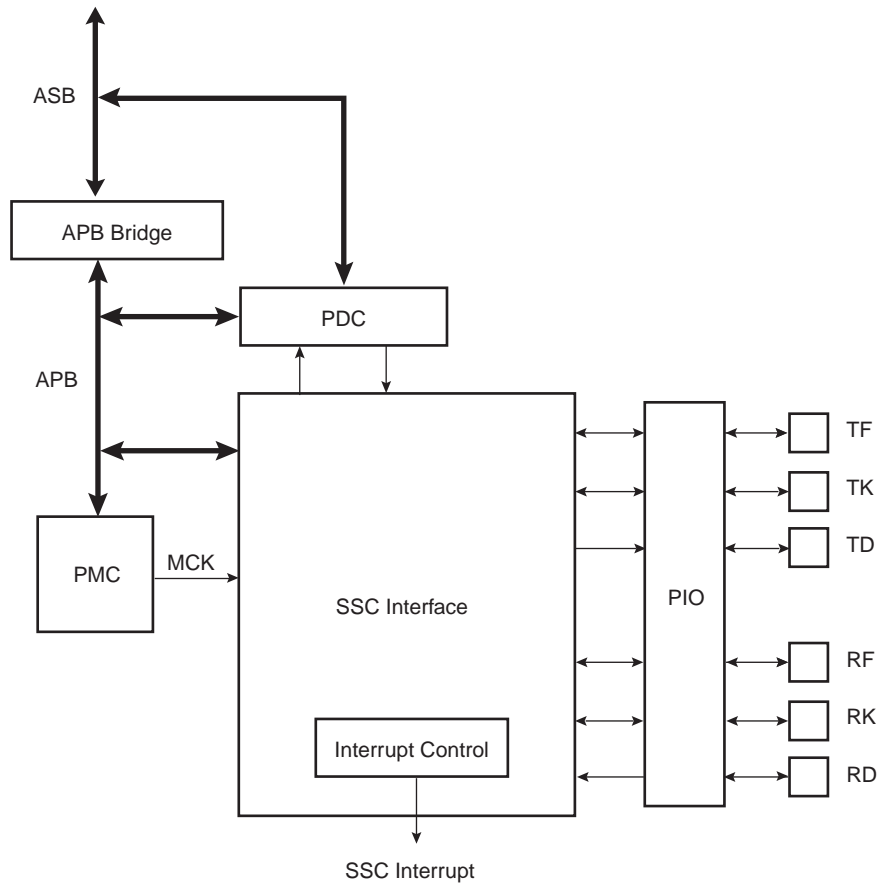
- CODECs in master or slave mode
- DAC through dedicated serial interface, particularly I2S
- Magnetic card reader

Features of the SSC are:

- Provides Serial Synchronous Communication Links Used in Audio and Telecom Applications
- Contains an Independent Receiver and Transmitter and a Common Clock Divider
- Interfaced with Two PDC Channels (DMA Access) to Reduce Processor Overhead
- Offers a Configurable Frame Sync and Data Length
- Receiver and Transmitter can be Programmed to Start Automatically or on Detection of Different Event on the Frame Sync Signal
- Receiver and Transmitter Include a Data Signal, a Clock Signal and a Frame Synchronization Signal

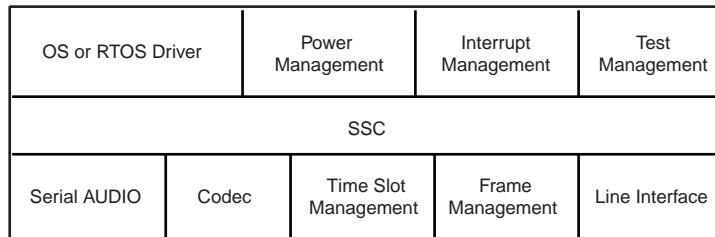
## Block Diagram

Figure 202. Block Diagram



## Application Block Diagram

Figure 203. Application Block Diagram





## Pin Name List

**Table 85.** I/O Lines Description

| Pin Name | Pin Description           | Type         |
|----------|---------------------------|--------------|
| RF       | Receiver Frame Synchro    | Input/Output |
| RK       | Receiver Clock            | Input/Output |
| RD       | Receiver Data             | Input        |
| TF       | Transmitter Frame Synchro | Input/Output |
| TK       | Transmitter Clock         | Input/Output |
| TD       | Transmitter Data          | Output       |

## Product Dependencies

### I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines.

Before using the SSC receiver, the PIO controller must be configured to dedicate the SSC receiver I/O lines to the SSC peripheral mode.

Before using the SSC transmitter, the PIO controller must be configured to dedicate the SSC transmitter I/O lines to the SSC peripheral mode.

### Power Management

The SSC is not continuously clocked. The SSC interface may be clocked through the Power Management Controller (PMC), therefore the programmer must first configure the PMC to enable the SSC clock.

### Interrupt

The SSC interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling interrupts requires programming the AIC before configuring the SSC.

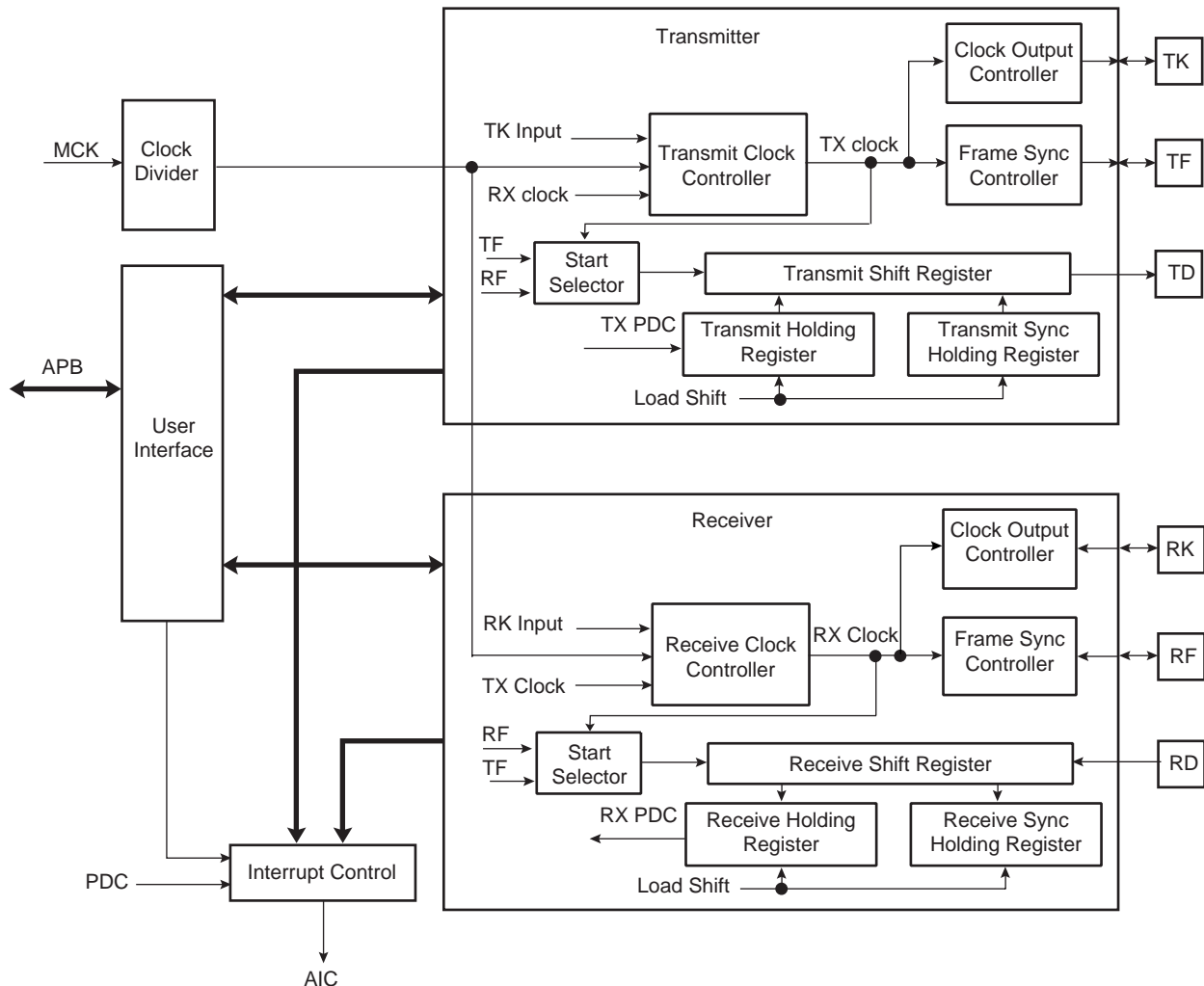
All SSC interrupts can be enabled/disabled configuring the SSC Interrupt mask register. Each pending and unmasked SSC interrupt will assert the SSC interrupt line. The SSC interrupt service routine can get the interrupt origin by reading the SSC interrupt status register.

## Functional Description

This chapter contains the functional description of the following: SSC Functional Block, Clock Management, Data format, Start, Transmitter, Receiver and Frame Sync.

The receiver and transmitter operate separately. However, they can work synchronously by programming the receiver to use the transmit clock and/or to start a data transfer when transmission starts. Alternatively, this can be done by programming the transmitter to use the receive clock and/or to start a data transfer when reception starts. The transmitter and the receiver can be programmed to operate with the clock signals provided on either the TK or RK pins. This allows the SSC to support many slave-mode data transfers. The maximum clock speed allowed on the TK and RK pins is the master clock divided by 2. Each level of the clock must be stable for at least two master clock periods.

**Figure 204.** SSC Functional Block Diagram



## Clock Management

The transmitter clock can be generated by:

- an external clock received on the TK I/O pad
- the receiver clock
- the internal clock divider

The receiver clock can be generated by:

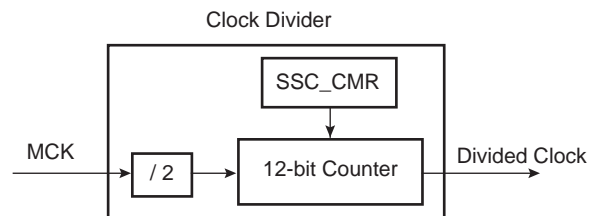
- an external clock received on the RK I/O pad
- the transmitter clock
- the internal clock divider

Furthermore, the transmitter block can generate an external clock on the TK I/O pad, and the receiver block can generate an external clock on the RK I/O pad.

This allows the SSC to support many Master and Slave-mode data transfers.

## Clock Divider

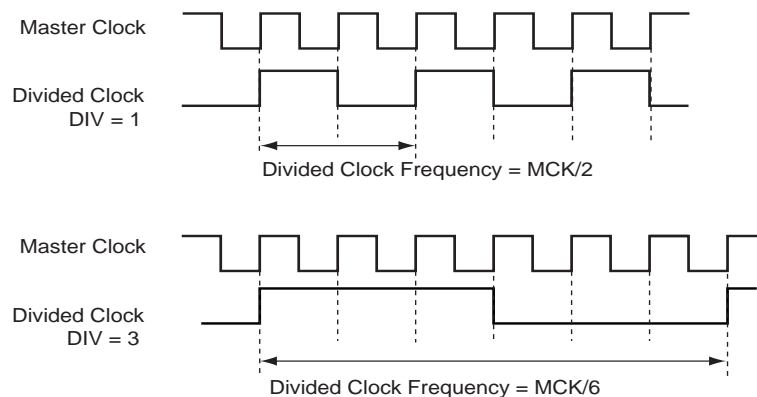
**Figure 205.** Divided Clock Block Diagram



The Master Clock divider is determined by the 12-bit field DIV counter and comparator (so its maximal value is 4095) in the Clock Mode Register SSC\_CMCR, allowing a Master Clock division by up to 8190. The Divided Clock is provided to both the Receiver and Transmitter. When this field is programmed to 0, the Clock Divider is not used and remains inactive.

When DIV is set to a value equal or greater to 1, the Divided Clock has a frequency of Master Clock divided by 2 times DIV. Each level of the Divided Clock has a duration of the Master Clock multiplied by DIV. This ensures a 50% duty cycle for the Divided Clock regardless if the DIV value is even or odd.

**Figure 206.** Divided Clock Generation



**Table 86.** Bit Rate

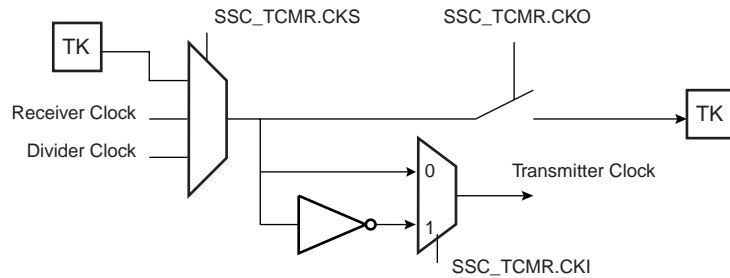
| Maximum | Minimum    |
|---------|------------|
| MCK / 2 | MCK / 8190 |

## Transmitter Clock Management

The transmitter clock is generated from the receiver clock or the divider clock or an external clock scanned on the TK I/O pad. The transmitter clock is selected by the CKS field in SSC\_TCMR (Transmit Clock Mode Register). Transmit Clock can be inverted independently by the CKI bits in SSC\_TCMR.

The transmitter can also drive the TK I/O pad continuously or be limited to the actual data transfer. The clock output is configured by the SSC\_TCMR register. The Transmit Clock Inversion (CKI) bits have no effect on the clock outputs. Programming the TCMR register to select TK pin (CKS field) and at the same time Continuous Transmit Clock (CKO field) might lead to unpredictable results.

**Figure 207.** Transmitter Clock Management

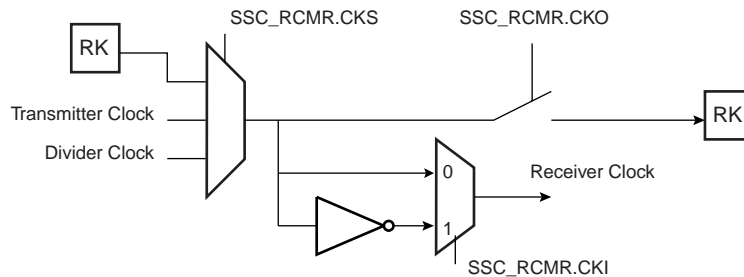


## Receiver Clock Management

The receiver clock is generated from the transmitter clock or the divider clock or an external clock scanned on the RK I/O pad. The Receive Clock is selected by the CKS field in SSC\_RCMR (Receive Clock Mode Register). Receive Clocks can be inverted independently by the CKI bits in SSC\_RCMR.

The receiver can also drive the RK I/O pad continuously or be limited to the actual data transfer. The clock output is configured by the SSC\_RCMR register. The Receive Clock Inversion (CKI) bits have no effect on the clock outputs. Programming the RCMR register to select RK pin (CKS field) and at the same time Continuous Receive Clock (CKO field) might lead to unpredictable results.

**Figure 208.** Receiver Clock Management



### Transmitter Operations

A transmitted frame is triggered by a start event and can be followed by synchronization data before data transmission.

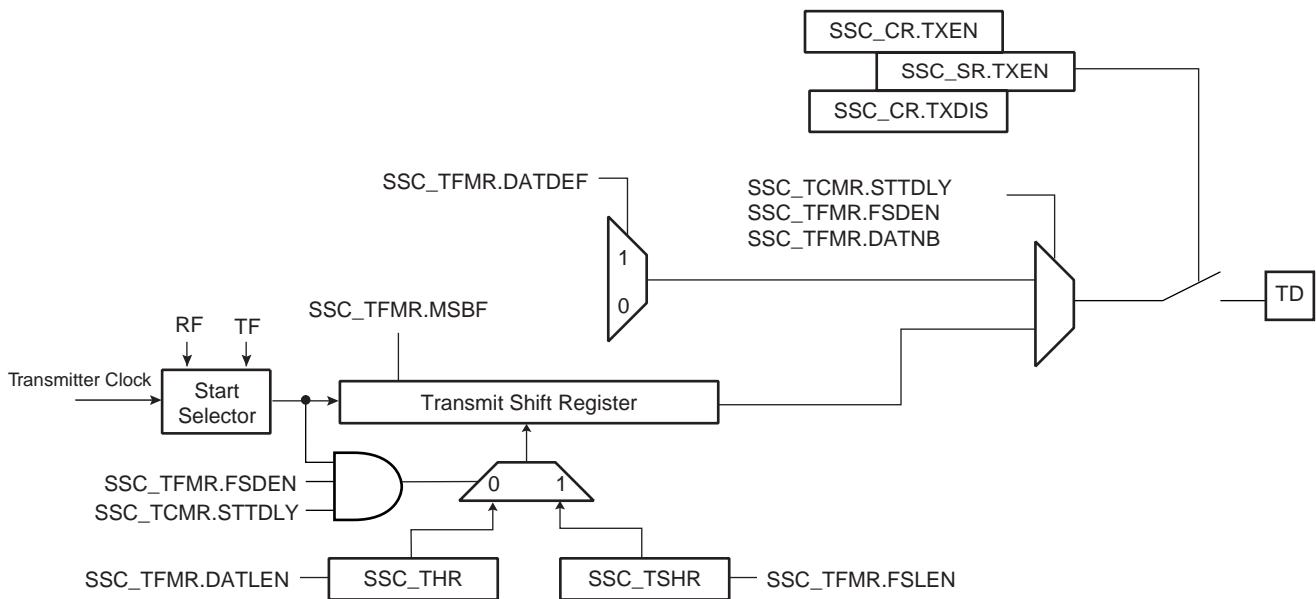
The start event is configured by setting the Transmit Clock Mode Register (SSC\_TCMR). See “Start” on page 448.

The frame synchronization is configured setting the Transmit Frame Mode Register (SSC\_TFMR). See “Frame Sync” on page 450.

To transmit data, the transmitter uses a shift register clocked by the transmitter clock signal and the start mode selected in the SSC\_TCMR. Data is written by the application to the SSC\_THR register then transferred to the shift register according to the data format selected.

When both the SSC\_THR and the transmit shift register are empty, the status flag TXEMPTY is set in SSC\_SR. When the Transmit Holding register is transferred in the Transmit shift register, the status flag TXRDY is set in SSC\_SR and additional data can be loaded in the holding register.

Figure 209. Transmitter Block Diagram



## Receiver Operations

A received frame is triggered by a start event and can be followed by synchronization data before data transmission.

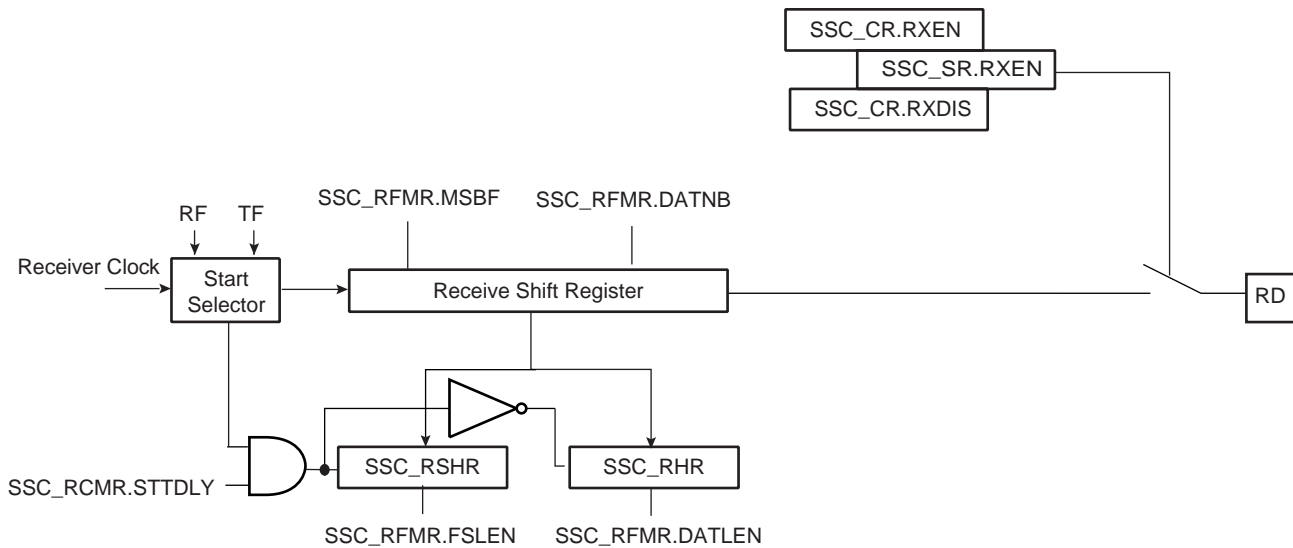
The start event is configured setting the Receive Clock Mode Register (SSC\_RCMR). See “Start” on page 448.

The frame synchronization is configured setting the Receive Frame Mode Register (SSC\_RFMR). See “Frame Sync” on page 450.

The receiver uses a shift register clocked by the receiver clock signal and the start mode selected in the SSC\_RCMR. The data is transferred from the shift register in function of data format selected.

When the receiver shift register is full, the SSC transfers this data in the holding register, the status flag RXRDY is set in SSC\_SR and the data can be read in the receiver holding register, if another transfer occurs before read the RHR register, the status flag OVERUN is set in SSC\_SR and the receiver shift register is transferred in the RHR register.

**Figure 210.** Receiver Block Diagram



## Start

The transmitter and receiver can both be programmed to start their operations when an event occurs, respectively in the Transmit Start Selection (START) field of SSC\_TCMR and in the Receive Start Selection (START) field of SSC\_RCMR.

Under the following conditions the start event is independently programmable:

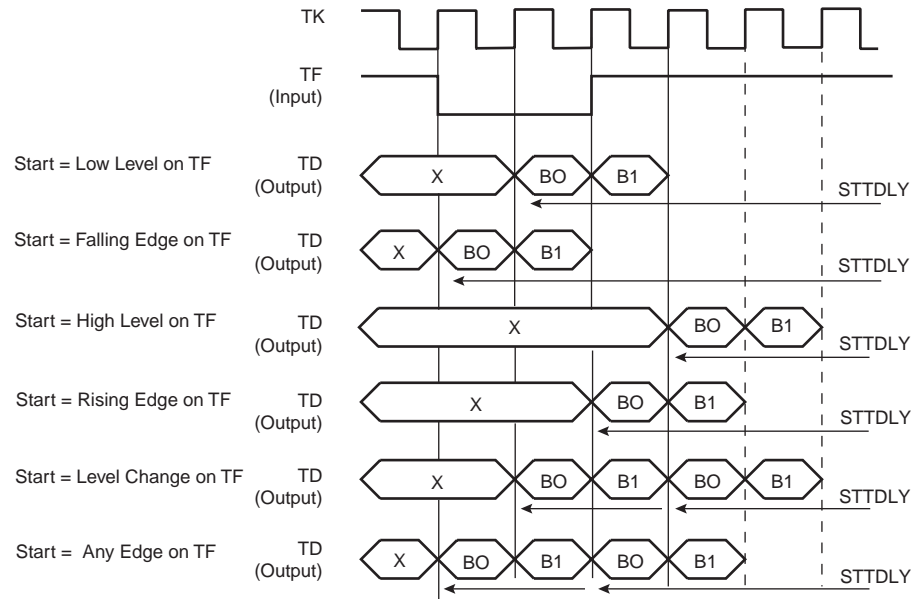
- Continuous. In this case, the transmission starts as soon as a word is written in SSC\_THR and the reception starts as soon as the Receiver is enabled.
- Synchronously with the transmitter/receiver
- On detection of a falling/rising edge on TK/RK
- On detection of a low level/high level on TK/RK
- On detection of a level change or an edge on TK/RK

A start can be programmed in the same manner on either side of the Transmit/Receive Clock Register (RCMR/TCMR). Thus, the start could be on TF (Transmit) or RF (Receive).

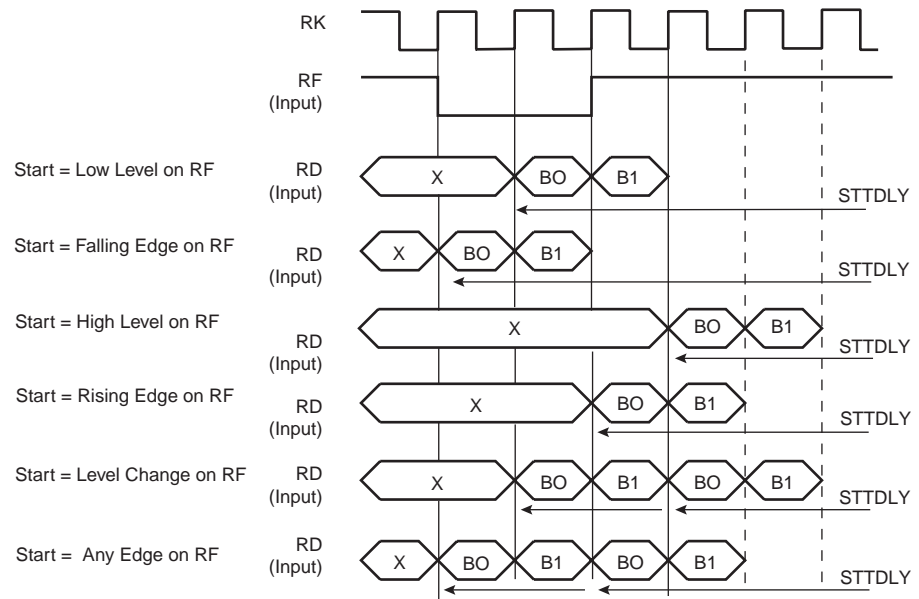
Detection on TF/RF input/output is done through the field FSOS of the Transmit / Receive Frame Mode Register (TFMR/RFMR).

Generating a Frame Sync signal is not possible without generating it on its related output.

**Figure 211. Transmit Start Mode**



**Figure 212. Receive Pulse/Edge Start Modes**



## Frame Sync

The Transmitter and Receiver Frame Sync pins, TF and RF, can be programmed to generate different kinds of frame synchronization signals. The Frame Sync Output Selection (FSOS) field in the Receive Frame Mode Register (SSC\_RFMR) and in the Transmit Frame Mode Register (SSC\_TFMR) are used to select the required waveform.

- Programmable low or high levels during data transfer are supported.
- Programmable high levels before the start of data transfers or toggling are also supported.

If a pulse waveform is selected, the Frame Sync Length (FSLEN) field in SSC\_RFMR and SSC\_TFMR programs the length of the pulse, from 1-bit time up to 16-bit time.

The periodicity of the Receive and Transmit Frame Sync pulse output can be programmed through the Period Divider Selection (PERIOD) field in SSC\_RCMR and SSC\_TCMR.

## Frame Sync Data

Frame Sync Data transmits or receives a specific tag during the Frame Synchro signal.

During the Frame Sync signal, the Receiver can sample the RD line and store the data in the Receive Sync Holding Register and the transmitter can transfer Transmit Sync Holding Register in the Shifter Register. The data length to be sampled/shifted out during the Frame Sync signal is programmed by the FSLEN field in SSC\_RFMR/SSC\_TFMR.

Concerning the Receive Frame Sync Data operation, if the Frame Sync Length is equal to or lower than the delay between the start event and the actual data reception, the data sampling operation is performed in the Receive Sync Holding Register through the Receive Shift Register.

The Transmit Frame Sync Operation is performed by the transmitter only if the bit Frame Sync Data Enable (FSDEN) in SSC\_TFMR is set. If the Frame Sync length is equal to or lower than the delay between the start event and the actual data transmission, the normal transmission has priority and the data contained in the Transmit Sync Holding Register is transferred in the Transmit Register then shifted out.

## Frame Sync Edge Detection

The Frame Sync Edge detection is programmed by the FSEDGE field in SSC\_RFMR/SSC\_TFMR. This sets the corresponding flags RXSYN/TXSYN in the SSC Status Register (SSC\_SR) on frame synchro edge detection (signals RF/TF).

## Data Format

The data framing format of both the transmitter and the receiver are largely programmable through the Transmitter Frame Mode Register (SSC\_TFMR) and the Receiver Frame Mode Register (SSC\_RFMR). In either case, the user can independently select:

- The event that starts the data transfer (START).
- The delay in number of bit periods between the start event and the first data bit (STTDLY).
- The length of the data (DATLEN)
- The number of data to be transferred for each start event (DATNB).
- The length of Synchronization transferred for each start event (FSLEN).
- The bit sense: most or lowest significant bit first (MSBF).

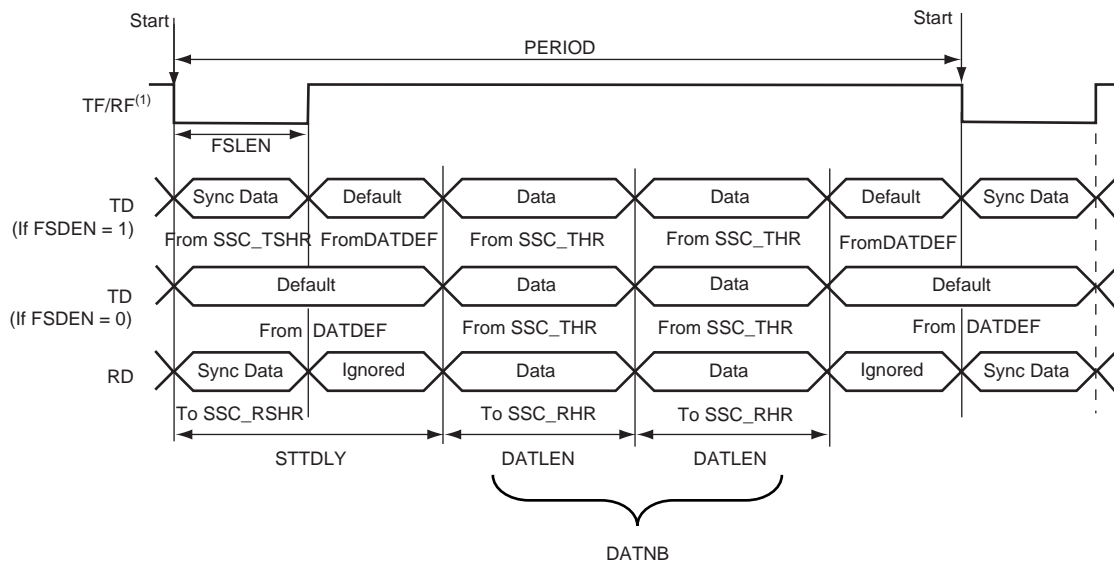
Additionally, the transmitter can be used to transfer Synchronization and select the level driven on the TD pin while not in data transfer operation. This is done respectively by the Frame Sync Data Enable (FSDEN) and by the Data Default Value (DATDEF) bits in SSC\_TFMR.



**Table 87.** Data Frame Registers

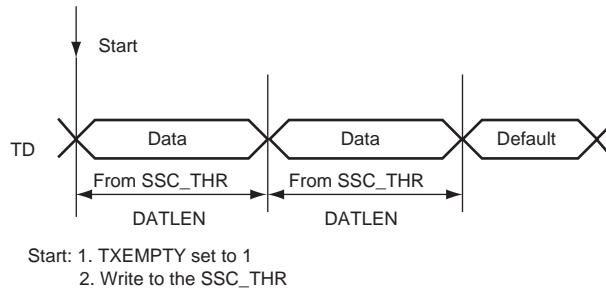
| Transmitter | Receiver | Field  | Length    | Comment                          |
|-------------|----------|--------|-----------|----------------------------------|
| SSC_TFMR    | SSC_RFMR | DATLEN | Up to 32  | Size of word                     |
| SSC_TFMR    | SSC_RFMR | DATNB  | Up to 16  | Number Word transmitter in frame |
| SSC_TFMR    | SSC_RFMR | MSBF   |           | 1 most significant bit in first  |
| SSC_TFMR    | SSC_RFMR | FSLEN  | Up to 16  | Size of Synchro data register    |
| SSC_TFMR    |          | DATDEF | 0 or 1    | Data default value ended         |
| SSC_TFMR    |          | FSDEN  |           | Enable send SSC_TSHR             |
| SSC_TCMR    | SSC_RCMR | PERIOD | up to 512 | Frame size                       |
| SSC_TCMR    | SSC_RCMR | STTDLY | up to 255 | Size of transmit start delay     |

**Figure 213.** Transmit and Receive Frame Format in Edge/Pulse Start Modes



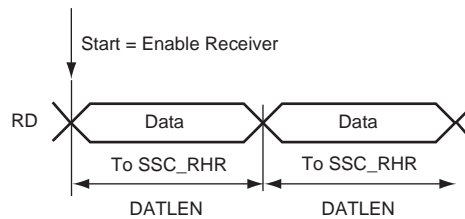
Note: 1. Input on falling edge on TF/RF example.

**Figure 214.** Transmit Frame Format in Continuous Mode



Note: 1. STTDLY is set to 0. In this example, SSC\_THR is loaded twice. The value of FSDEN has no effect on transmission. SyncData cannot be output in continuous mode.

**Figure 215.** Receive Frame Format in Continuous Mode



Note: 1. STTDLY is set to 0.

## Loop Mode

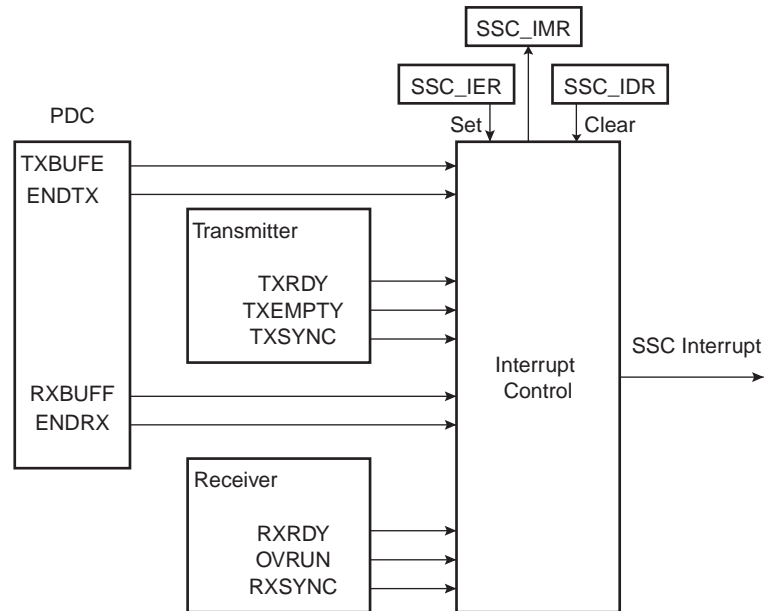
The receiver can be programmed to receive transmissions from the transmitter. This is done by setting the Loop Mode (LOOP) bit in SSC\_RFMR. In this case, RD is connected to TD, RF is connected to TF and RK is connected to TK.

## Interrupt

Most bits in SSC\_SR have a corresponding bit in interrupt management registers.

The SSC Controller can be programmed to generate an interrupt when it detects an event. The Interrupt is controlled by writing SSC\_IER (Interrupt Enable Register) and SSC\_IDR (Interrupt Disable Register), which respectively enable and disable the corresponding interrupt by setting and clearing the corresponding bit in SSC\_IMR (Interrupt Mask Register), which controls the generation of interrupts by asserting the SSC interrupt line connected to the AIC.

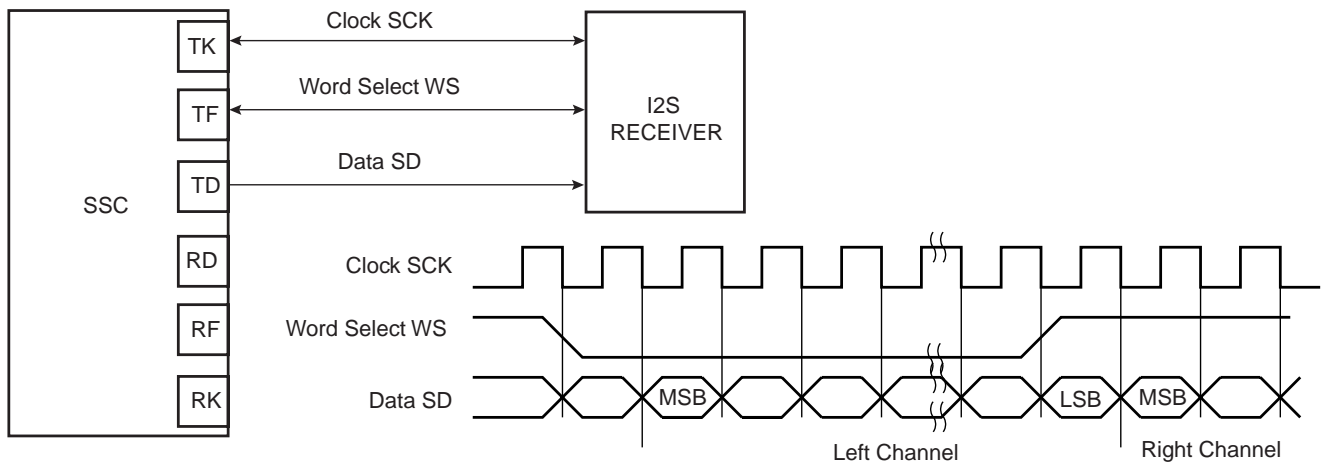
Figure 216. Interrupt Block Diagram



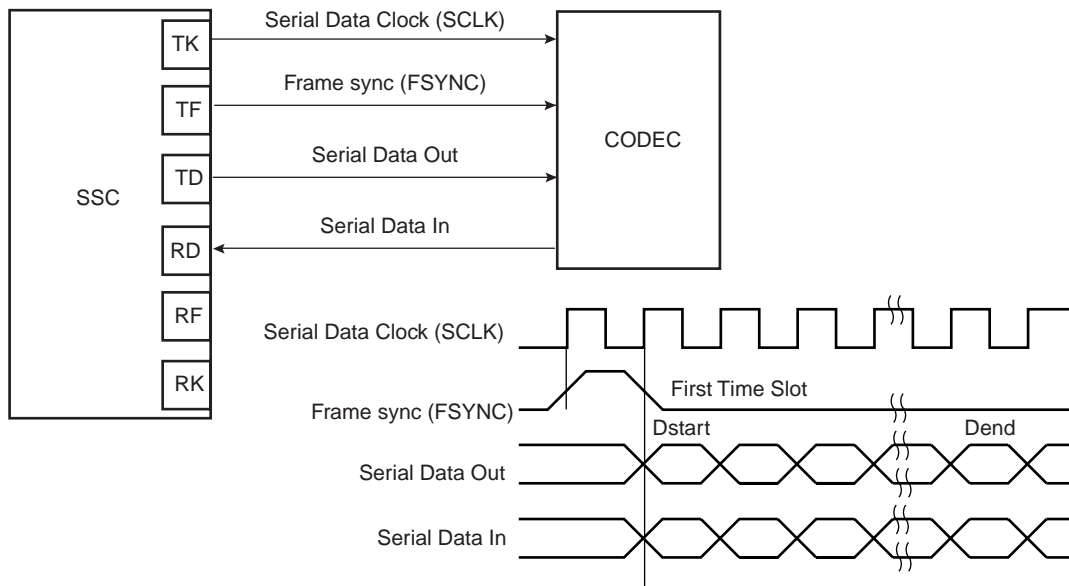
## SSC Application Examples

The SSC can support several serial communication modes used in audio or high speed serial links. Some standard applications are shown in the following figures. All serial link applications supported by the SSC are not listed here.

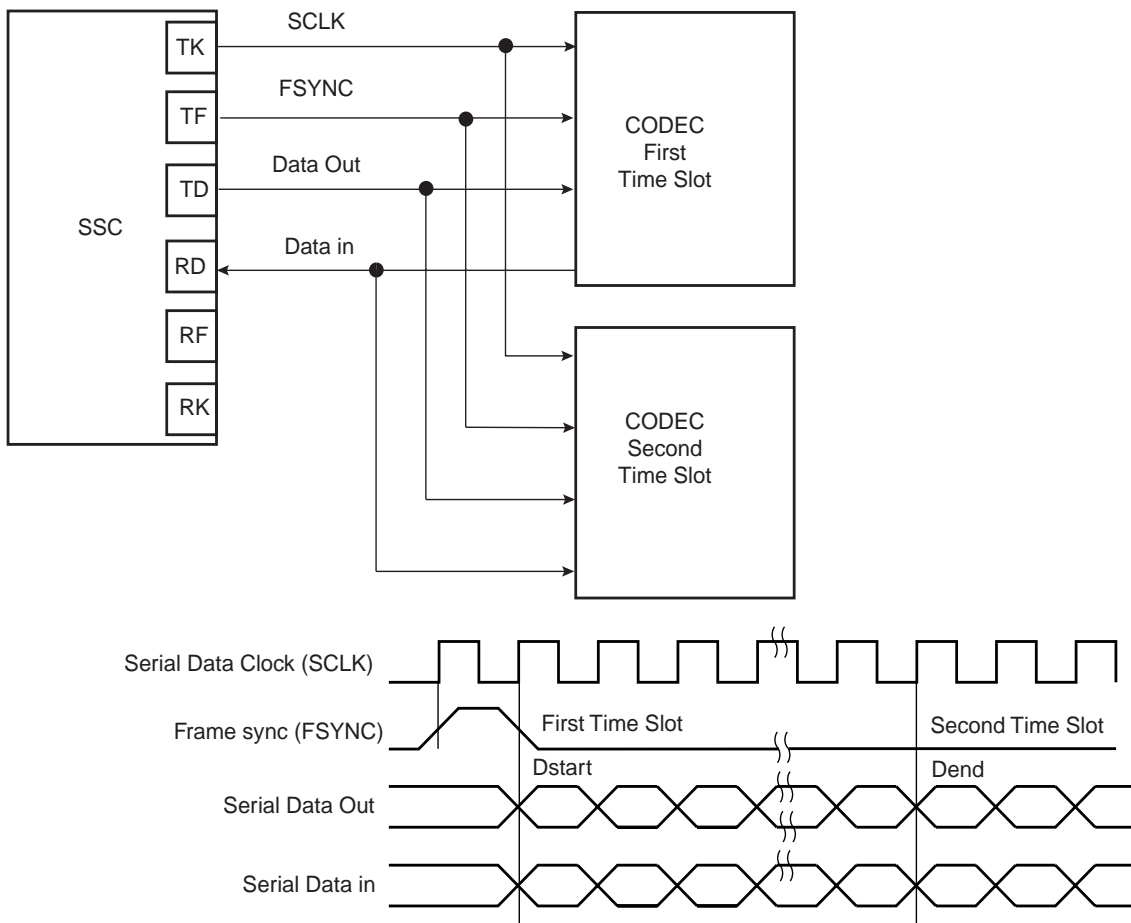
Figure 217. Audio Application Block Diagram



**Figure 218.** Codec Application Block Diagram



**Figure 219.** Time Slot Application Block Diagram



## Serial Synchronous Controller (SSC) User Interface

**Table 88.** SSC Register Mapping

| Offset       | Register                                      | Register Name | Access     | Reset      |
|--------------|---|---------------|------------|------------|
| 0x0          | Control Register                              | SSC_CR        | Write      | –          |
| 0x4          | Clock Mode Register                           | SSC_CMR       | Read/Write | 0x0        |
| 0x8          | Reserved                                      | –             | –          | –          |
| 0xC          | Reserved                                      | –             | –          | –          |
| 0x10         | Receive Clock Mode Register                   | SSC_RCMR      | Read/Write | 0x0        |
| 0x14         | Receive Frame Mode Register                   | SSC_RFMR      | Read/Write | 0x0        |
| 0x18         | Transmit Clock Mode Register                  | SSC_TCMR      | Read/Write | 0x0        |
| 0x1C         | Transmit Frame Mode Register                  | SSC_TFMR      | Read/Write | 0x0        |
| 0x20         | Receive Holding Register                      | SSC_RHR       | Read       | 0x0        |
| 0x24         | Transmit Holding Register                     | SSC_THR       | Write      | –          |
| 0x28         | Reserved                                      | –             | –          | –          |
| 0x2C         | Reserved                                      | –             | –          | –          |
| 0x30         | Receive Sync. Holding Register                | SSC_RSHR      | Read       | 0x0        |
| 0x34         | Transmit Sync. Holding Register               | SSC_TSHR      | Read/Write | 0x0        |
| 0x38         | Reserved                                      | –             | –          | –          |
| 0x3C         | Reserved                                      | –             | –          | –          |
| 0x40         | Status Register                               | SSC_SR        | Read       | 0x000000CC |
| 0x44         | Interrupt Enable Register                     | SSC_IER       | Write      | –          |
| 0x48         | Interrupt Disable Register                    | SSC_IDR       | Write      | –          |
| 0x4C         | Interrupt Mask Register                       | SSC_IMR       | Read       | 0x0        |
| 0x50-0xFF    | Reserved                                      | –             | –          | –          |
| 0x100- 0x124 | Reserved for Peripheral Data Controller (PDC) | –             | –          | –          |



## SSC Control Register

Name: SSC\_CR

Access Type: Write-only

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

- **RXEN: Receive Enable**

0: No effect.

1: Enables Data Receive if RXDIS is not set<sup>(1)</sup>.

- **RXDIS: Receive Disable**

0: No effect.

1: Disables Data Receive<sup>(1)</sup>.

- **TXEN: Transmit Enable**

0: No effect.

1: Enables Data Transmit if TXDIS is not set<sup>(1)</sup>.

- **TXDIS: Transmit Disable**

0: No effect.

1: Disables Data Transmit<sup>(1)</sup>.

- **SWRST: Software Reset**

0: No effect.

1: Performs a software reset. Has priority on any other bit in SSC\_CR.

Note: 1. Only the data management is affected

**SSC Clock Mode Register**

**Name:** SSC\_CMCR

**Access Type:** Read/Write

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

• **DIV: Clock Divider**

0: The Clock Divider is not active.

Any Other Value: The Divided Clock equals the Master Clock divided by 2 times DIV. The maximum bit rate is MCK/2. The minimum bit rate is  $MCK/2 \times 4095 = MCK/8190$ .

## SSC Receive Clock Mode Register

Name: SSC\_RCMR

Access Type: Read/Write

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

### • CKS: Receive Clock Selection

| CKS | Selected Receive Clock |
|-----|------------------------|
| 0x0 | Divided Clock          |
| 0x1 | TK Clock Signal        |
| 0x2 | RK Pin                 |
| 0x3 | Reserved               |

### • CKO: Receive Clock Output Mode Selection

| CKO     | Receive Clock Output Mode | RK pin     |
|---------|---------------------------|------------|
| 0x0     | None                      | Input-only |
| 0x1     | Continuous Receive Clock  | Output     |
| 0x2-0x7 | Reserved                  |            |

### • CKI: Receive Clock Inversion

0: The data and the Frame Sync signal are sampled on Receive Clock falling edge.

1: The data and the Frame Sync signal are shifted out on Receive Clock rising edge.

CKI does not affect the RK output clock signal.

### • START: Receive Start Selection

| START   | Receive Start   |
|---------|---|
| 0x0     | Continuous, as soon as the receiver is enabled, and immediately after the end of transfer of the previous data. |
| 0x1     | Transmit Start  |
| 0x2     | Detection of a low level on RF input  |
| 0x3     | Detection of a high level on RF input   |
| 0x4     | Detection of a falling edge on RF input   |
| 0x5     | Detection of a rising edge on RF input  |
| 0x6     | Detection of any level change on RF input   |
| 0x7     | Detection of any edge on RF input   |
| 0x8-0xF | Reserved  |



- **STTDLY: Receive Start Delay**

If STTDLY is not 0, a delay of STTDLY clock cycles is inserted between the start event and the actual start of reception. When the Receiver is programmed to start synchronously with the Transmitter, the delay is also applied.

Please Note: It is very important that STTDLY be set carefully. If STTDLY must be set, it should be done in relation to TAG (Receive Sync Data) reception.

- **PERIOD: Receive Period Divider Selection**

This field selects the divider to apply to the selected Receive Clock in order to generate a new Frame Sync Signal. If 0, no PERIOD signal is generated. If not 0, a PERIOD signal is generated each  $2 \times (\text{PERIOD} + 1)$  Receive Clock.

## SSC Receive Frame Mode Register

**Name:** SSC\_RFMR

**Access Type:** Read/Write

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

- **DATLEN: Data Length**

0x0 is not supported. The value of DATLEN can be set between 0x1 and 0x1F.

The bit stream contains DATLEN + 1 data bits. Moreover, it defines the transfer size performed by the PDC assigned to the Receiver.

If DATLEN is less than or equal to 7, data transfers are in bytes. If DATLEN is between 8 and 15 (included), half-words are transferred. For any other value, 32-bit words are transferred.

- **LOOP: Loop Mode**

0: Normal operating mode.

1: RD is driven by TD, RF is driven by TF and TK drives RK.

- **MSBF: Most Significant Bit First**

0: The lowest significant bit of the data register is sampled first in the bit stream.

1: The most significant bit of the data register is sampled first in the bit stream.

- **DATNB: Data Number per Frame**

This field defines the number of data words to be received after each transfer start. If 0, only 1 data word is transferred. Up to 16 data words can be transferred.

- **FSLEN: Receive Frame Sync Length**

This field defines the length of the Receive Frame Sync Signal and the number of bits sampled and stored in the Receive Sync Data Register. Only when FSOS is set on negative or positive pulse.

- **FSOS: Receive Frame Sync Output Selection**

| FSOS    | Selected Receive Frame Sync Signal      | RF pin     |
|---------|---|------------|
| 0x0     | None                                    | Input-only |
| 0x1     | Negative Pulse                          | Output     |
| 0x2     | Positive Pulse                          | Output     |
| 0x3     | Driven Low during data transfer         | Output     |
| 0x4     | Driven High during data transfer        | Output     |
| 0x5     | Toggling at each start of data transfer | Output     |
| 0x6-0x7 | Reserved                                | Undefined  |

- **FSEEDGE: Frame Sync Edge Detection**

Determines which edge on Frame Sync sets RXSYN in the SSC Status Register.

| <b>FSEEDGE</b> | <b>Frame Sync Edge Detection</b> |
|----------------|----------------------------------|
| 0x0            | Positive Edge Detection          |
| 0x1            | Negative Edge Detection          |

## SSC Transmit Clock Mode Register

Name: SSC\_TCMR

Access Type: Read/Write

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

### • CKS: Transmit Clock Selection

| CKS | Selected Transmit Clock |
|-----|-------------------------|
| 0x0 | Divided Clock           |
| 0x1 | RK Clock signal         |
| 0x2 | TK Pin                  |
| 0x3 | Reserved                |

### • CKO: Transmit Clock Output Mode Selection

| CKO     | Transmit Clock Output Mode | TK pin     |
|---------|----------------------------|------------|
| 0x0     | None                       | Input-only |
| 0x1     | Continuous Transmit Clock  | Output     |
| 0x2-0x7 | Reserved                   |            |

### • CKI: Transmit Clock Inversion

0: The data and the Frame Sync signal are shifted out on Transmit Clock falling edge.

1: The data and the Frame Sync signal are shifted out on Transmit Clock rising edge.

CKI affects only the Transmit Clock and not the output clock signal.

### • START: Transmit Start Selection

| START   | Transmit Start  |
|---------|---|
| 0x0     | Continuous, as soon as a word is written in the SSC_THR Register (if Transmit is enabled) and immediately after the end of transfer of the previous data. |
| 0x1     | Receive Start   |
| 0x2     | Detection of a low level on TF signal   |
| 0x3     | Detection of a high level on TF signal  |
| 0x4     | Detection of a falling edge on TF signal  |
| 0x5     | Detection of a rising edge on TF signal   |
| 0x6     | Detection of any level change on TF signal  |
| 0x7     | Detection of any edge on TF signal  |
| 0x8-0xF | Reserved  |

- **STTDLY: Transmit Start Delay**

If STTDLY is not 0, a delay of STTDLY clock cycles is inserted between the start event and the actual start of transmission of data. When the Transmitter is programmed to start synchronously with the Receiver, the delay is also applied.

Please Note: STTDLY must be set carefully. If STTDLY is too short in respect to TAG (Transmit Sync Data) emission, data is emitted instead of the end of TAG.

- **PERIOD: Transmit Period Divider Selection**

This field selects the divider to apply to the selected Transmit Clock to generate a new Frame Sync Signal. If 0, no period signal is generated. If not 0, a period signal is generated at each  $2 \times (\text{PERIOD} + 1)$  Transmit Clock.

## SSC Transmit Frame Mode Register

**Name:** SSC\_TFMR

**Access Type:** Read/Write

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

- **DATLEN: Data Length**

0x0 is not supported. The value of DATLEN can be set between 0x1 and 0x1F.

The bit stream contains DATLEN + 1 data bits. Moreover, it defines the transfer size performed by the PDC assigned to the Receiver.

If DATLEN is less than or equal to 7, data transfers are in bytes. If DATLEN is between 8 and 15 (included), half-words are transferred. For any other value, 32-bit words are transferred.

- **DATDEF: Data Default Value**

This bit defines the level driven on the TD pin while out of transmission. Note that if the pin is defined as multi-drive by the PIO Controller, the pin is enabled only if the SCC TD output is 1.

- **MSBF: Most Significant Bit First**

0: The lowest significant bit of the data register is shifted out first in the bit stream.

1: The most significant bit of the data register is shifted out first in the bit stream.

- **DATNB: Data Number per frame**

This field defines the number of data words to be transferred after each transfer start. If 0, only 1 data word is transferred and up to 16 data words can be transferred.

- **FSLEN: Transmit Frame Sync Length**

This field defines the length of the Transmit Frame Sync signal and the number of bits shifted out from the Transmit Sync Data Register if FSDEN is 1. If 0, the Transmit Frame Sync signal is generated during one Transmit Clock period and up to 16 clock period pulse length is possible.

- **FSOS: Transmit Frame Sync Output Selection**

| FSOS    | Selected Transmit Frame Sync Signal     | TF pin     |
|---------|---|------------|
| 0x0     | None                                    | Input-only |
| 0x1     | Negative Pulse                          | Output     |
| 0x2     | Positive Pulse                          | Output     |
| 0x3     | Driven Low during data transfer         | Output     |
| 0x4     | Driven High during data transfer        | Output     |
| 0x5     | Toggling at each start of data transfer | Output     |
| 0x6-0x7 | Reserved                                | Undefined  |

- **FSDEN: Frame Sync Data Enable**

0: The TD line is driven with the default value during the Transmit Frame Sync signal.

1: SSC\_TSHR value is shifted out during the transmission of the Transmit Frame Sync signal.

- **FSEDGE: Frame Sync Edge Detection**

Determines which edge on frame sync sets TXSYN (Status Register).

| <b>FSEDGE</b> | <b>Frame Sync Edge Detection</b> |
|---------------|----------------------------------|
| 0x0           | Positive Edge Detection          |
| 0x1           | Negative Edge Detection          |



## SSC Receive Holding Register

Name: SSC\_RHR

Access Type: Read-only

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

- **RDAT: Receive Data**

Right aligned regardless of the number of data bits defined by DATLEN in SSC\_RFMR.

## SSC Transmit Holding Register

Name: SSC\_THR

Access Type: Write only

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

- **TDAT: Transmit Data**

Right aligned regardless of the number of data bits defined by DATLEN in SSC\_TFMR.



### SSC Receive Synchronization Holding Register

Name: SSC\_RSHR

Access Type: Read/Write

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

- RSDAT: Receive Synchronization Data**

Right aligned regardless of the number of data bits defined by FSLEN in SSC\_RFMR.

### SSC Transmit Synchronization Holding Register

Name: SSC\_TSHR

Access Type: Read/Write

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

- TSDAT: Transmit Synchronization Data**

Right aligned regardless of the number of data bits defined by FSLEN in SSC\_TFMR.

## SSC Status Register

**Register Name:** SSC\_SR

**Access Type:** Read-only

|        |       |       |       |        |       |         |       |
|--------|-------|-------|-------|--------|-------|---------|-------|
| 31     | 30    | 29    | 28    | 27     | 26    | 25      | 24    |
| –      | –     | –     | –     | –      | –     | –       | –     |
| 23     | 22    | 21    | 20    | 19     | 18    | 17      | 16    |
| –      | –     | –     | –     | –      | –     | RXEN    | TXEN  |
| 15     | 14    | 13    | 12    | 11     | 10    | 9       | 8     |
| –      | –     | –     | –     | RXSYN  | TXSYN | –       | –     |
| 7      | 6     | 5     | 4     | 3      | 2     | 1       | 0     |
| RXBUFF | ENDRX | OVRUN | RXRDY | TXBUFE | ENDTX | TXEMPTY | TXRDY |

- **TXRDY: Transmit Ready**

0: Data has been loaded in SSC\_THR and is waiting to be loaded in the Transmit Shift Register.

1: SSC\_THR is empty.

- **TXEMPTY: Transmit Empty**

0: Data remains in SSC\_THR or is currently transmitted from Transmit Shift Register.

1: Last data written in SSC\_THR has been loaded in Transmit Shift Register and transmitted by it.

- **ENDTX: End of Transmission**

0: The register SSC\_TCR has not reached 0 since the last write in SSC\_TCR or SSC\_TNCR.

1: The register SSC\_TCR has reached 0 since the last write in SSC\_TCR or SSC\_TNCR.

- **TXBUFE: Transmit Buffer Empty**

0: SSC\_TCR or SSC\_TNCR have a value other than 0.

1: Both SSC\_TCR and SSC\_TNCR have a value of 0.

- **RXRDY: Receive Ready**

0: SSC\_RHR is empty.

1: Data has been received and loaded in SSC\_RHR.

- **OVRUN: Receive Overrun**

0: No data has been loaded in SSC\_RHR while previous data has not been read since the last read of the Status Register.

1: Data has been loaded in SSC\_RHR while previous data has not yet been read since the last read of the Status Register.

- **ENDRX: End of Reception**

0: Data is written on the Receive Counter Register or Receive Next Counter Register.

1: End of PDC transfer when Receive Counter Register has arrived at zero.

- **RXBUFF: Receive Buffer Full**

0: SSC\_RCR or SSC\_RNCR have a value other than 0.

1: Both SSC\_RCR and SSC\_RNCR have a value of 0.

- **TXSYN: Transmit Sync**

0: A Tx Sync has not occurred since the last read of the Status Register.

1: A Tx Sync has occurred since the last read of the Status Register.

- **RXSYN: Receive Sync**

0: A Rx Sync has not occurred since the last read of the Status Register.

1: A Rx Sync has occurred since the last read of the Status Register.

- **TXEN: Transmit Enable**

0: Transmit data is disabled.

1: Transmit data is enabled.

- **RXEN: Receive Enable**

0: Receive data is disabled.

1: Receive data is enabled.

## SSC Interrupt Enable Register

Register Name: SSC\_IER

Access Type: Write-only

|        |       |       |       |        |       |         |       |
|--------|-------|-------|-------|--------|-------|---------|-------|
| 31     | 30    | 29    | 28    | 27     | 26    | 25      | 24    |
| –      | –     | –     | –     | –      | –     | –       | –     |
| 23     | 22    | 21    | 20    | 19     | 18    | 17      | 16    |
| –      | –     | –     | –     | –      | –     | –       | –     |
| 15     | 14    | 13    | 12    | 11     | 10    | 9       | 8     |
| –      | –     | –     | –     | RXSYN  | TXSYN | –       | –     |
| 7      | 6     | 5     | 4     | 3      | 2     | 1       | 0     |
| RXBUFF | ENDRX | OVRUN | RXRDY | TXBUFE | ENDTX | TXEMPTY | TXRDY |

- **TXRDY: Transmit Ready**
- **TXEMPTY: Transmit Empty**
- **ENDTX: End of Transmission**
- **TXBUFE: Transmit Buffer Empty**
- **RXRDY: Receive Ready**
- **OVRUN: Receive Overrun**
- **ENDRX: End of Reception**
- **RXBUFF: Receive Buffer Full**
- **TXSYN: Tx Sync**
- **RXSYN: Rx Sync**

0: No effect.

1: Enables the corresponding interrupt.

**SSC Interrupt Disable Register**

Register Name: SSC\_IDR

Access Type: Write-only

|        |       |       |       |        |       |         |       |
|--------|-------|-------|-------|--------|-------|---------|-------|
| 31     | 30    | 29    | 28    | 27     | 26    | 25      | 24    |
| –      | –     | –     | –     | –      | –     | –       | –     |
| 23     | 22    | 21    | 20    | 19     | 18    | 17      | 16    |
| –      | –     | –     | –     | –      | –     | –       | –     |
| 15     | 14    | 13    | 12    | 11     | 10    | 9       | 8     |
| –      | –     | –     | –     | RXSYN  | TXSYN | –       | –     |
| 7      | 6     | 5     | 4     | 3      | 2     | 1       | 0     |
| RXBUFF | ENDRX | OVRUN | RXRDY | TXBUFE | ENDTX | TXEMPTY | TXRDY |

- **TXRDY: Transmit Ready**
- **TXEMPTY: Transmit Empty**
- **ENDTX: End of Transmission**
- **TXBUFE: Transmit Buffer Empty**
- **RXRDY: Receive Ready**
- **OVRUN: Receive Overrun**
- **ENDRX: End of Reception**
- **RXBUFF: Receive Buffer Full**
- **TXSYN: Tx Sync**
- **RXSYN: Rx Sync**

0: No effect.

1: Disables the corresponding interrupt.

## SSC Interrupt Mask Register

Register Name: SSC\_IMR

Access Type: Read-only

|        |       |       |       |        |       |         |       |
|--------|-------|-------|-------|--------|-------|---------|-------|
| 31     | 30    | 29    | 28    | 27     | 26    | 25      | 24    |
| –      | –     | –     | –     | –      | –     | –       | –     |
| 23     | 22    | 21    | 20    | 19     | 18    | 17      | 16    |
| –      | –     | –     | –     | –      | –     | –       | –     |
| 15     | 14    | 13    | 12    | 11     | 10    | 9       | 8     |
| –      | –     | –     | –     | RXSYN  | TXSYN | –       | –     |
| 7      | 6     | 5     | 4     | 3      | 2     | 1       | 0     |
| RXBUFF | ENDRX | OVRUN | RXRDY | TXBUFE | ENDTX | TXEMPTY | TXRDY |

- **TXRDY: Transmit Ready**
- **TXEMPTY: Transmit Empty**
- **ENDTX: End of Transmission**
- **TXBUFE: Transmit Buffer Empty**
- **RXRDY: Receive Ready**
- **OVRUN: Receive Overrun**
- **ENDRX: End of Reception**
- **RXBUFF: Receive Buffer Full**
- **TXSYN: Tx Sync**
- **RXSYN: Rx Sync**

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

## Timer Counter (TC)

### Overview

The Timer Counter (TC) includes 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 channel has three external clock inputs, five internal clock inputs and two 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.

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 channel, allowing them to be chained.

Key Features of the Timer Counter are:

- Three 16-bit Timer Counter Channels
- A Wide Range of Functions Including:
  - Frequency Measurement
  - Event Counting
  - Interval Measurement
  - Pulse Generation
  - Delay Timing
  - Pulse Width Modulation
  - Up/down Capabilities
- Each Channel is User-configurable and Contains:
  - Three External Clock Inputs
  - Five Internal Clock Inputs
  - Two Multi-purpose Input/Output Signals
- Internal Interrupt Signal

Two Global Registers that Act on All Three TC Channels

# Block Diagram

Figure 220. Timer Counter Block Diagram

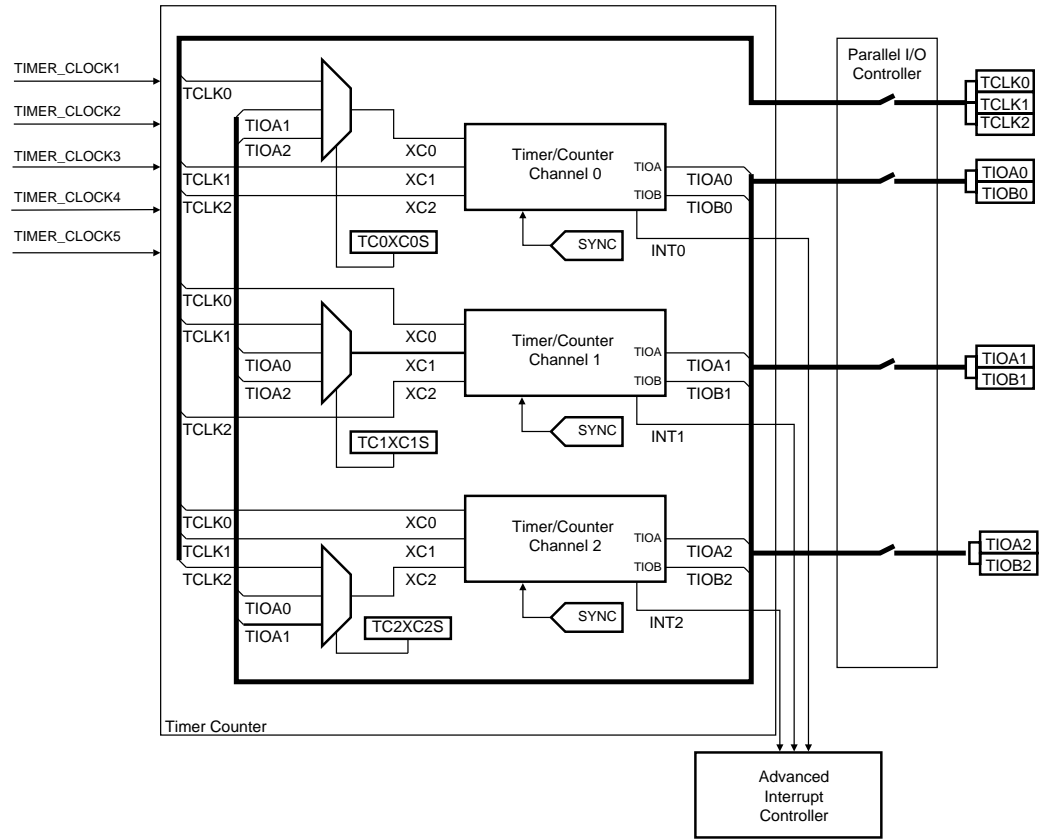


Table 89. Signal Name Description

| Block/Channel  | Signal Name         | Description  |
|----------------|---------------------|--|
| Channel Signal | XC0, XC1, XC2       | External Clock Inputs  |
|                | TIOA                | Capture Mode: General-purpose Input<br>Waveform Mode: General-purpose Output       |
|                | TIOB                | Capture Mode: General-purpose Input<br>Waveform Mode: General-purpose Input/output |
|                | INT                 | Interrupt Signal Output  |
|                | SYNC                | Synchronization Input Signal   |
| Block Signal   | TCLK0, TCLK1, TCLK2 | External Clock Inputs  |
|                | 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  |



## Pin Name List

**Table 90.** Timer Counter pin list

| Pin Name    | Description          | Type  |
|-------------|----------------------|-------|
| TCLK0-TCLK2 | External Clock Input | Input |
| TIOA0-TIOA2 | I/O Line A           | I/O   |
| TIOB0-TIOB2 | I/O Line B           | I/O   |

### Product Dependencies

For further details on the Timer Counter hardware implementation, see the specific Product Properties document.

### I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the TC pins to their peripheral functions.

### Power Management

The TC must be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the Timer Counter.

### Interrupt

The TC interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling the TC interrupt requires programming the AIC before configuring the TC.

## Functional Description

### TC Description

The three channels of the Timer Counter are independent and identical in operation. The registers for channel programming are listed in Table 90 on page 475.

### 16-bit Counter

Each channel is organized around a 16-bit counter. The value of the counter is incremented at each positive edge of the selected clock. When the counter has reached the value 0xFFFF and passes to 0x0000, an overflow occurs and the COVFS bit in TC\_SR (Status Register) is set.

The current value of the counter is accessible in real time by reading the Counter Value Register, 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 selected 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). See Figure 221.

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

- Internal clock signals: TIMER\_CLOCK1, TIMER\_CLOCK2, TIMER\_CLOCK3, TIMER\_CLOCK4, TIMER\_CLOCK5
- External clock signals: XC0, XC1 or XC2

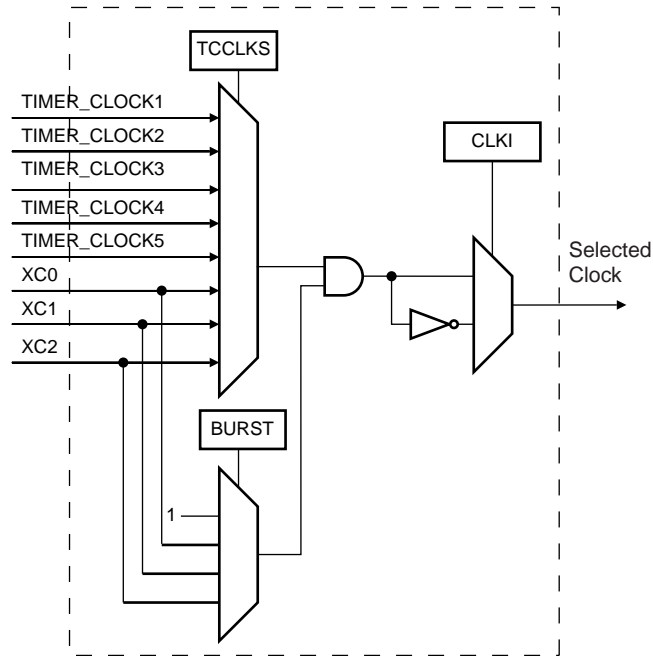
This selection is made by the TCCLKS bits in the TC Channel Mode Register (Capture Mode).

The selected clock can be inverted with the CLKI bit in TC\_CMR (Capture 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 master clock period. The external clock frequency must be at least 2.5 times lower than the master clock

**Figure 221.** Clock Selection

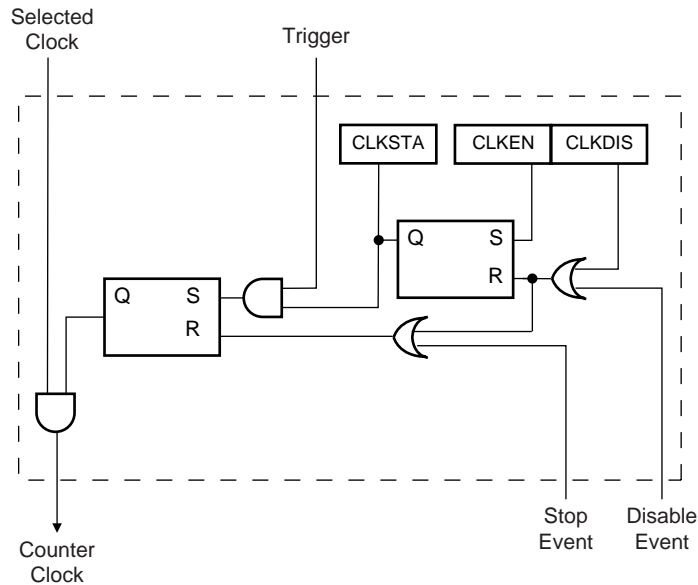


### Clock Control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See Figure 222.

- 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 222. Clock Control



**TC Operating Modes**

Each channel can independently operate in two different modes:

- Capture Mode provides measurement on signals.
- Waveform Mode provides wave generation.

The TC Operating Mode is programmed with the WAVE bit in the TC Channel 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 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 master clock period in order to be detected.

Regardless of the trigger used, it will be taken into account at the following active edge of the selected clock. This means that the counter value can be read differently from zero just after a trigger, especially when a low frequency signal is selected as the clock.

## 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 inputs.

Figure 223 shows the configuration of the TC channel when programmed in Capture Mode.

## Capture Registers A and B

Registers A and B (RA and RB) 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 LDRA parameter in TC\_CMR defines the TIOA edge for the loading of register A, and the LDRB parameter 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.

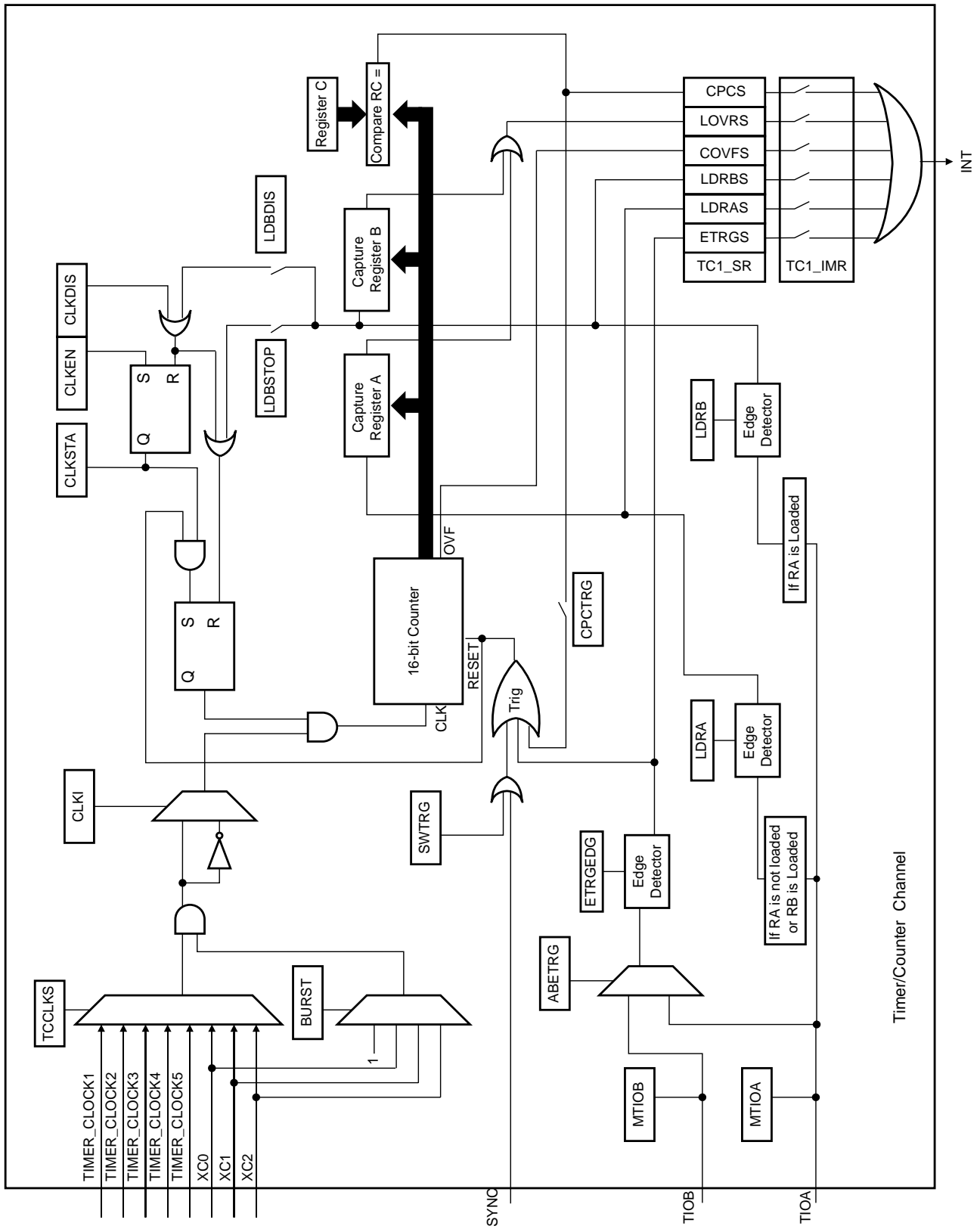
Loading RA or RB before the read of the last value loaded sets the Overrun Error Flag (LOVRS) in TC\_SR (Status Register). In this case, the old value is overwritten.

## Trigger Conditions

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

The ABETRG bit in TC\_CMR selects TIOA or TIOB input signal as an external trigger. The ETRGEDG parameter defines the edge (rising, falling or both) detected to generate an external trigger. If ETRGEDG = 0 (none), the external trigger is disabled.

Figure 223. Capture Mode



## Waveform Operating Mode

Waveform operating mode is entered by setting the WAVE parameter in TC\_CMR (Channel Mode Register).

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

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

Figure 224 shows the configuration of the TC channel when programmed in Waveform Operating Mode.

## Waveform Selection

Depending on the WAVSEL parameter in TC\_CMR (Channel Mode Register), the behavior of TC\_CV varies.

With any selection, RA, RB and RC can all be used as compare registers.

RA Compare is used to control the TIOA output, RB Compare is used to control the TIOB output (if correctly configured) and RC Compare is used to control TIOA and/or TIOB outputs.



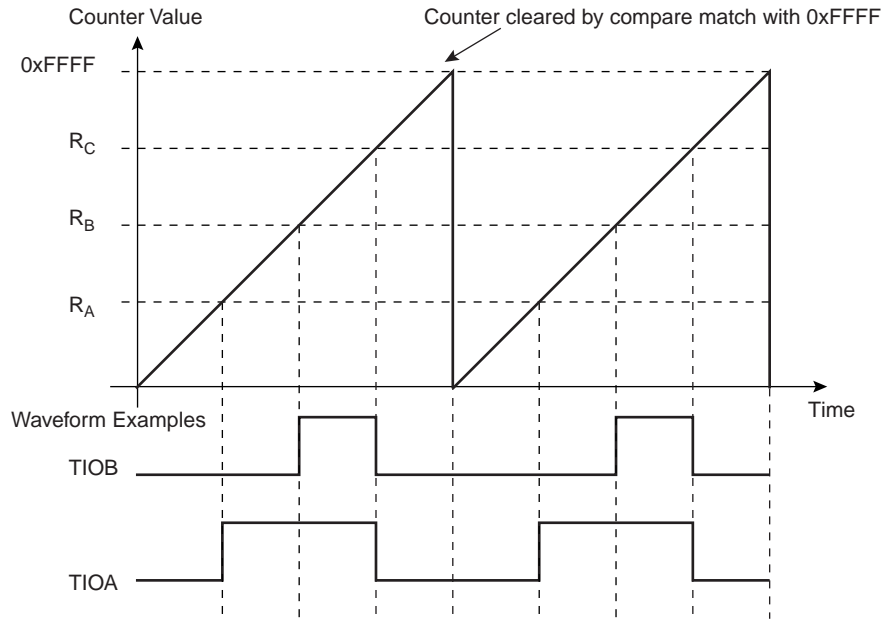
WAVSEL = 00

When WAVSEL = 00, the value of TC\_CV is incremented from 0 to 0xFFFF. Once 0xFFFF has been reached, the value of TC\_CV is reset. Incrementation of TC\_CV starts again and the cycle continues. See Figure 225.

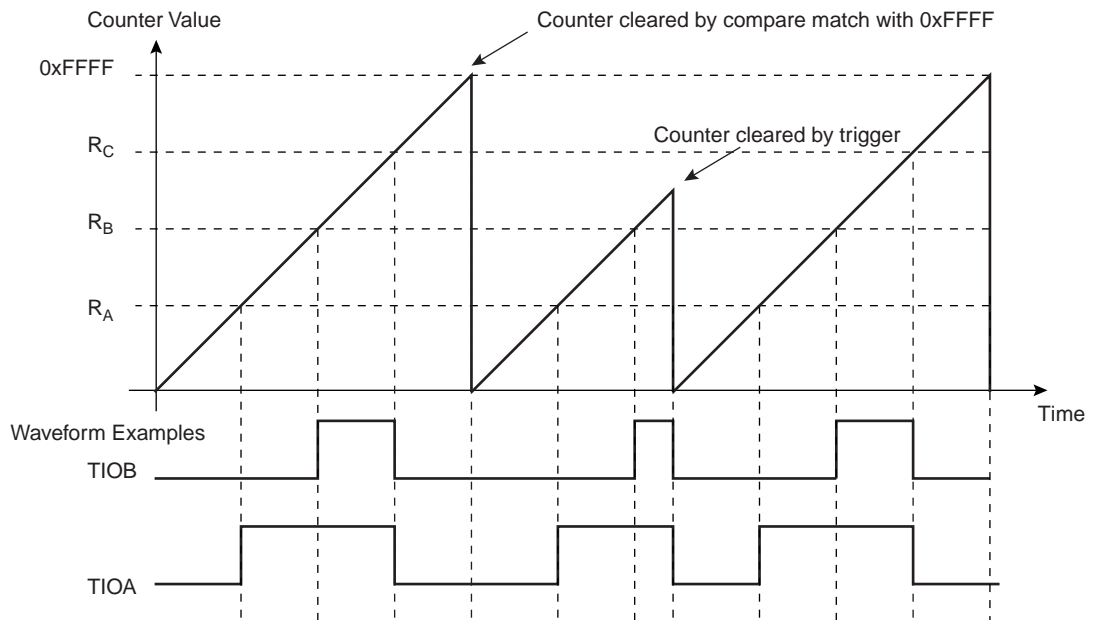
An external event trigger or a software trigger can reset the value of TC\_CV. It is important to note that the trigger may occur at any time. See Figure 226.

RC Compare cannot be programmed to generate a trigger in this configuration. At the same time, RC Compare can stop the counter clock (CPCSTOP = 1 in TC\_CMR) and/or disable the counter clock (CPCDIS = 1 in TC\_CMR).

**Figure 225.** WAVSEL= 00 without trigger



**Figure 226.** WAVSEL= 00 with trigger





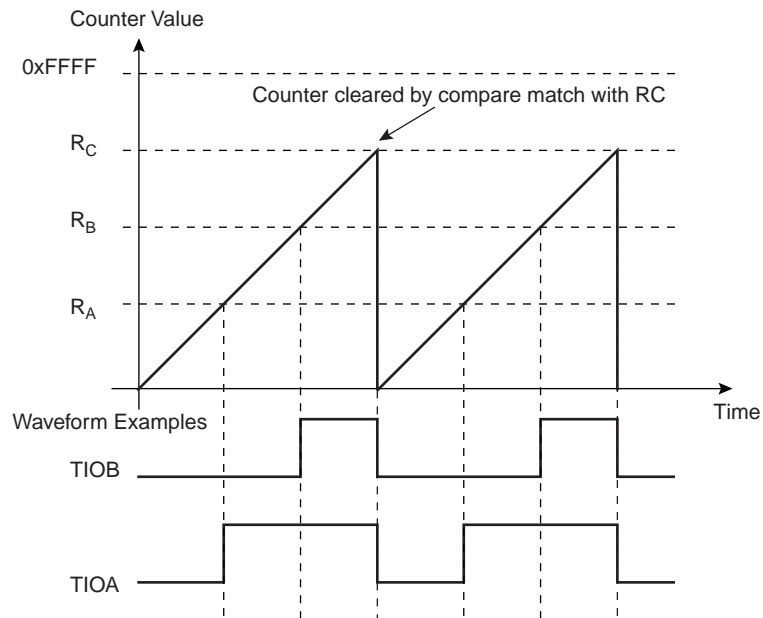
WAVSEL = 10

When WAVSEL = 10, the value of TC\_CV is incremented from 0 to the value of RC, then automatically reset on a RC Compare. Once the value of TC\_CV has been reset, it is then incremented and so on. See Figure 227.

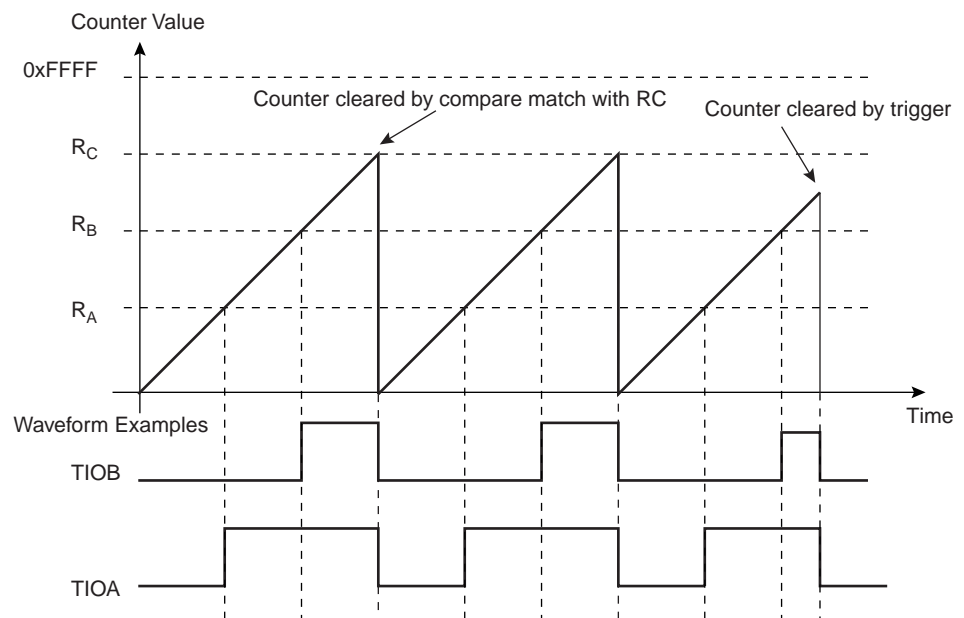
It is important to note that TC\_CV can be reset at any time by an external event or a software trigger if both are programmed correctly. See Figure 228.

In addition, RC Compare can stop the counter clock (CPCSTOP = 1 in TC\_CMR) and/or disable the counter clock (CPCDIS = 1 in TC\_CMR).

**Figure 227.** WAVSEL = 10 Without Trigger



**Figure 228.** WAVSEL = 10 With Trigger



WAVSEL = 01

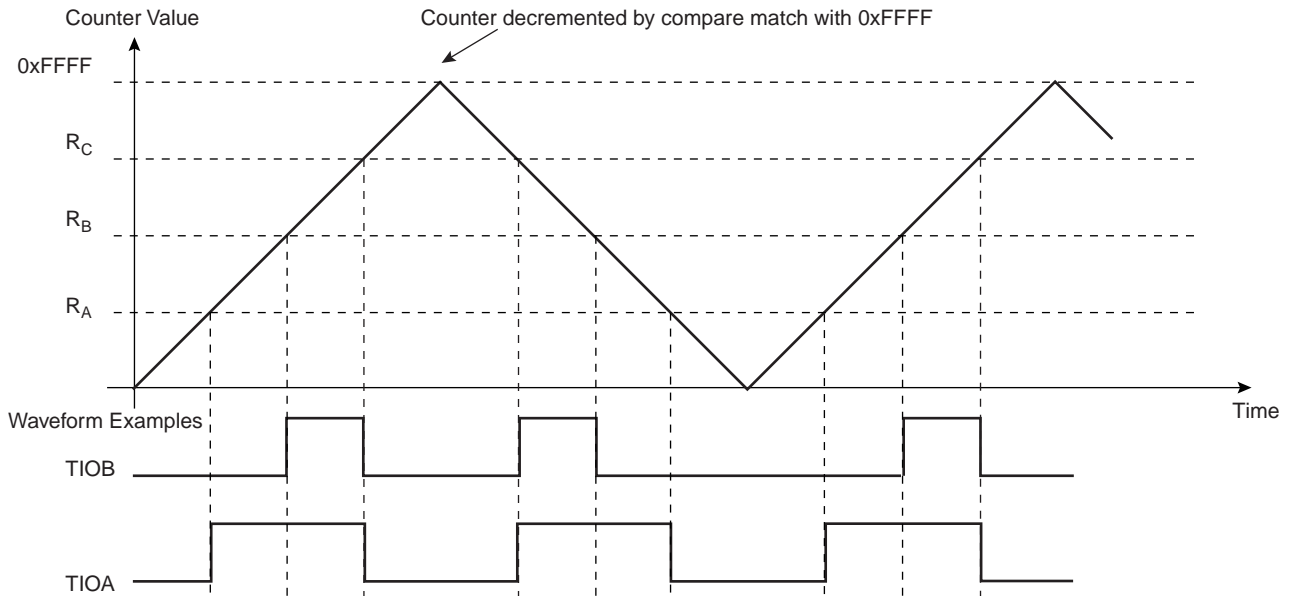
When WAVSEL = 01, the value of TC\_CV is incremented from 0 to 0xFFFF. Once 0xFFFF is reached, the value of TC\_CV is decremented to 0, then re-incremented to 0xFFFF and so on. See Figure 229.

A trigger such as an external event or a software trigger can modify TC\_CV at any time. If a trigger occurs while TC\_CV is incrementing, TC\_CV then decrements. If a trigger is received while TC\_CV is decrementing, TC\_CV then increments. See Figure 230.

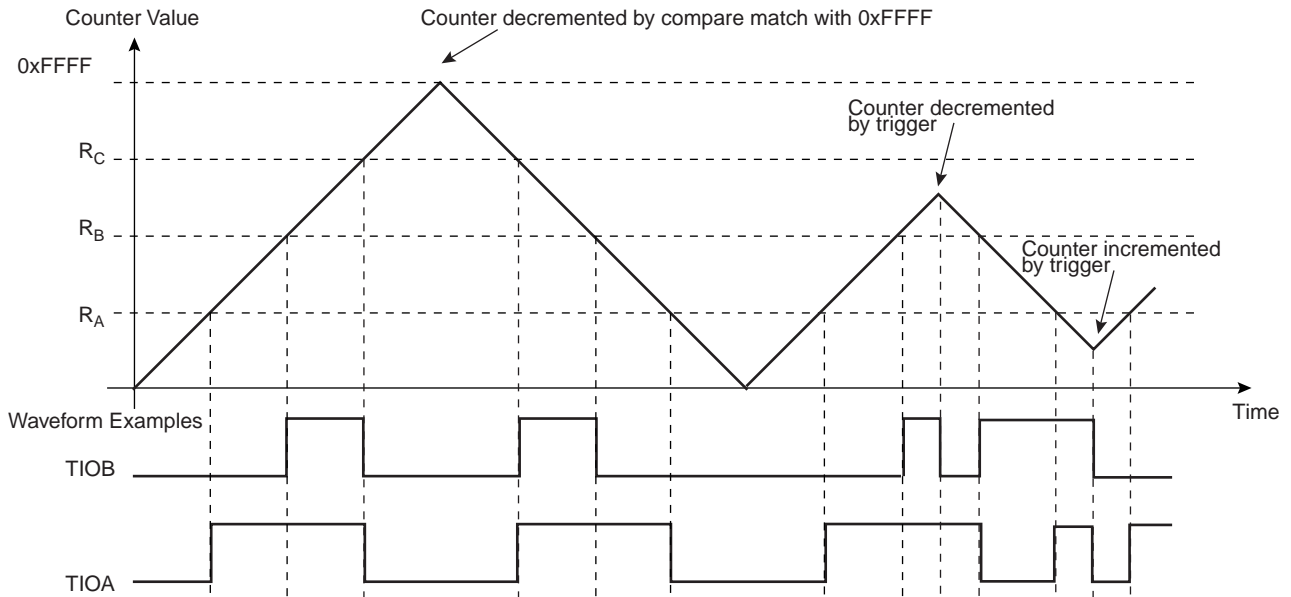
RC Compare cannot be programmed to generate a trigger in this configuration.

At the same time, RC Compare can stop the counter clock (CPCSTOP = 1) and/or disable the counter clock (CPCDIS = 1).

**Figure 229.** WAVSEL = 01 Without Trigger



**Figure 230.** WAVSEL = 01 With Trigger



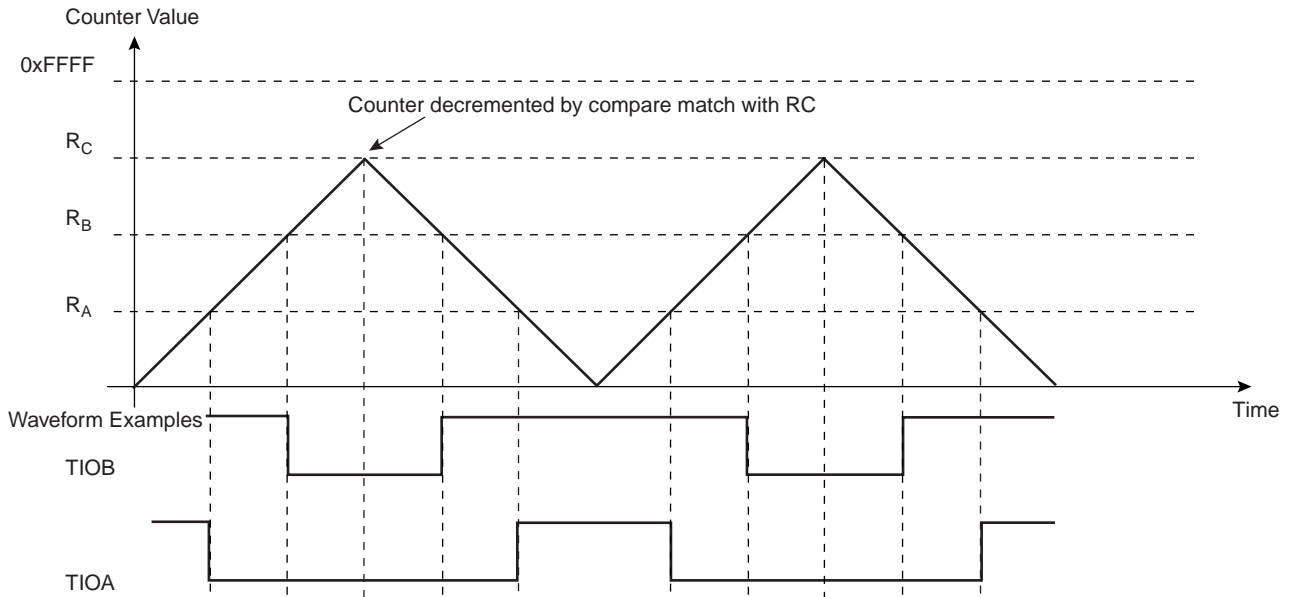
WAVSEL = 11

When WAVSEL = 11, the value of TC\_CV is incremented from 0 to RC. Once RC is reached, the value of TC\_CV is decremented to 0, then re-incremented to RC and so on. See Figure 231.

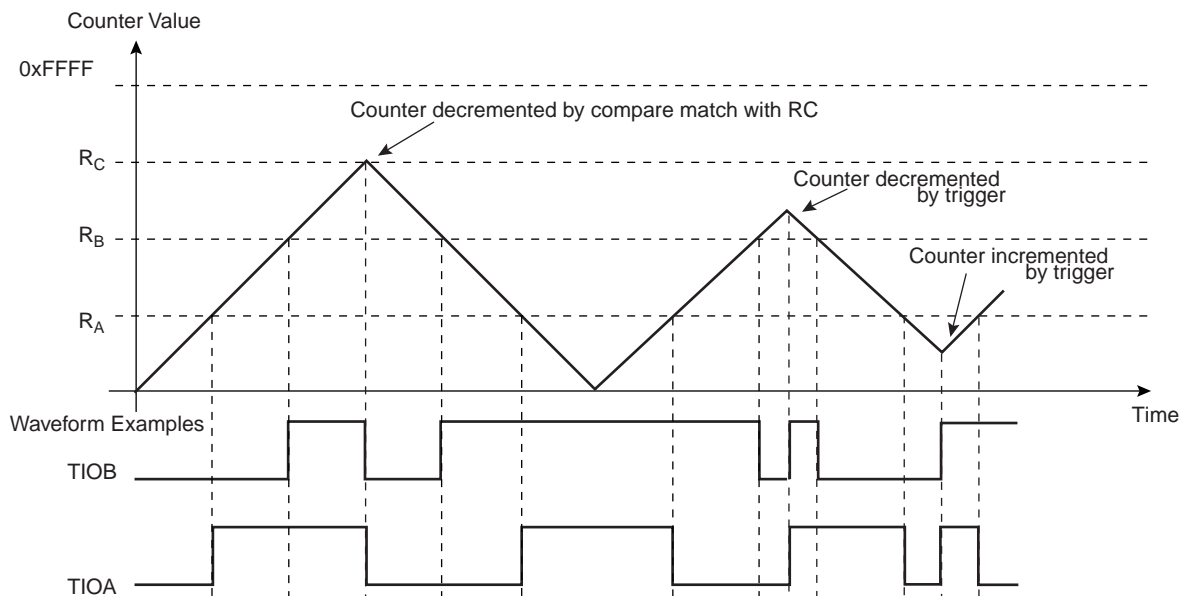
A trigger such as an external event or a software trigger can modify TC\_CV at any time. If a trigger occurs while TC\_CV is incrementing, TC\_CV then decrements. If a trigger is received while TC\_CV is decrementing, TC\_CV then increments. See Figure 232.

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

**Figure 231.** WAVSEL = 11 Without Trigger



**Figure 232.** WAVSEL = 11 With Trigger



### **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 parameter in TC\_CMR selects the external trigger. The EEVTEDG parameter 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 an 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 and the software trigger are also available as triggers. RC Compare can also be used as a trigger depending on the parameter WAVSEL.

### **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.

## Timer Counter (TC) User Interface

**Table 91.** Timer Counter Global Memory Map

| Offset | Channel/Register          | Name   | Access       | Reset Value |
|--------|---------------------------|--------|--------------|-------------|
| 0x00   | TC Channel 0              |        | See Table 92 |             |
| 0x40   | TC Channel 1              |        | See Table 92 |             |
| 0x80   | TC Channel 2              |        | See Table 92 |             |
| 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 whole TC block. TC channels are controlled by the registers listed in Table 92. The offset of each of the channel registers in Table 92 is in relation to the offset of the corresponding channel as mentioned in Table 92.

**Table 92.** Timer Counter Channel Memory Map

| Offset | Register                   | Name   | Access                    | Reset Value |
|--------|----------------------------|--------|---------------------------|-------------|
| 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-only                 | 0           |
| 0x14   | Register A                 | TC_RA  | Read/Write <sup>(1)</sup> | 0           |
| 0x18   | Register B                 | TC_RB  | Read/Write <sup>(1)</sup> | 0           |
| 0x1C   | Register C                 | TC_RC  | Read/Write                | 0           |
| 0x20   | Status Register            | TC_SR  | Read-only                 | 0           |
| 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           |

Notes: 1. Read only if WAVE = 0

## TC Block Control Register

Register Name: TC\_BCR

Access Type: Write-only

|    |    |    |    |    |    |    |      |
|----|----|----|----|----|----|----|------|
| 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**

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

|    |    |         |    |        |    |         |    |
|----|----|---------|----|--------|----|---------|----|
| 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 |    | TCXC1S |    | TC0XC0S |    |

- **TC0XC0S: External Clock Signal 0 Selection**

| TC0XC0S |   | Signal Connected to XC0 |
|---------|---|-------------------------|
| 0       | 0 | TCLK0                   |
| 0       | 1 | none                    |
| 1       | 0 | TIOA1                   |
| 1       | 1 | TIOA2                   |

- **TC1XC1S: External Clock Signal 1 Selection**

| TC1XC1S |   | Signal Connected to XC1 |
|---------|---|-------------------------|
| 0       | 0 | TCLK1                   |
| 0       | 1 | none                    |
| 1       | 0 | TIOA0                   |
| 1       | 1 | TIOA2                   |

• **TC2XC2S: External Clock Signal 2 Selection**

| TC2XC2S |   | Signal Connected to XC2 |
|---------|---|-------------------------|
| 0       | 0 | TCLK2                   |
| 0       | 1 | none                    |
| 1       | 0 | TIOA0                   |
| 1       | 1 | TIOA1                   |

**TC Channel Control Register**

**Register Name:** TC\_CCR

**Access Type:** Write-only

|    |    |    |    |    |       |        |       |
|----|----|----|----|----|-------|--------|-------|
| 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**

0 = No effect.

1 = Enables the clock if CLKDIS is not 1.

• **CLKDIS: Counter Clock Disable Command**

0 = No effect.

1 = Disables the clock.

• **SWTRG: Software Trigger Command**

0 = No effect.

1 = A software trigger is performed: the counter is reset and the clock is started.

## TC Channel Mode Register: Capture Mode

Register Name: TC\_CMCR

Access Type: Read/Write

|          |         |       |    |      |        |         |    |
|----------|---------|-------|----|------|--------|---------|----|
| 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 |
|--------|---|---|----------------|
| 0      | 0 | 0 | TIMER_CLOCK1   |
| 0      | 0 | 1 | TIMER_CLOCK2   |
| 0      | 1 | 0 | TIMER_CLOCK3   |
| 0      | 1 | 1 | TIMER_CLOCK4   |
| 1      | 0 | 0 | TIMER_CLOCK5   |
| 1      | 0 | 1 | XC0            |
| 1      | 1 | 0 | XC1            |
| 1      | 1 | 1 | XC2            |

- **CLKI: Clock Invert**

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 |   |   |
|-------|---|---|
| 0     | 0 | The clock is not gated by an external signal. |
| 0     | 1 | XC0 is ANDed with the selected clock.         |
| 1     | 0 | XC1 is ANDed with the selected clock.         |
| 1     | 1 | XC2 is ANDed with the selected clock.         |

- **LDBSTOP: Counter Clock Stopped with RB Loading**

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**

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         |
|---------|---|--------------|
| 0       | 0 | none         |
| 0       | 1 | rising edge  |
| 1       | 0 | falling edge |
| 1       | 1 | each edge    |

- **ABETRG: TIOA or TIOB External Trigger Selection**

0 = TIOB is used as an external trigger.

1 = TIOA is used as an external trigger.

- **CPCTRG: RC Compare Trigger Enable**

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 = Capture Mode is enabled.

1 = Capture Mode is disabled (Waveform Mode is enabled).

- **LDRA: RA Loading Selection**

| LDRA |   | Edge                 |
|------|---|----------------------|
| 0    | 0 | none                 |
| 0    | 1 | rising edge of TIOA  |
| 1    | 0 | falling edge of TIOA |
| 1    | 1 | each edge of TIOA    |

- **LDRB: RB Loading Selection**

| LDRB |   | Edge                 |
|------|---|----------------------|
| 0    | 0 | none                 |
| 0    | 1 | rising edge of TIOA  |
| 1    | 0 | falling edge of TIOA |
| 1    | 1 | each edge of TIOA    |

## TC Channel Mode Register: Waveform Mode

Register Name: TC\_CMCR

Access Type: Read/Write

|          |         |       |        |      |        |         |    |
|----------|---------|-------|--------|------|--------|---------|----|
| 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 | WAVSEL  |       | ENETRQ | EEVT |        | EEVTEDG |    |
| 7        | 6       | 5     | 4      | 3    | 2      | 1       | 0  |
| CPCDIS   | CPCSTOP | BURST |        | CLKI | TCCLKS |         |    |

- **TCCLKS: Clock Selection**

| TCCLKS |   |   | Clock Selected |
|--------|---|---|----------------|
| 0      | 0 | 0 | TIMER_CLOCK1   |
| 0      | 0 | 1 | TIMER_CLOCK2   |
| 0      | 1 | 0 | TIMER_CLOCK3   |
| 0      | 1 | 1 | TIMER_CLOCK4   |
| 1      | 0 | 0 | TIMER_CLOCK5   |
| 1      | 0 | 1 | XC0            |
| 1      | 1 | 0 | XC1            |
| 1      | 1 | 1 | XC2            |

- **CLKI: Clock Invert**

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 |   |   |
|-------|---|---|
| 0     | 0 | The clock is not gated by an external signal. |
| 0     | 1 | XC0 is ANDed with the selected clock.         |
| 1     | 0 | XC1 is ANDed with the selected clock.         |
| 1     | 1 | XC2 is ANDed with the selected clock.         |

- **CPCSTOP: Counter Clock Stopped with RC Compare**

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**

0 = Counter clock is not disabled when counter reaches RC.

1 = Counter clock is disabled when counter reaches RC.

- **EEVTEG: External Event Edge Selection**

| EEVTEG |   | Edge         |
|--------|---|--------------|
| 0      | 0 | none         |
| 0      | 1 | rising edge  |
| 1      | 0 | falling edge |
| 1      | 1 | each edge    |

- **EEVT: External Event Selection**

| EEVT |   | Signal selected as external event | TIOB Direction       |
|------|---|-----------------------------------|----------------------|
| 0    | 0 | TIOB                              | input <sup>(1)</sup> |
| 0    | 1 | XC0                               | output               |
| 1    | 0 | XC1                               | output               |
| 1    | 1 | XC2                               | output               |

Note: 1. If TIOB is chosen as the external event signal, it is configured as an input and no longer generates waveforms.

- **ENETR: External Event Trigger Enable**

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.

- **WAVSEL: Waveform Selection**

| WAVSEL |   | Effect  |
|--------|---|---|
| 0      | 0 | UP mode without automatic trigger on RC Compare     |
| 1      | 0 | UP mode with automatic trigger on RC Compare        |
| 0      | 1 | UPDOWN mode without automatic trigger on RC Compare |
| 1      | 1 | UPDOWN mode with automatic trigger on RC Compare    |

- **WAVE = 1**

0 = Waveform Mode is disabled (Capture Mode is enabled).

1 = Waveform Mode is enabled.

- **ACPA: RA Compare Effect on TIOA**

| ACPA |   | Effect |
|------|---|--------|
| 0    | 0 | none   |
| 0    | 1 | set    |
| 1    | 0 | clear  |
| 1    | 1 | toggle |

- **ACPC: RC Compare Effect on TIOA**

| ACPC |   | Effect |
|------|---|--------|
| 0    | 0 | none   |
| 0    | 1 | set    |
| 1    | 0 | clear  |
| 1    | 1 | toggle |

- **AAEVT: External Event Effect on TIOA**

| AAEVT |   | Effect |
|-------|---|--------|
| 0     | 0 | none   |
| 0     | 1 | set    |
| 1     | 0 | clear  |
| 1     | 1 | toggle |

- **ASWTRG: Software Trigger Effect on TIOA**

| ASWTRG |   | Effect |
|--------|---|--------|
| 0      | 0 | none   |
| 0      | 1 | set    |
| 1      | 0 | clear  |
| 1      | 1 | toggle |

- **BCPB: RB Compare Effect on TIOB**

| BCPB |   | Effect |
|------|---|--------|
| 0    | 0 | none   |
| 0    | 1 | set    |
| 1    | 0 | clear  |
| 1    | 1 | toggle |

- **BCPC: RC Compare Effect on TIOB**

| BCPC |   | Effect |
|------|---|--------|
| 0    | 0 | none   |
| 0    | 1 | set    |
| 1    | 0 | clear  |
| 1    | 1 | toggle |

- **BEEVT: External Event Effect on TIOB**

| BEEVT |   | Effect |
|-------|---|--------|
| 0     | 0 | none   |
| 0     | 1 | set    |
| 1     | 0 | clear  |
| 1     | 1 | toggle |

- **BSWTRG: Software Trigger Effect on TIOB**

| BSWTRG |   | Effect |
|--------|---|--------|
| 0      | 0 | none   |
| 0      | 1 | set    |
| 1      | 0 | clear  |
| 1      | 1 | toggle |

## TC Counter Value Register

**Register Name:** TC\_CV

**Access Type:** Read-only

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 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**

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

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 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**

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

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 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**

RB contains the Register B value in real time.

## TC Register C

Register Name: TC\_RC

Access Type: Read/Write

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 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**

RC contains the Register C value in real time.

## TC Status Register

Register Name: TC\_SR

Access Type: Read-only

|       |       |       |      |      |       |       |        |
|-------|-------|-------|------|------|-------|-------|--------|
| 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**

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**

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**

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**

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**

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**

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**

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**

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**

0 = Clock is disabled.

1 = Clock is enabled.

- **MTIOA: TIOA Mirror**

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**

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

|       |       |       |      |      |      |       |       |
|-------|-------|-------|------|------|------|-------|-------|
| 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**

0 = No effect.

1 = Enables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun**

0 = No effect.

1 = Enables the Load Overrun Interrupt.

- **CPAS: RA Compare**

0 = No effect.

1 = Enables the RA Compare Interrupt.

- **CPBS: RB Compare**

0 = No effect.

1 = Enables the RB Compare Interrupt.

- **CPCS: RC Compare**

0 = No effect.

1 = Enables the RC Compare Interrupt.

- **LDRAS: RA Loading**

0 = No effect.

1 = Enables the RA Load Interrupt.

- **LDRBS: RB Loading**

0 = No effect.

1 = Enables the RB Load Interrupt.

- **ETRGS: External Trigger**

0 = No effect.

1 = Enables the External Trigger Interrupt.



**TC Interrupt Disable Register**

Register Name: TC\_IDR

Access Type: Write-only

|       |       |       |      |      |      |       |       |
|-------|-------|-------|------|------|------|-------|-------|
| 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**

0 = No effect.

1 = Disables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun**

0 = No effect.

1 = Disables the Load Overrun Interrupt (if WAVE = 0).

- **CPAS: RA Compare**

0 = No effect.

1 = Disables the RA Compare Interrupt (if WAVE = 1).

- **CPBS: RB Compare**

0 = No effect.

1 = Disables the RB Compare Interrupt (if WAVE = 1).

- **CPCS: RC Compare**

0 = No effect.

1 = Disables the RC Compare Interrupt.

- **LDRAS: RA Loading**

0 = No effect.

1 = Disables the RA Load Interrupt (if WAVE = 0).

- **LDRBS: RB Loading**

0 = No effect.

1 = Disables the RB Load Interrupt (if WAVE = 0).

- **ETRGS: External Trigger**

0 = No effect.

1 = Disables the External Trigger Interrupt.

## TC Interrupt Mask Register

Register Name: TC\_IMR

Access Type: Read-only

|       |       |       |      |      |      |       |       |
|-------|-------|-------|------|------|------|-------|-------|
| 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**

0 = The Counter Overflow Interrupt is disabled.

1 = The Counter Overflow Interrupt is enabled.

- **LOVRS: Load Overrun**

0 = The Load Overrun Interrupt is disabled.

1 = The Load Overrun Interrupt is enabled.

- **CPAS: RA Compare**

0 = The RA Compare Interrupt is disabled.

1 = The RA Compare Interrupt is enabled.

- **CPBS: RB Compare**

0 = The RB Compare Interrupt is disabled.

1 = The RB Compare Interrupt is enabled.

- **CPCS: RC Compare**

0 = The RC Compare Interrupt is disabled.

1 = The RC Compare Interrupt is enabled.

- **LDRAS: RA Loading**

0 = The Load RA Interrupt is disabled.

1 = The Load RA Interrupt is enabled.

- **LDRBS: RB Loading**

0 = The Load RB Interrupt is disabled.

1 = The Load RB Interrupt is enabled.

- **ETRGS: External Trigger**

0 = The External Trigger Interrupt is disabled.

1 = The External Trigger Interrupt is enabled.

## MultiMedia Card Interface (MCI)

### Overview

The MultiMedia Card Interface (MCI) supports the MultiMediaCard (MMC) Specification V2.2 and the SD Memory Card Specification V1.0.

The MCI includes a command register, response registers, data registers, timeout counters and error detection logic that automatically handle the transmission of commands and, when required, the reception of the associated responses and data with limited processor overhead.

The MCI supports stream, block and multi-block data read and write, and is compatible with the Peripheral Data Controller channels, minimizing processor intervention for large buffer transfers.

The MCI operates at a rate of up to Master Clock divided by 2 and supports interfacing of up to 16 slots (depending on the product). Each slot may be used to interface with a MultiMediaCard bus (up to 30 Cards) or with an SD Memory Card. Only one slot can be selected at a time (slots are multiplexed). A bit in the Command Register performs this selection.

The SD Memory Card communication is based on a 9-pin interface (clock, command, four data and three power lines) and the MultiMediaCard on a 7-pin interface (clock, command, one data and three power lines).

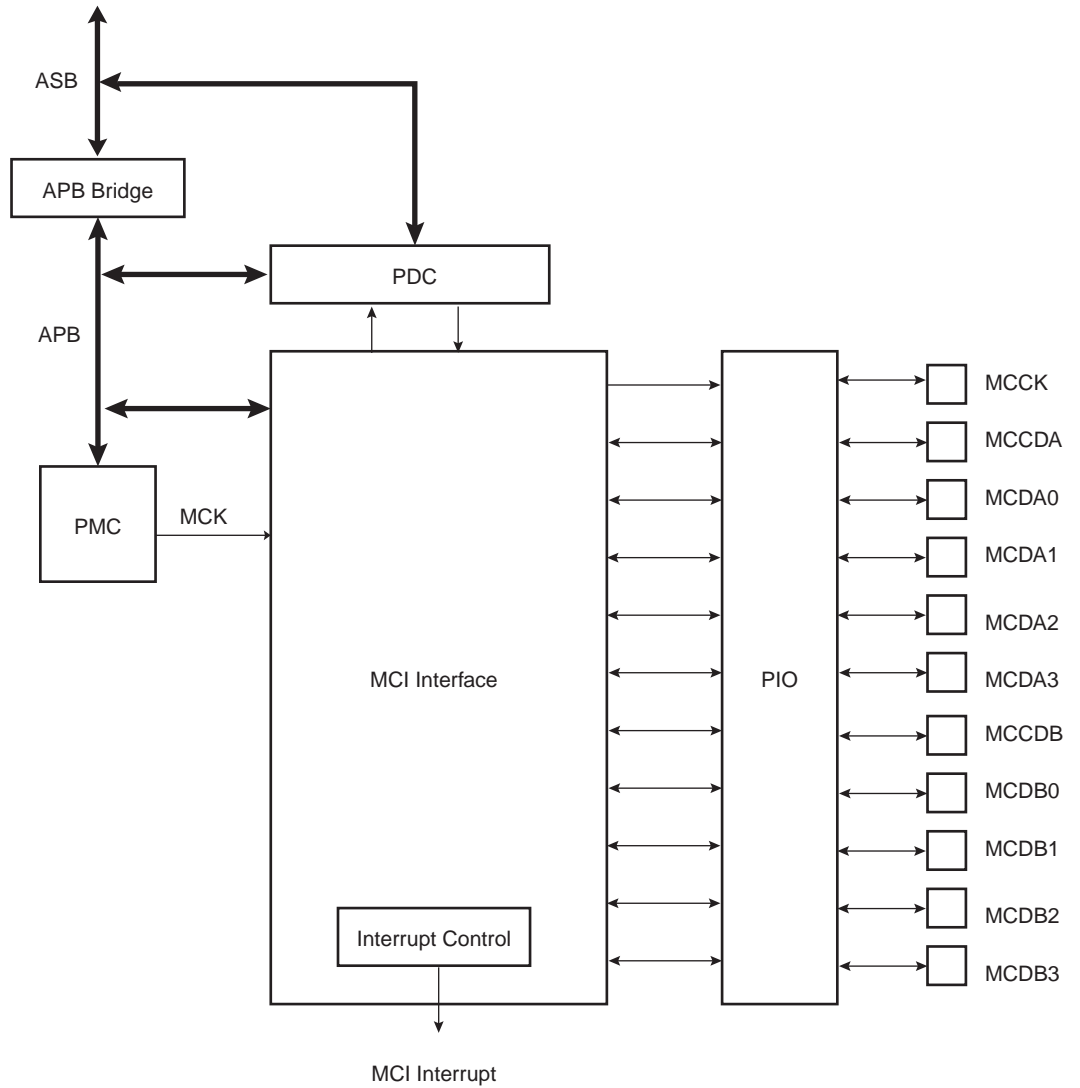
The SD Memory Card interface also supports MultiMedia Card operations. The main differences between SD and MultiMedia Cards are the initialization process and the bus topology.

The main features of the MCI are:

- Compatibility with MultiMedia Card Specification Version 2.2
- Compatibility with SD Memory Card Specification Version 1.0
- Cards clock rate up to Master Clock divided by 2
- Embedded power management to slow down clock rate when not used
- Supports up to sixteen multiplexed slots (product-dependent)
  - One slot for one MultiMediaCard bus (up to 30 cards) or one SD Memory Card
- Support for stream, block and multi-block data read and write
- Supports connection to Peripheral Data Controller
  - Minimizes processor intervention for large buffer transfers

# Block Diagram

Figure 233. Block Diagram



## Application Block Diagram

Figure 234. Application Block Diagram

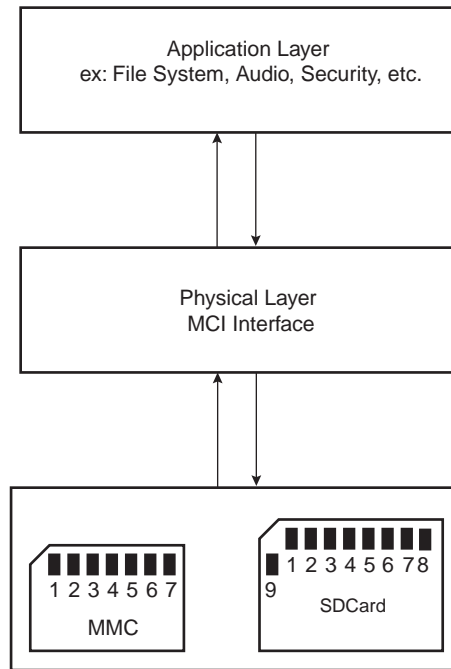


Table 93. I/O Lines Description

| Pin Name      | Pin Description     | Type <sup>(1)</sup> | Comments                                  |
|---------------|---------------------|---------------------|---|
| MCCDA/MCCDB   | Command/response    | I/O/PP/OD           | CMD of an MMC or SD Card                  |
| MCKK          | Clock               | I                   | CLK of an MMC or SD Card                  |
| MCDA0 - MCDA3 | Data 0..3 of Slot A | I/O/PP              | DAT0 of an MMC<br>DAT[0..3] of an SD Card |
| MCDB0 - MCDB3 | Data 0..3 of Slot B | I/O/PP              | DAT0 of an MMC<br>DAT[0..3] of an SD Card |

Note: 1. I: Input, O: Output, PP: Push/Pull, OD: Open Drain.

## Product Dependencies

### I/O Lines

The pins used for interfacing the MultiMedia Cards or SD Cards may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the peripheral functions to MCI pins.

### Power Management

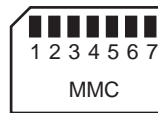
The MCI may be clocked through the Power Management Controller (PMC), so the programmer must first to configure the PMC to enable the MCI clock.

### Interrupt

The MCI interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling the MCI interrupt requires programming the AIC before configuring the MCI.

## Bus Topology

**Figure 235.** MultiMedia Memory Card Bus Topology



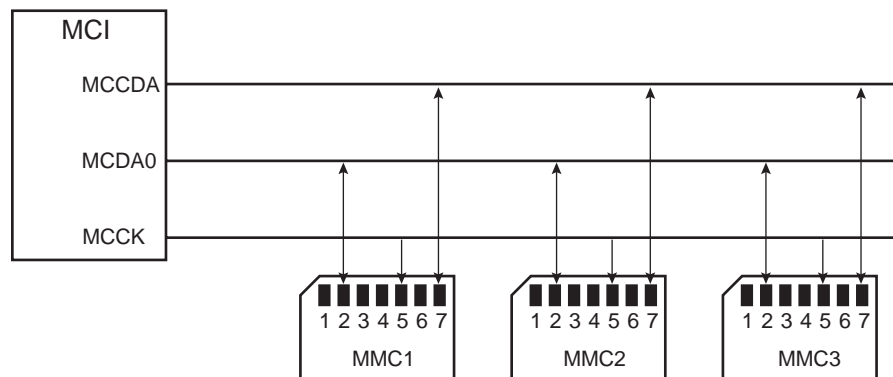
The MultiMedia Card communication is based on a 7-pin serial bus interface. It has three communication lines and four supply lines.

**Table 94.** Bus Topology

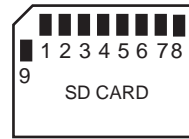
| Pin Number | Name   | Type <sup>(1)</sup> | Description           | MCI Pin Name |
|------------|--------|---------------------|-----------------------|--------------|
| 1          | RSV    | NC                  | Not connected         |              |
| 2          | CMD    | I/O/PP/OD           | Command/response      | MCCDA/MCCDB  |
| 3          | VSS1   | S                   | Supply voltage ground | VSS          |
| 4          | VDD    | S                   | Supply voltage        | VDD          |
| 5          | CLK    | I                   | Clock                 | MCCK         |
| 6          | VSS2   | S                   | Supply voltage ground | VSS          |
| 7          | DAT[0] | I/O/PP              | Data 0                | MCDA0/MCDB0  |

Note: 1. I: Input, O: Output, PP: Push/Pull, OD: Open Drain.

**Figure 236.** MMC Bus Connections



**Figure 237.** SD Memory Card Bus Topology



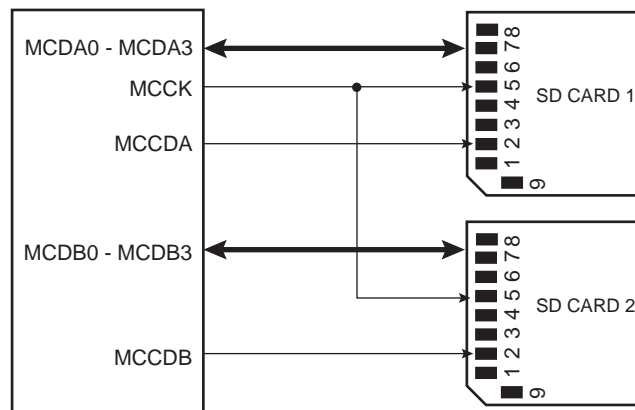
The SD Memory Card bus includes the signals listed in Table 95.

**Table 95.** SD Memory Card Bus Signals

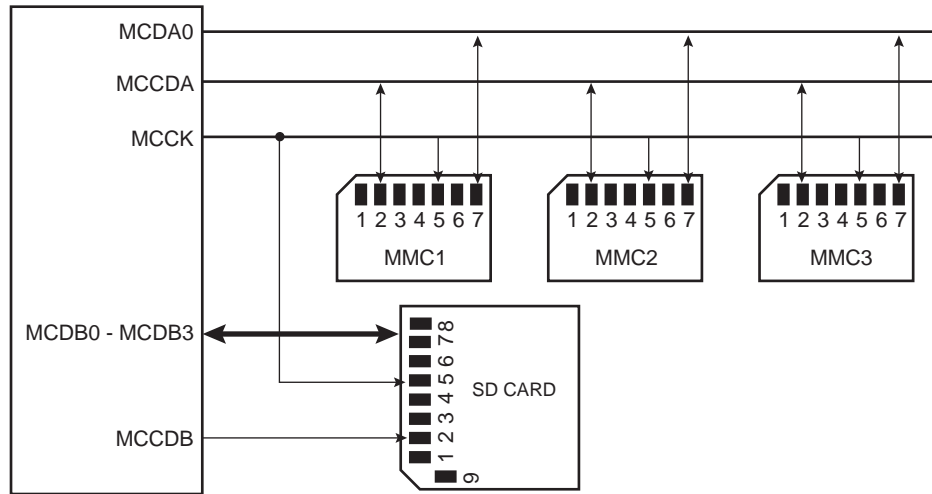
| Pin Number | Name      | Type <sup>(1)</sup> | Description                  | MCI Pin Name |
|------------|-----------|---------------------|------------------------------|--------------|
| 1          | CD/DAT[3] | I/O/PP              | Card detect/ Data line Bit 3 | MCDA3/MCDB3  |
| 2          | CMD       | PP                  | Command/response             | MCCDA/MCCDB  |
| 3          | VSS1      | S                   | Supply voltage ground        | VSS          |
| 4          | VDD       | S                   | Supply voltage               | VDD          |
| 5          | CLK       | I                   | Clock                        | MCKK         |
| 6          | VSS2      | S                   | Supply voltage ground        | VSS          |
| 7          | DAT[0]    | I/O/PP              | Data line Bit 0              | MCDA0/MCDB0  |
| 8          | DAT[1]    | I/O/PP              | Data line Bit 1              | MCDA1/MCDB1  |
| 9          | DAT[2]    | I/O/PP              | Data line Bit 2              | MCDA2/MCDB2  |

Note: 1. I: input, O: output, PP: Push Pull, OD: Open Drain

**Figure 238.** SD Card Bus Connections



**Figure 239.** Mixing MultiMedia and SD Memory Cards



When the MCI is configured to operate with SD memory cards, the width of the data bus can be selected in the MCI\_SDCR register. Clearing the SDCBUS bit in this register means that the width is one bit and setting it means that the width is four bits. In the case of multimedia cards, only the data line 0 is used. The other data lines can be used as independent PIOs.

## MultiMedia Card Operations

After a power-on reset, the cards are initialized by a special message-based MultiMedia Card bus protocol. Each message is represented by one of the following tokens:

- **Command:** A command is a token that starts an operation. A command is sent from the host either to a single card (addressed command) or to all connected cards (broadcast command). A command is transferred serially on the CMD line.
- **Response:** A response is a token which is sent from an addressed card or (synchronously) from all connected cards to the host as an answer to a previously received command. A response is transferred serially on the CMD line.
- **Data:** Data can be transferred from the card to the host or vice versa. Data is transferred via the data line.

Card addressing is implemented using a session address assigned during the initialization phase by the bus controller to all currently connected cards. Their unique CID number identifies individual cards.

The structure of commands, responses and data blocks is described in the MultiMedia-Card System Specification Version 2.2. See also Table 96 on page 507.

MultiMediaCard bus data transfers are composed of these tokens.

There are different types of operations. Addressed operations always contain a command and a response token. In addition, some operations have a data token; the others transfer their information directly within the command or response structure. In this case, no data token is present in an operation. The bits on the DAT and the CMD lines are transferred synchronous to the clock MCKK.

Two types of data transfer commands are defined:

- **Sequential commands:** These commands initiate a continuous data stream. They are terminated only when a stop command follows on the CMD line. This mode reduces the command overhead to an absolute minimum.
- **Block-oriented commands:** These commands send a data block succeeded by CRC bits.



## Command-response Operation

Both read and write operations allow either single or multiple block transmission. A multiple block transmission is terminated when a stop command follows on the CMD line similarly to the sequential read.

The MCI provides a set of registers to perform the entire range of MultiMediaCard operations.

After reset the MCI is disabled and becomes valid after setting the MCIEN bit in the MCI\_CR Control Register. The bit PWSEN allows saving power by dividing the MCI clock by 2 power PWSDIV (MCI\_MR) when the bus is inactive.

The command and the response of the card are clocked out with the rising edge of the MCK.

All the timings for MultiMediaCard are defined in the MultiMediaCard System Specification Version 2.2.

The two bus modes (open drain and push/pull) needed to process all the operations are defined in the MCI command register. The MCI\_CMDR allows a command to be carried out.

For example, to perform an ALL\_SEND\_CID command:

| CMD | Host Command |   |         |     | N <sub>ID</sub> Cycles |   |       |   |   | CID or OCR |         |   |   |   |
|-----|--------------|---|---------|-----|------------------------|---|-------|---|---|------------|---------|---|---|---|
|     | S            | T | Content | CRC | E                      | Z | ***** | Z | S | T          | Content | Z | Z | Z |
|     |              |   |         |     |                        |   |       |   |   |            |         |   |   |   |

The command ALL\_SEND\_CID and the fields and values for the MCI\_CMDR Control Register are described in Table 96 and Table 97.

**Table 96.** ALL\_SEND\_CID command description

| CMD Index | Type | Argument          | Resp | Abbreviation | Command Description                                      |
|-----------|------|-------------------|------|--------------|--|
| CMD2      | bcr  | [31:0] stuff bits | R2   | ALL_SEND_CID | Asks all cards to send their CID numbers on the CMD line |

**Table 97.** Fields and Values for MCI\_CMDR Command Register

| Field  | Value                                  |
|--|--|
| CMDNB (command number)                       | 2 (CMD2)                               |
| RSPTYP (response type)                       | 2 (R2: 136 bits response)              |
| SPCMD (special command)                      | 0 (not a special command)              |
| OPCMD (open drain command)                   | 1                                      |
| MAXLAT (max latency for command to response) | 0 (NID cycles ==> 5 cycles)            |
| TRCMD (transfer command)                     | 0 (No transfer)                        |
| TRDIR (transfer direction)                   | X (available only in transfer command) |
| TRTYP (transfer type)                        | X (available only in transfer command) |

The MCI\_ARGR contains the argument field of the command.

To send a command, the user must perform the following steps:

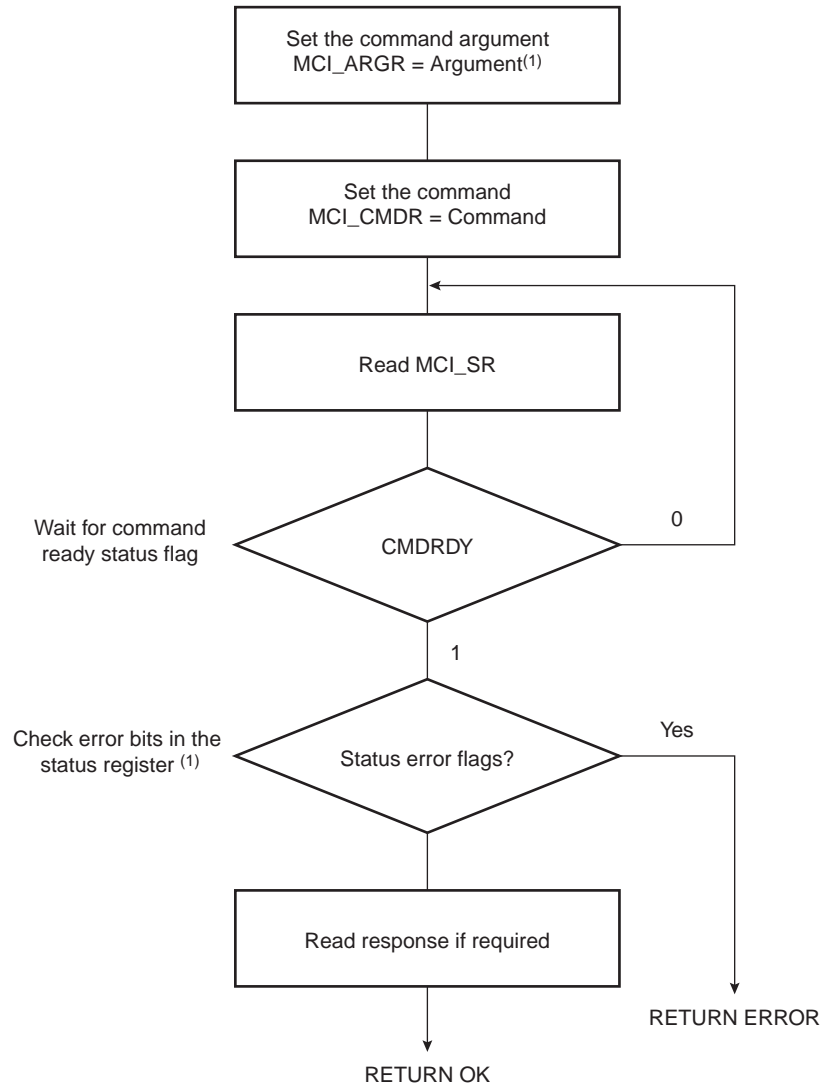
- Fill the argument register (MCI\_ARGR) with the command argument.
- Set the command register (MCI\_CMDR) (see Table 97).

The command is sent immediately after writing the command register. The status bit CMDRDY in the status register (MCI\_SR) is asserted until the command is completed. If the

command requires a response, it can be read in the MCI response register (MCI\_RSPR). The response size can be 48 bits up to 136 bits according to the command. The MCI embeds an error detection to prevent any corrupted data during the transfer.

The following flowchart shows how to send a command to the card and read the response if needed. In this example, the status register bits are polled but setting the appropriate bits in the interrupt enable register (MCI\_IER) allows using an interrupt method.

**Figure 240.** Command/Response Functional Flow Diagram



Note: 1. If the command is SEND\_OP\_COND, the CRC error flag is always present (refer to R3 response in the MultiMediaCard specification).

## Data Transfer Operation

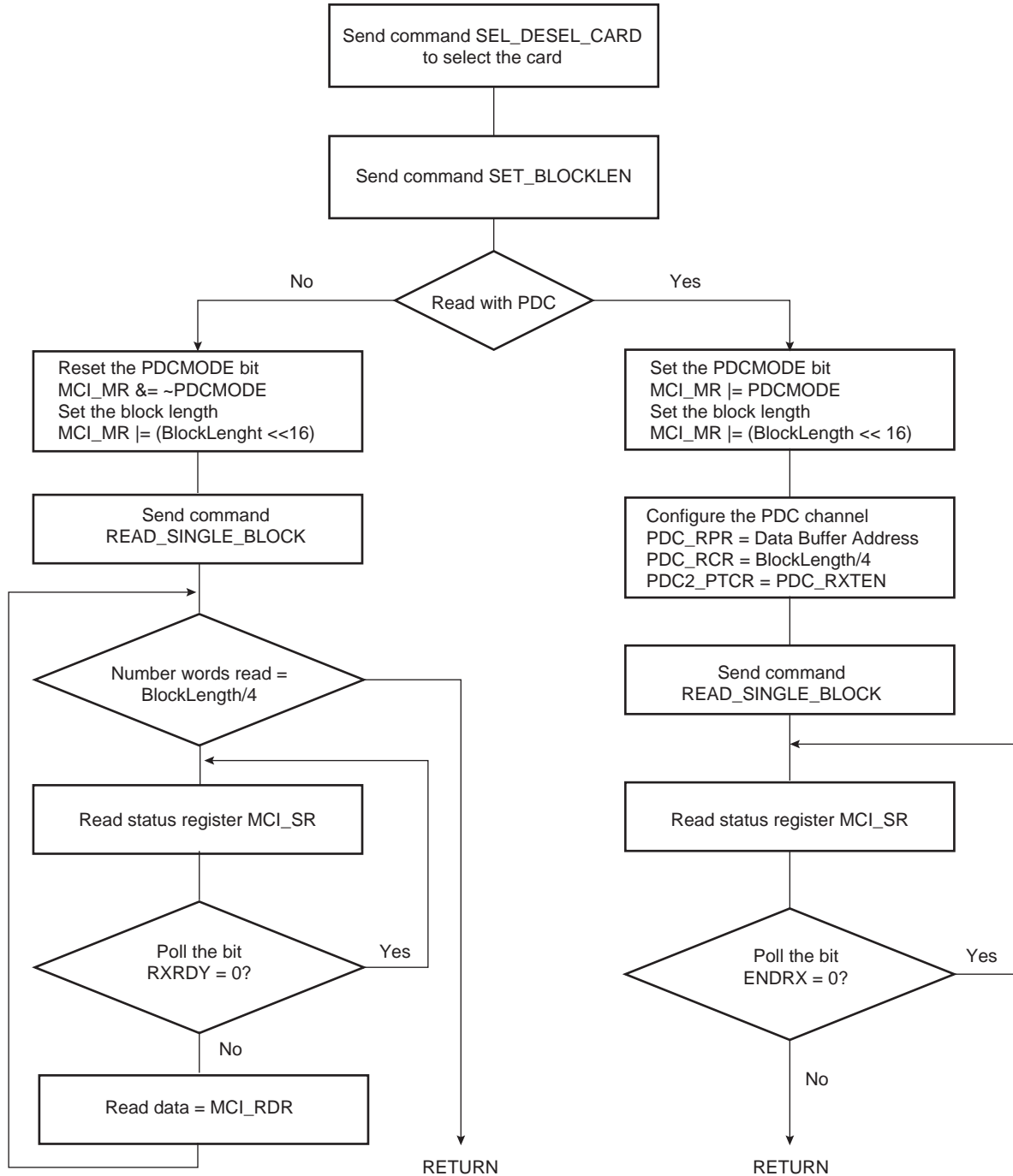
The MultiMedia Card allows several read/write operations (single block, multiple blocks, stream, etc.).

These operations can be done using the Peripheral Data Controller (PDC) features. If the PDCMODE bit is set in MCI\_MR, then all reads and writes use the PDC facilities. In all cases, the block length must be defined in the mode register.

**Read Operation**

The following flowchart shows how to read a single block with or without use of PDC facilities. In this example, a polling method is used to wait for the end of read. Similarly, the user can configure the interrupt enable register (MCI\_IER) to trigger an interrupt at the end of read. These two methods can be applied for all MultiMediaCard read functions.

**Figure 241.** Read Functional Flow Diagram



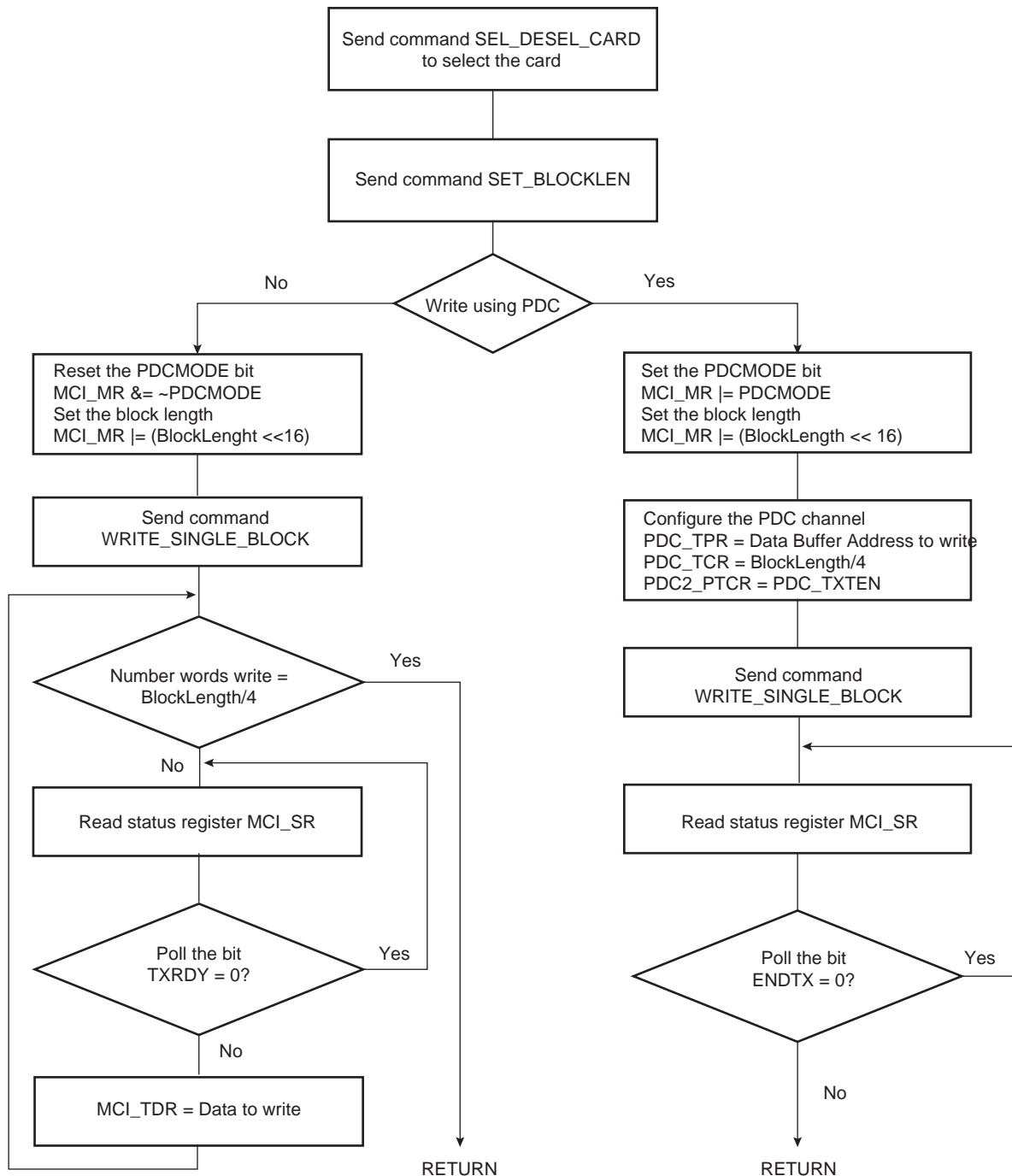
## Write Operation

In write operation the MCI Mode Register (MCI\_MR) is used to define the padding value when writing non-multiple block size. If the bit PDCPADV is 0, then 0x00 value is used when padding data, otherwise 0xFF is used. If set, the bit PDCMODE enables PDC transfer.

The following flowchart shows how to write a single block with or without use of PDC facilities. Polling or interrupt method can be used to wait for the end of write according to the contents of the Interrupt Mask Register (MCI\_IMR).

This flowchart can be adapted to perform all the MultiMedia Card write functions.

**Figure 242.** Write Functional Flow Diagram



## SD Card Operations

The MultiMedia Card Interface allows processing of SD Memory Card (Secure Digital Memory Card) commands. The SD Memory Card will include a copyright protection mechanism that complies with the security requirements of the SDMI standard, is faster and applicable to higher memory capacity.

The physical form factor, pin assignment and data transfer protocol are forward-com-patible with the MultiMedia Card with some additions.

The SD Memory Card communication is based on a 9-pin interface (Clock, Command, 4 x Data and 3 x Power lines). The communication protocol is defined as a part of this specification. The main difference between the SD Memory Card and the MultiMedia Card is the initialization process.

The SD Card Control Register (MCI\_SDCR) allows selection of the card slot and the data bus width.

The SD Card bus allows dynamic configuration of the number of data lines. After power up, by default, the SD Memory Card will use only DAT0 for data transfer. After initialization, the host can change the bus width (number of active data lines).

## MultiMedia Card (MCI) User Interface

**Table 98.** MCI Register Mapping

| Offset      | Register                         | Register Name | Read/Write | Reset  |
|-------------|----------------------------------|---------------|------------|--------|
| 0x00        | Control Register                 | MCI_CR        | Write      | ---    |
| 0x04        | Mode Register                    | MCI_MR        | Read/write | 0x0    |
| 0x08        | Data Timeout Register            | MCI_DTOR      | Read/write | 0x0    |
| 0x0C        | SD Card Register                 | MCI_SDCR      | Read/write | 0x0    |
| 0x10        | Argument Register                | MCI_ARGR      | Read/write | 0x0    |
| 0x14        | Command Register                 | MCI_CMDR      | Write      | ---    |
| 0x18 - 0x1C | Reserved                         |               |            |        |
| 0x20        | Response Register <sup>(1)</sup> | MCI_RSPR      | Read       | 0x0    |
| 0x24        | Response Register <sup>(1)</sup> | MCI_RSPR      | Read       | 0x0    |
| 0x28        | Response Register <sup>(1)</sup> | MCI_RSPR      | Read       | 0x0    |
| 0x2C        | Response Register <sup>(1)</sup> | MCI_RSPR      | Read       | 0x0    |
| 0x30        | Receive Data Register            | MCI_RDR       | Read       | 0x0    |
| 0x34        | Transmit Data Register           | MCI_TDR       | Write      | ---    |
| 0x38 - 0x3C | Reserved                         |               |            |        |
| 0x40        | Status Register                  | MCI_SR        | Read       | 0xC0E5 |
| 0x44        | Interrupt Enable Register        | MCI_IER       | Write      | ---    |
| 0x48        | Interrupt Disable Register       | MCI_IDR       | Write      | ---    |
| 0x4C        | Interrupt Mask Register          | MCI_IMR       | Read       | 0x0    |
| 0x50-0xFF   | Reserved                         |               |            |        |
| 0x100-0x124 | Reserved for the PDC             |               |            |        |

Note: 1. The response register can be read by N accesses at the same MCI\_RSPR or at consecutive addresses (0x20 to 0x2C). N depends on the size of the response.

**MCI Control Register**

Register name: MCI\_CR

Access Type: Write-only

|    |    |    |    |        |       |        |       |
|----|----|----|----|--------|-------|--------|-------|
| 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     |
| –  | –  | –  | –  | PWSDIS | PWSEN | MCIDIS | MCIEN |

- **MCIEN: Multi-Media Interface Enable**

0 = No effect.

1 = Enables the Multi-Media Interface if MCIDIS is 0.

- **MCIDIS: Multi-Media Interface Disable**

0 = No effect.

1 = Disables the Multi-Media Interface.

- **PWSEN: Power Save Mode Enable**

0 = No effect.

1 = Enables the Power Saving Mode if PWSDIS is 0.

- **PWSDIS: Power Save Mode Disable**

0 = No effect.

1 = Disables the Power Saving Mode.

## MCI Mode Register

**Name:** MCI\_MR

**Access Type:** Read/write

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

- **CLKDIV: Clock Divider**

Multi-Media Card Interface clock (MCCK) is Master Clock (MCK) divided by  $(2*(CLKDIV+1))$ .

- **PWSDIV: Power Saving Divider**

Multimedia Card Interface clock is divided by  $2^{(PWSDIV)}$  when entering Power Saving Mode.

- **PDCPADV: PDC Padding Value**

0 = 0x00 value is used when padding data in write transfer (not only PDC transfer).

1 = 0xFF value is used when padding data in write transfer (not only PDC transfer).

- **PDCMODE: PDC-oriented Mode**

0 = Disables PDC transfer

1 = Enables PDC transfer. In this case, UNRE and OVRE (MCI\_SR) are deactivated.

- **BLKLEN: Data Block Length**

This field determines the size of the data block.

Bits 16 and 17 must be 0.



### MCI Data Timeout Register

Name: MCI\_DTOR

Access Type: Read/write

|    |        |    |    |        |    |    |    |
|----|--------|----|----|--------|----|----|----|
| 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  |
| –  | DTOMUL |    |    | DTOCYC |    |    |    |

- **DTOCYC: Data Timeout Cycle Number**
- **DTOMUL: Data Timeout Multiplier**

These fields determine the maximum number of Master Clock cycles that the MCI waits between two data block transfers. It equals (DTCYC x Multiplier).

Multiplier is defined by DTOMUL as shown in the following table:

| DTOMUL |   |   | Multiplier |
|--------|---|---|------------|
| 0      | 0 | 0 | 1          |
| 0      | 0 | 1 | 16         |
| 0      | 1 | 0 | 128        |
| 0      | 1 | 1 | 256        |
| 1      | 0 | 0 | 1024       |
| 1      | 0 | 1 | 4096       |
| 1      | 1 | 0 | 65536      |
| 1      | 1 | 1 | 1048576    |

## MCI SD Card Register

**Name:** MCI\_SDCR

**Access Type:** Read/write

|        |    |    |    |        |    |    |    |
|--------|----|----|----|--------|----|----|----|
| 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  |
| SDCBUS | –  | –  | –  | SDCSEL |    |    |    |

- **SDCSEL: SD Card Selector**

0 = SD card A selected.

1 = SD card B selected.

- **SDCBUS**

0 = 1-bit data bus

1 = 4-bit data bus

## MCI Argument Register

**Name:** MCI\_ARGR

**Access Type:** Read/write

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

- **ARG: Command Argument**

## MCI Command Register

**Name:** MCI\_CMDR

**Access Type:** Write-only

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

This register is write-protected while CMDRDY is 0 in MCI\_SR and in the case of a no Interrupt command sent (bit SPCMD). This means that the current command execution cannot be interrupted or modified.

- **CMDNB: Command Number**
- **RSPTYP: Response Type**

| RSP |   | Response Type     |
|-----|---|-------------------|
| 0   | 0 | No response.      |
| 0   | 1 | 48-bit response.  |
| 1   | 0 | 136-bit response. |
| 1   | 1 | Reserved.         |

- **SPCMD: Special CMD**

| SPCMD |   |   | CMD  |
|-------|---|---|--|
| 0     | 0 | 0 | Not a special CMD.   |
| 0     | 0 | 1 | Initialization CMD:<br>74 clock cycles for initialization sequence.  |
| 0     | 1 | 0 | Synchronized CMD:<br>Wait for the end of the current data block transfer before sending the pending command. |
| 0     | 1 | 1 | Reserved.  |
| 1     | 0 | 0 | Interrupt command:<br>Corresponds to the Interrupt Mode (CMD40).   |
| 1     | 0 | 1 | Interrupt response:<br>Corresponds to the Interrupt Mode (CMD40).  |

- **OPDCMD: Open Drain Command**

0 = Push pull command

1 = Open drain command

- **MAXLAT: Max Latency for Command to Response**

0 = 5-cycle max latency  
 1 = 64-cycle max latency

- **TRCMD: Transfer Command**

| TRCMD |   | Transfer Type   |
|-------|---|-----------------|
| 0     | 0 | No transfer.    |
| 0     | 1 | Start Transfer. |
| 1     | 0 | Stop Transfer.  |
| 1     | 1 | Reserved.       |

- **TRDIR: Transfer Direction**

0 = Write  
 1 = Read

- **TRTYP: Transfer Type**

| TRTYP |   | Transfer Type   |
|-------|---|-----------------|
| 0     | 0 | Block.          |
| 0     | 1 | Multiple Block. |
| 1     | 0 | Stream.         |
| 1     | 1 | Reserved.       |

### MCI SD Response Register

**Name:** MCI\_RSPR

**Access Type:** Read-only

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

- **RSP: Response**

**MCI SD Receive Data Register**

Name: MCI\_RDR

Access Type: Read-only

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

- DATA: Data to Read

**MCI SD Transmit Data Register**

Name: MCI\_TDR

Access Type: Write-only

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

- DATA: Data to Write

## MCI Status Register

**Name:** MCI\_SR

**Access Type:** Read-only

|        |          |         |      |       |        |       |        |
|--------|----------|---------|------|-------|--------|-------|--------|
| 31     | 30       | 29      | 28   | 27    | 26     | 25    | 24     |
| UNRE   | OVRE     | –       | –    | –     | –      | –     | –      |
| 23     | 22       | 21      | 20   | 19    | 18     | 17    | 16     |
| –      | DTOE     | TCRCE   | RTOE | RENDE | RRCRCE | RDIRE | RINDE  |
| 15     | 14       | 13      | 12   | 11    | 10     | 9     | 8      |
| TXBUFE | RXBUFFER | –       | –    | –     | –      | –     | –      |
| 7      | 6        | 5       | 4    | 3     | 2      | 1     | 0      |
| ENDTX  | ENDRX    | NOTBUSY | DTIP | BLKE  | TXRDY  | RXRDY | CMDRDY |

- **CMDRDY: Command Ready**

0 = A command is in progress.

1 = The last command has been sent. Cleared when writing in the MCI\_CMDR.

- **RXRDY: Receiver Ready**

0 = Data has not yet been received since the last read of MCI\_RDR.

1 = Data has been received since the last read of MCI\_RDR.

- **TXRDY: Transmit Ready**

0 = The last data written in MCI\_TDR has not yet been transferred in the Shift Register.

1 = The last data written in MCI\_TDR has been transferred in the Shift Register.

- **BLKE: Data Block Ended**

0 = A data block transfer is not yet finished.

1 = A data block transfer has ended. Set at the end of the last block in PDCMODE, otherwise at the end of the first block. Cleared when reading the MCI\_SR.

- **DTIP: Data Transfer in Progress**

0 = No data transfer in progress.

1 = The current data transfer is still in progress, including CRC16 calculation. Cleared at the end of the CRC16 calculation.

- **NOTBUSY: Data Not Busy**

0 = The card is not ready for new data transfer.

1 = The card is ready for new data transfer (Data line DAT0 high corresponding to a free data receive buffer in the card).

- **ENDRX: End of RX Buffer**

0 = The Receive Counter Register has not reached 0 since the last write in MCI\_RCR or MCI\_RNCR.

1 = The Receive Counter Register has reached 0 since the last write in MCI\_RCR or MCI\_RNCR.

- **ENDTX: End of TX Buffer**

0 = The Transmit Counter Register has not reached 0 since the last write in MCI\_TCR or MCI\_TNCR.

1 = The Transmit Counter Register has reached 0 since the last write in MCI\_TCR or MCI\_TNCR.

- **RXBUFFER: RX Buffer Full**

0 = MCI\_RCR or MCI\_RNCR has a value other than 0.

1 = Both MCI\_RCR and MCI\_RNCR have a value of 0.

- **TXBUFE: TX Buffer Empty**

0 = MCI\_TCR or MCI\_TNCR has a value other than 0.

1 = Both MCI\_TCR and MCI\_TNCR have a value of 0.

- **RINDE: Response Index Error**

0 = No error.

1 = A mismatch is detected between the command index sent and the response index received. Cleared when writing in the MCI\_CMDR.

- **RDIRE: Response Direction Error**

0 = No error.

1 = The direction bit from card to host in the response has not been detected.

- **RCRCE: Response CRC Error**

0 = No error.

1 = A CRC7 error has been detected in the response. Cleared when writing in the MCI\_CMDR.

- **RENDE: Response End Bit Error**

0 = No error.

1 = The end bit of the response has not been detected. Cleared when writing in the MCI\_CMDR.

- **RTOE: Response Time-out Error**

0 = No error.

1 = The response time-out set by MAXLAT in the MCI\_CMDR has been exceeded. Cleared when writing in the MCI\_CMDR.

- **DCRCE: Data CRC Error**

0 = No error.

1 = A CRC16 error has been detected in the last data block. Cleared when sending a new data transfer command.

- **DTOE: Data Time-out Error**

0 = No error.

1 = The data time-out set by DTOCYC and DTOMUL in MCI\_DTOR has been exceeded. Cleared when writing in the MCI\_CMDR.

- **OVRE: Overrun**

0 = No error.

1 = At least one 8-bit received data has been lost (not read). Cleared when sending a new data transfer command.

- **UNRE: Underrun**

0 = No error.

1 = At least one 8-bit data has been sent without valid information (not written). Cleared when sending a new data transfer command.

## MCI Interrupt Enable Register

Name: MCI\_IER

Access Type: Write-only

|        |          |         |      |       |       |       |        |
|--------|----------|---------|------|-------|-------|-------|--------|
| 31     | 30       | 29      | 28   | 27    | 26    | 25    | 24     |
| UNRE   | OVRE     | –       | –    | –     | –     | –     | –      |
| 23     | 22       | 21      | 20   | 19    | 18    | 17    | 16     |
| –      | DTOE     | TCRCE   | RTOE | RENDE | RRCCE | RDIRE | RINDE  |
| 15     | 14       | 13      | 12   | 11    | 10    | 9     | 8      |
| TXBUFE | RXBUFFER | –       | –    | –     | –     | –     | –      |
| 7      | 6        | 5       | 4    | 3     | 2     | 1     | 0      |
| ENDTX  | ENDRX    | NOTBUSY | DTIP | BLKE  | TXRDY | RXRDY | CMDRDY |

- **CMDRDY: Command Ready Interrupt Enable**
- **RXRDY: Receiver Ready Interrupt Enable**
- **TXRDY: Transmit Ready Interrupt Enable**
- **BLKE: Data Block Ended Interrupt Enable**
- **DTIP: Data Transfer in Progress Interrupt Enable**
- **NOTBUSY: Data Not Busy Interrupt Enable**
- **ENDRX: End of Receive Buffer Interrupt Enable**
- **ENDTX: End of Transmit Buffer Interrupt Enable**
- **RXBUFFER: Receive Buffer Full Interrupt Enable**
- **TXBUFE: Transmit Buffer Empty Interrupt Enable**
- **RINDE: Response Index Error Interrupt Enable**
- **RDIRE: Response Direction Error Interrupt Enable**
- **RRCCE: Response CRC Error Interrupt Enable**
- **RENDE: Response End Bit Error Interrupt Enable**
- **RTOE: Response Time-out Error Interrupt Enable**
- **DCRCE: Data CRC Error Interrupt Enable**
- **DTOE: Data Time-out Error Interrupt Enable**
- **OVRE: Overrun Interrupt Enable**
- **UNRE: UnderRun Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.



**MCI Interrupt Disable Register**

**Name:** MCI\_IDR

**Access Type:** Write-only

|        |        |         |      |       |       |       |        |
|--------|--------|---------|------|-------|-------|-------|--------|
| 31     | 30     | 29      | 28   | 27    | 26    | 25    | 24     |
| UNRE   | OVRE   | –       | –    | –     | –     | –     | –      |
| 23     | 22     | 21      | 20   | 19    | 18    | 17    | 16     |
| –      | DTOE   | TCRCE   | RTOE | RENDE | RCRCE | RDIRE | RINDE  |
| 15     | 14     | 13      | 12   | 11    | 10    | 9     | 8      |
| TXBUFE | RXBUFF | –       | –    | –     | –     | –     | –      |
| 7      | 6      | 5       | 4    | 3     | 2     | 1     | 0      |
| ENDTX  | ENDRX  | NOTBUSY | DTIP | BLKE  | TXRDY | RXRDY | CMDRDY |

- **CMDRDY: Command Ready Interrupt Disable**
- **RXRDY: Receiver Ready Interrupt Disable**
- **TXRDY: Transmit Ready Interrupt Disable**
- **BLKE: Data Block Ended Interrupt Disable**
- **DTIP: Data Transfer in Progress Interrupt Disable**
- **NOTBUSY: Data Not Busy Interrupt Disable**
- **ENDRX: End of Receive Buffer Interrupt Disable**
- **ENDTX: End of Transmit Buffer Interrupt Disable**
- **RXBUFF: Receive Buffer Full Interrupt Disable**
- **TXBUFE: Transmit Buffer Empty Interrupt Disable**
- **RINDE: Response Index Error Interrupt Disable**
- **RDIRE: Response Direction Error Interrupt Disable**
- **RCRCE: Response CRC Error Interrupt Disable**
- **RENDE: Response End Bit Error Interrupt Disable**
- **RTOE: Response Time-out Error Interrupt Disable**
- **DCRCE: Data CRC Error Interrupt Disable**
- **DTOE: Data Time-out Error Interrupt Disable**
- **OVRE: Overrun Interrupt Disable**
- **UNRE: UnderRun Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

## MCI Interrupt Mask Register

Name: MCI\_IMR

Access Type: Read-only

|        |        |         |      |       |       |       |        |
|--------|--------|---------|------|-------|-------|-------|--------|
| 31     | 30     | 29      | 28   | 27    | 26    | 25    | 24     |
| UNRE   | OVRE   | –       | –    | –     | –     | –     | –      |
| 23     | 22     | 21      | 20   | 19    | 18    | 17    | 16     |
| –      | DTOE   | TCRCE   | RTOE | RENDE | RRCCE | RDIRE | RINDE  |
| 15     | 14     | 13      | 12   | 11    | 10    | 9     | 8      |
| TXBUFE | RXBUFF | –       | –    | –     | –     | –     | –      |
| 7      | 6      | 5       | 4    | 3     | 2     | 1     | 0      |
| ENDTX  | ENDRX  | NOTBUSY | DTIP | BLKE  | TXRDY | RXRDY | CMDRDY |

- **CMDRDY: Command Ready Interrupt Mask**
- **RXRDY: Receiver Ready Interrupt Mask**
- **TXRDY: Transmit Ready Interrupt Mask**
- **BLKE: Data Block Ended Interrupt Mask**
- **DTIP: Data Transfer in Progress Interrupt Mask**
- **NOTBUSY: Data Not Busy Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask**
- **ENDTX: End of Transmit Buffer Interrupt Mask**
- **RXBUFF: Receive Buffer Full Interrupt Mask**
- **TXBUFE: Transmit Buffer Empty Interrupt Mask**
- **RINDE: Response Index Error Interrupt Mask**
- **RDIRE: Response Direction Error Interrupt Mask**
- **RRCCE: Response CRC Error Interrupt Mask**
- **RENDE: Response End Bit Error Interrupt Mask**
- **RTOE: Response Time-out Error Interrupt Mask**
- **DCRCE: Data CRC Error Interrupt Mask**
- **DTOE: Data Time-out Error Interrupt Mask**
- **OVRE: Overrun Interrupt Mask**
- **UNRE: UnderRun Interrupt Mask**

0 = The corresponding interrupt is not enabled.

1 = The corresponding interrupt is enabled.

## USB Device Port (UDP)

### Overview

The USB Device Port (UDP) is compliant with the Universal Serial Bus (USB) V2.0 full-speed device specification. It is designed to be associated with Atmel's embedded USB transceiver and interfaced with an ARM7TDMI and ARM9TDMI core.

The number and size of endpoints is product-dependent. Each endpoint is associated with one or two banks of a dual-port RAM used to store the current data payload. If two banks are used, one DPR bank is read or written by the processor, while the other is read or written by the USB device peripheral. This feature is mandatory for isochronous endpoints. Thus the device maintains the maximum bandwidth (1M bytes/s) by working with endpoints with two banks of DPR.

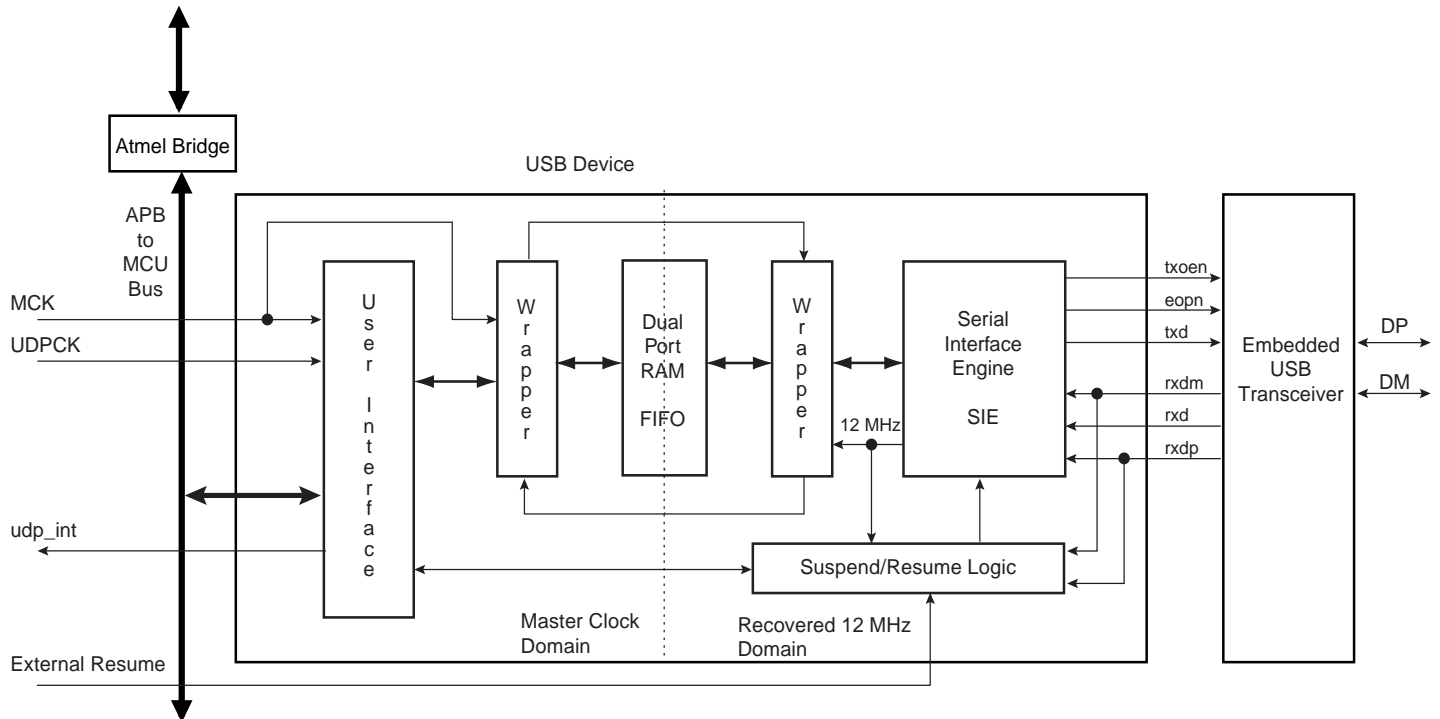
Suspend and resume are automatically detected by the USB device, which notifies the processor by raising an interrupt. Depending on the product, an external signal can be used to send a wake-up to the USB host controller.

The main features of the UDP are:

- USB V2.0 Full-speed Compliant, 12 Mbits per second
- Embedded USB V2.0 Full-speed Transceiver
- Number and Size of Endpoints Fully Parametrizable in RTL
- Embedded Dual-port RAM for Endpoints
- Suspend/Resume Logic
- Ping-pong Mode (2 Memory Banks) for Isochronous and Bulk Endpoints

## Block Diagram

Figure 243. USB Device Port Block Diagram



Access to the UDP is via the APB bus interface. Read and write to the data FIFO are done by reading and writing 8-bit values to APB registers.

The UDP peripheral requires two clocks: one peripheral clock used by the MCK domain and a 48 MHz clock used by the 12 MHz domain.

A USB 2.0 full-speed pad is embedded and controlled by the SIE.

The signal `external_resume` is optional. It allows the UDP peripheral to wake-up once in system mode. The host will then be notified that the device asks for a resume. This optional feature must be also negotiated with the host during the enumeration.

## Product Dependencies

The USB physical transceiver is integrated into the product. The bi-directional differential signals DP and DM are available from the product boundary.

Two I/O lines may be used by the application:

- One to check that VBUS is still available from the host. Self-powered devices may use this entry to be notified that the host has been powered off. In this case, the board pull-up on DP must be disabled in order to prevent feeding current to the host.
- One to control the board pull-up on DP. Thus, when the device is ready to communicate with the host, it activates its DP pull-up through this control line.

## I/O Lines

DP and DM are not controlled by any PIO controllers. The embedded USB physical transceiver is controlled by the USB device peripheral.

To reserve an I/O line to check VBUS, the programmer must first program the PIO controller to assign this I/O in input PIO mode.

To reserve an I/O line to control the board pull-up, the programmer must first program the PIO controller to assign this I/O in output PIO mode.

## Power Management

The USB device peripheral requires a 48 MHz clock. This clock must be generated by a PLL with an accuracy of  $\pm 0.25\%$ .

Thus, the USB device receives two clocks from the Power Management Controller (PMC): the master clock, MCK, used to drive the peripheral user interface and the UDPCK used to interface with the bus USB signals (recovered 12 MHz domain).

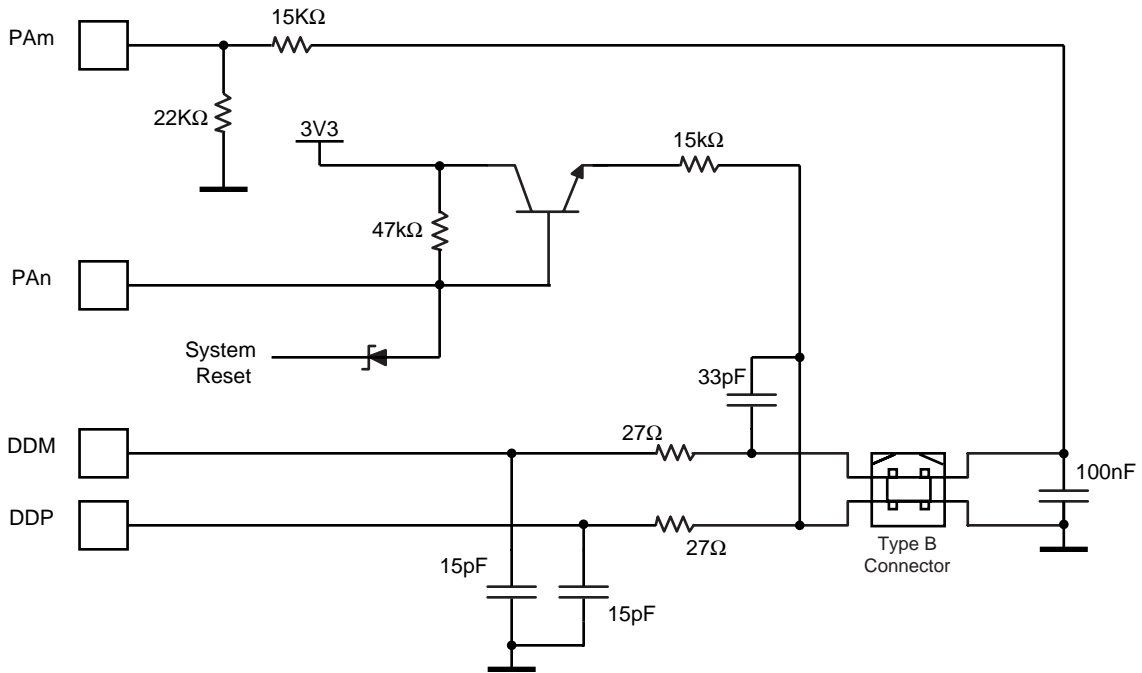
## Interrupt

The USB device interface has an interrupt line connected to the Advanced Interrupt Controller (AIC).

Handling the USB device interrupt requires programming the AIC before configuring the UDP.

## Typical Connection

**Figure 244.** Board Schematic to Interface USB Device Peripheral



USB\_CNx is an input signal used to check if the host is connected

USB\_DP\_PUP is an output signal used to enable pull-up on DP.

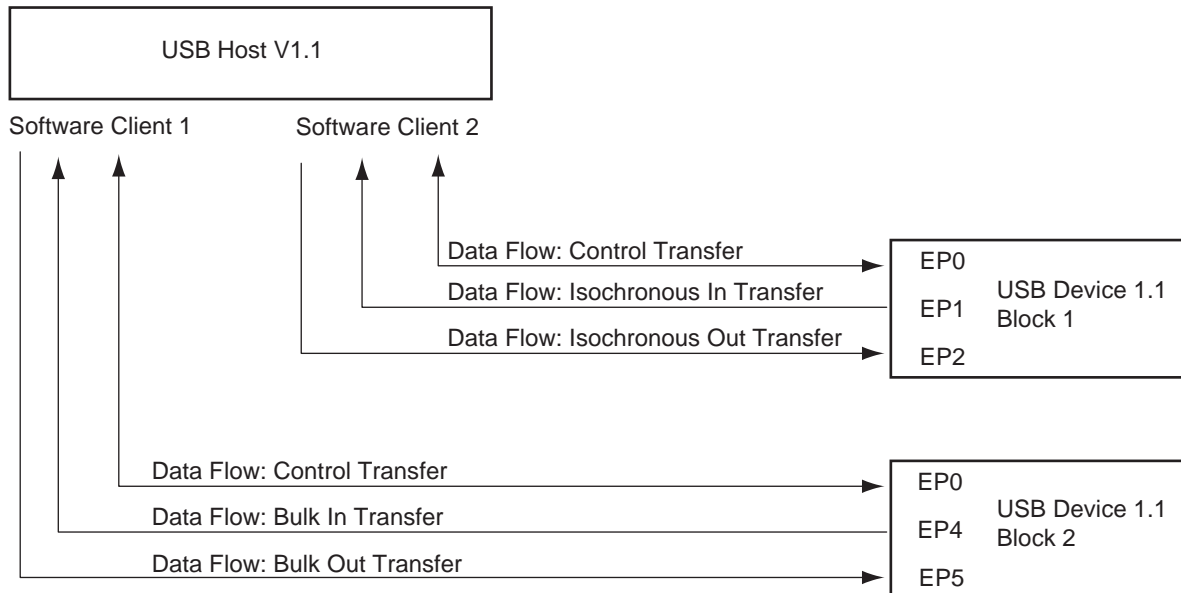
Figure 244 shows automatic activation of pull-up after reset.

## Functional Description

### USB V2.0 Full-speed Introduction

The USB V2.0 full-speed provides communication services between host and attached USB devices. Each device is offered with a collection of communication flows (pipes) associated with each endpoint. Software on the host communicates with an USB device through a set of communication flows.

**Figure 245.** Example of USB V2.0 Full-speed Communication Control



### USB V2.0 Full-speed Transfer Types

A communication flow is carried over one of four transfer types defined by the USB device.

**Table 99.** USB Communication Flow

| Transfer    | Direction       | Bandwidth      | Endpoint Size | Error Detection | Retrying  |
|-------------|-----------------|----------------|---------------|-----------------|-----------|
| Control     | Bi-directional  | Not guaranteed | 8, 16, 32, 64 | Yes             | Automatic |
| Isochronous | Uni-directional | Guaranteed     | 1 - 1023      | Yes             | No        |
| Interrupt   | Uni-directional | Not guaranteed | ≤ 64          | Yes             | Yes       |
| Bulk        | Uni-directional | Not guaranteed | 8, 16, 32, 64 | Yes             | Yes       |

## USB Bus Transactions

Each transfer results in one or more transactions over the USB bus. There are five kinds of transactions flowing across the bus in packets:

1. Setup Transaction
2. Data IN Transaction
3. Data OUT Transaction
4. Status IN Transaction
5. Status OUT Transaction

## USB Transfer Event Definitions

As shown in Table 100, transfers are sequential events carried out on the USB bus.

**Table 100.** USB Transfer Events

|   |  |
|---|--|
| Control Transfers <sup>(1) (3)</sup>                            | <ul style="list-style-type: none"> <li>• Setup transaction &gt; Data IN transactions &gt; Status OUT transaction</li> <li>• Setup transaction &gt; Data OUT transactions &gt; Status IN transaction</li> <li>• Setup transaction &gt; Status IN transaction</li> </ul> |
| Interrupt IN Transfer<br>(device toward host)                   | <ul style="list-style-type: none"> <li>• Data IN transaction &gt; Data IN transaction</li> </ul>   |
| Interrupt OUT Transfer<br>(host toward device)                  | <ul style="list-style-type: none"> <li>• Data OUT transaction &gt; Data OUT transaction</li> </ul>   |
| Isochronous IN Transfer <sup>(2)</sup><br>(device toward host)  | <ul style="list-style-type: none"> <li>• Data IN transaction &gt; Data IN transaction</li> </ul>   |
| Isochronous OUT Transfer <sup>(2)</sup><br>(host toward device) | <ul style="list-style-type: none"> <li>• Data OUT transaction &gt; Data OUT transaction</li> </ul>   |
| Bulk IN Transfer<br>(device toward host)                        | <ul style="list-style-type: none"> <li>• Data IN transaction &gt; Data IN transaction</li> </ul>   |
| Bulk OUT Transfer<br>(host toward device)                       | <ul style="list-style-type: none"> <li>• Data OUT transaction &gt; Data OUT transaction</li> </ul>   |

- Notes:
1. Control transfer must use endpoints with no ping-pong attributes.
  2. Isochronous transfers must use endpoints with ping-pong attributes.
  3. Control transfers can be aborted using a stall handshake.



## Handling Transactions with USB V2.0 Device Peripheral

### Setup Transaction

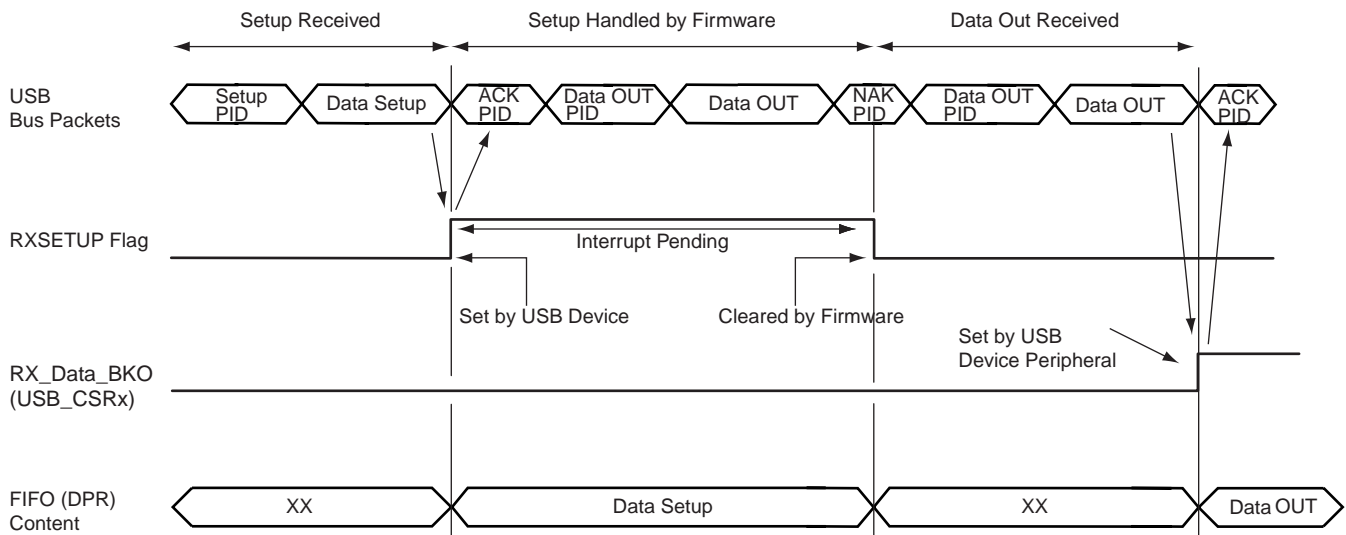
Setup is a special type of host-to-device transaction used during control transfers. Control transfers must be performed using endpoints with no ping-pong attributes. A setup transaction needs to be handled as soon as possible by the firmware. It is used to transmit requests from the host to the device. These requests are then handled by the USB device and may require more arguments. The arguments are sent to the device by a Data OUT transaction which follows the setup transaction. These requests may also return data. The data is carried out to the host by the next Data IN transaction which follows the setup transaction. A status transaction ends the control transfer.

When a setup transfer is received by the USB endpoint:

- The USB device automatically acknowledges the setup packet
- RXSETUP is set in the USB\_CSRx register
- An endpoint interrupt is generated while the RXSETUP is not cleared. This interrupt is carried out to the microcontroller if interrupts are enabled for this endpoint.

Thus, firmware must detect the RXSETUP polling the USB\_CSRx or catching an interrupt, read the setup packet in the FIFO, then clear the RXSETUP. RXSETUP cannot be cleared before the setup packet has been read in the FIFO. Otherwise, the USB device would accept the next Data OUT transfer and overwrite the setup packet in the FIFO.

**Figure 246.** Setup Transaction Followed by a Data OUT Transaction



## Data IN Transaction

Data IN transactions are used in control, isochronous, bulk and interrupt transfers and conduct the transfer of data from the device to the host. Data IN transactions in isochronous transfer must be done using endpoints with ping-pong attributes.

### Using Endpoints Without Ping-pong Attributes

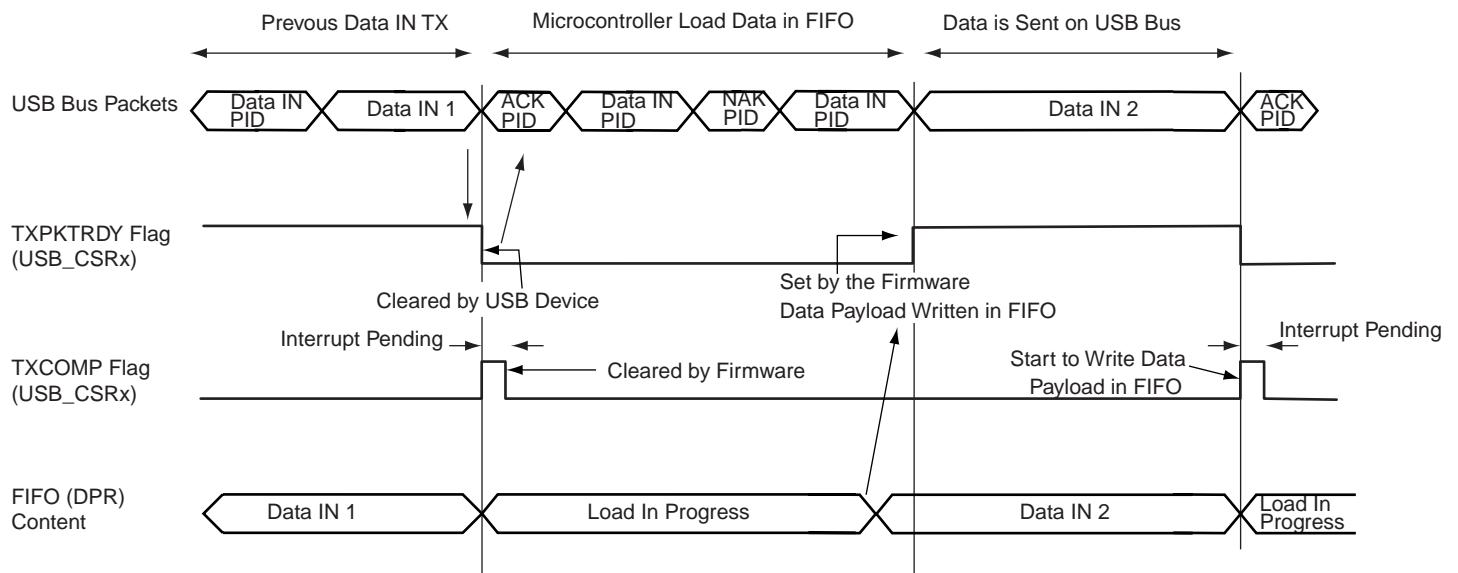
To perform a Data IN transaction, using a non ping-pong endpoint:

1. The microcontroller checks if it is possible to write in the FIFO by polling TXPKTRDY in the endpoint's USB\_CSRx register (TXPKTRDY must be cleared).
2. The microcontroller writes data to be sent in the endpoint's FIFO, writing zero or more byte values in the endpoint's USB\_FDRx register,
3. The microcontroller notifies the USB peripheral it has finished by setting the TXPKTRDY in the endpoint's USB\_CSRx register,
4. The microcontroller is notified that the endpoint's FIFO has been released by the USB device when TXCOMP in the endpoint's USB\_CSRx register has been set. Then an interrupt for the corresponding endpoint is pending while TXCOMP is set.

TXCOMP is set by the USB device when it has received an ACK PID signal for the Data IN packet. An interrupt is pending while TXCOMP is set.

Note: Please refer to Chapter 8 of the *Universal Serial Bus Specification, Rev 1.1*, for more information on the Data IN protocol layer.

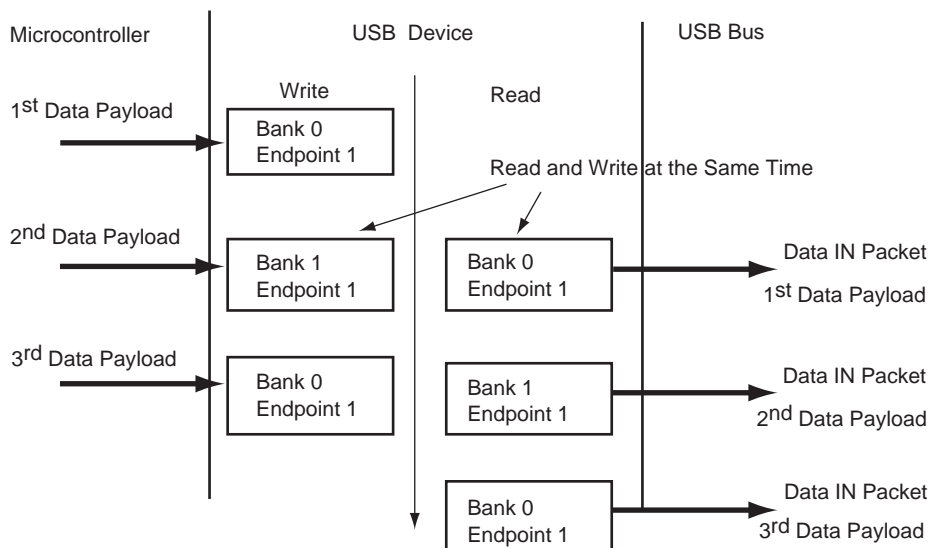
**Figure 247.** Data IN Transfer for Non Ping-pong Endpoint



## Using Endpoints With Ping-pong Attribute

The use of an endpoint with ping-pong attributes is necessary during isochronous transfer. To be able to guarantee a constant bandwidth, the microcontroller must prepare the next data payload to be sent while the current one is being sent by the USB device. Thus two banks of memory are used. While one is available for the microcontroller, the other one is locked by the USB device.

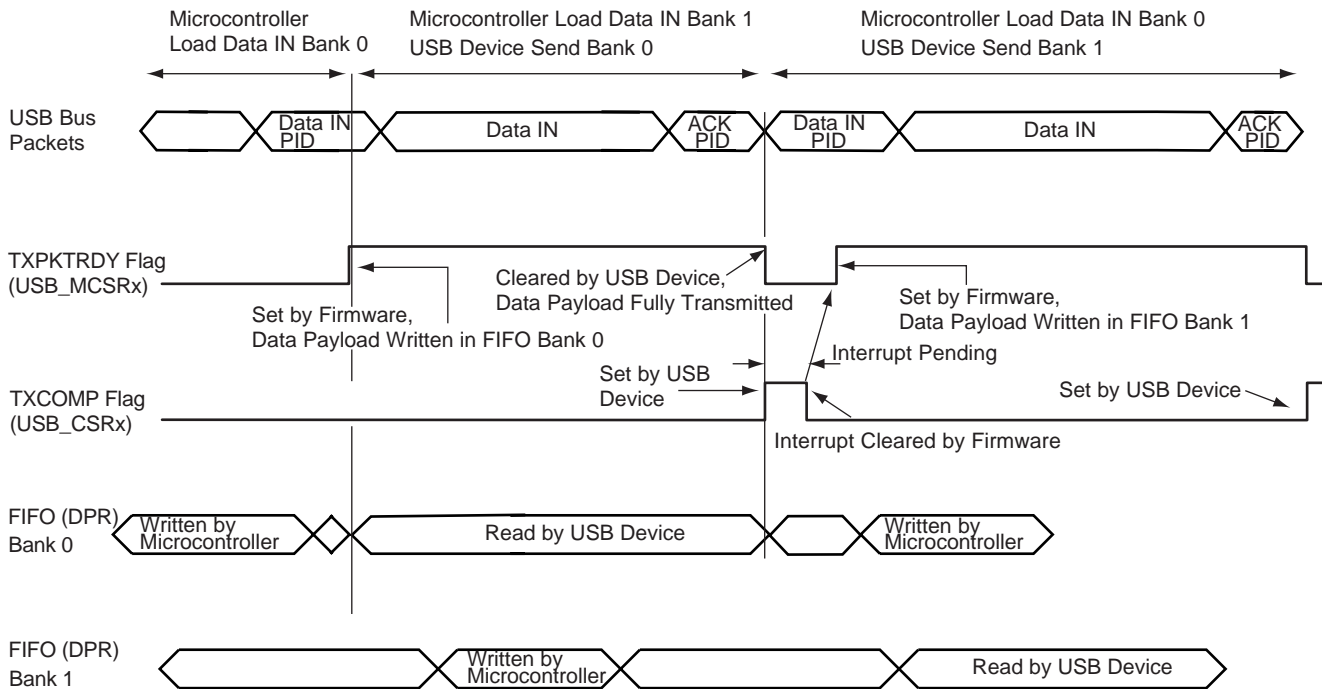
**Figure 248.** Bank Swapping Data IN Transfer for Ping-pong Endpoints



When using a ping-pong endpoint, the following procedures are required to perform Data IN transactions:

1. The microcontroller checks if it is possible to write in the FIFO by polling TXPKTRDY to be cleared in the endpoint's USB\_CSRx register.
2. The microcontroller writes the first data payload to be sent in the FIFO (Bank 0), writing zero or more byte values in the endpoint's USB\_FDRx register.
3. The microcontroller notifies the USB peripheral it has finished writing in Bank 0 of the FIFO by setting the TXPKTRDY in the endpoint's USB\_CSRx register.
4. Without waiting for TXPKTRDY to be cleared, the microcontroller writes the second data payload to be sent in the FIFO (Bank 1), writing zero or more byte values in the endpoint's USB\_FDRx register.
5. The microcontroller is notified that the first Bank has been released by the USB device when TXCOMP in the endpoint's USB\_CSRx register is set. An interrupt is pending while TXCOMP is being set.
6. Once the microcontroller has received TXCOMP for the first Bank, it notifies the USB device that it has prepared the second Bank to be sent rising TXPKTRDY in the endpoint's USB\_CSRx register.
7. At this step, Bank 0 is available and the microcontroller can prepare a third data payload to be sent.

**Figure 249.** Data IN Transfer for Ping-pong Endpoint



**Warning:** There is software critical path due to the fact that once the second bank is filled, the driver has to wait for TX\_COMP to set TX\_PKTRDY. If the delay between receiving TX\_COMP is set and TX\_PKTRDY is set is too long, some Data IN packets may be NACKed, reducing the bandwidth.

## Data OUT Transaction

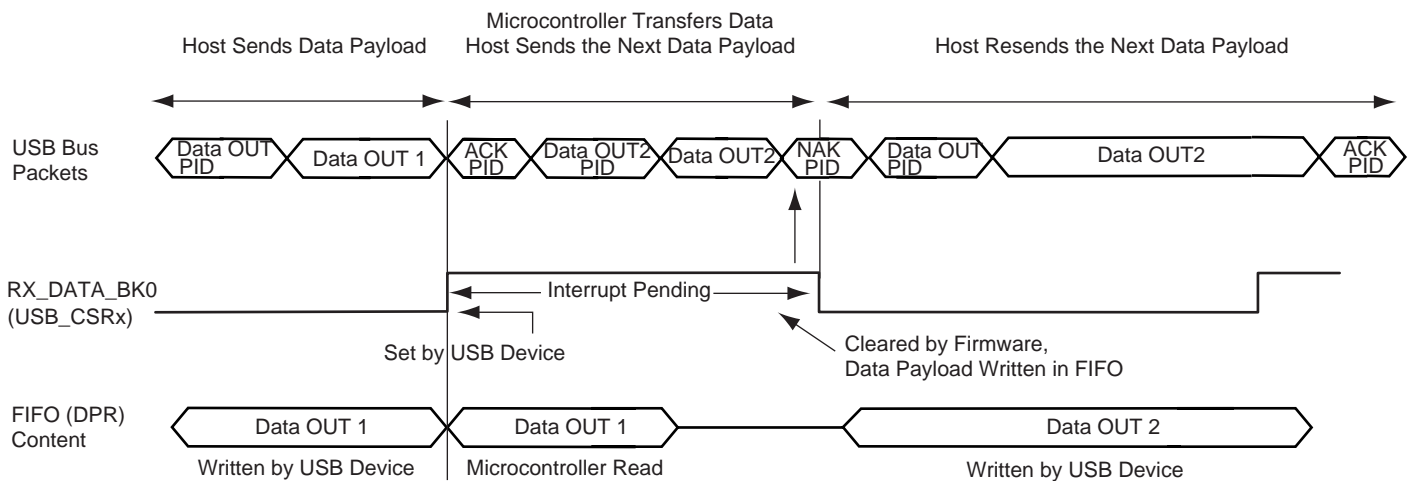
Data OUT transactions are used in control, isochronous, bulk and interrupt transfers and conduct the transfer of data from the host to the device. Data OUT transactions in isochronous transfers must be done using endpoints with ping-pong attributes.

### Data OUT Transaction Without Ping-pong Attributes

To perform a Data OUT transaction, using a non ping-pong endpoint:

1. The host generates a Data OUT packet.
2. This packet is received by the USB device endpoint. While the FIFO associated to this endpoint is being used by the microcontroller, a NAK PID is returned to the host. Once the FIFO is available, data are written to the FIFO by the USB device and an ACK is automatically carried out to the host.
3. The microcontroller is notified that the USB device has received a data payload polling RX\_DATA\_BK0 in the endpoint's USB\_CSRx register. An interrupt is pending for this endpoint while RX\_DATA\_BK0 is set.
4. The number of bytes available in the FIFO is made available by reading RXBYTECNT in the endpoint's USB\_CSRx register.
5. The microcontroller carries out data received from the endpoint's memory to its memory. Data received is available by reading the endpoint's USB\_FDRx register.
6. The microcontroller notifies the USB device that it has finished the transfer by clearing RX\_DATA\_BK0 in the endpoint's USB\_CSRx register.
7. A new Data OUT packet can be accepted by the USB device.

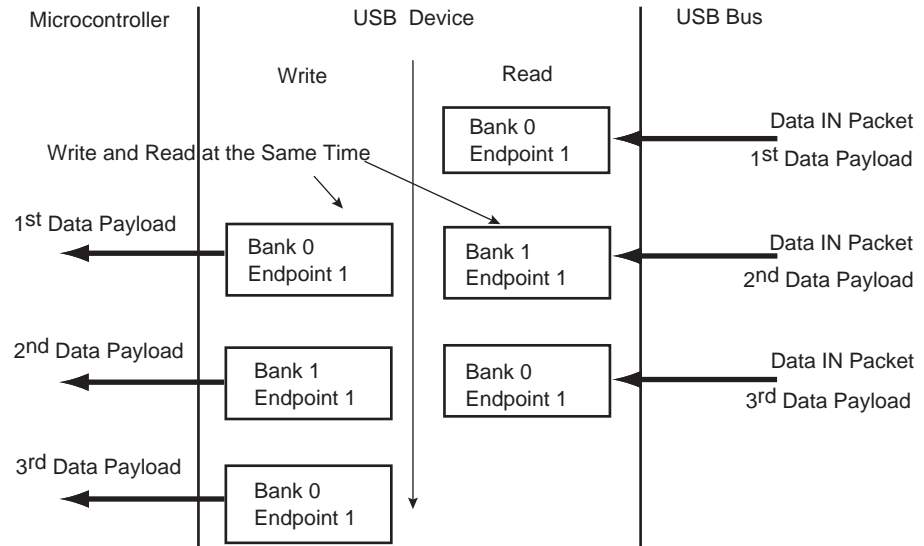
**Figure 250.** Data OUT Transfer for Non Ping-pong Endpoints



An interrupt is pending while the flag RX\_DATA\_BK0 is set. Memory transfer between the USB device, the FIFO and microcontroller memory can not be done after RX\_DATA\_BK0 has been cleared. Otherwise, the USB device would accept the next Data OUT transfer and overwrite the current Data OUT packet in the FIFO.

During isochronous transfer, using an endpoint with ping-pong attributes is necessary. To be able to guarantee a constant bandwidth, the microcontroller must read the previous data payload sent by the host, while the current data payload is received by the USB device. Thus two banks of memory are used. While one is available for the microcontroller, the other one is locked by the USB device.

**Figure 251.** Bank Swapping in Data OUT Transfers for Ping-pong Endpoints

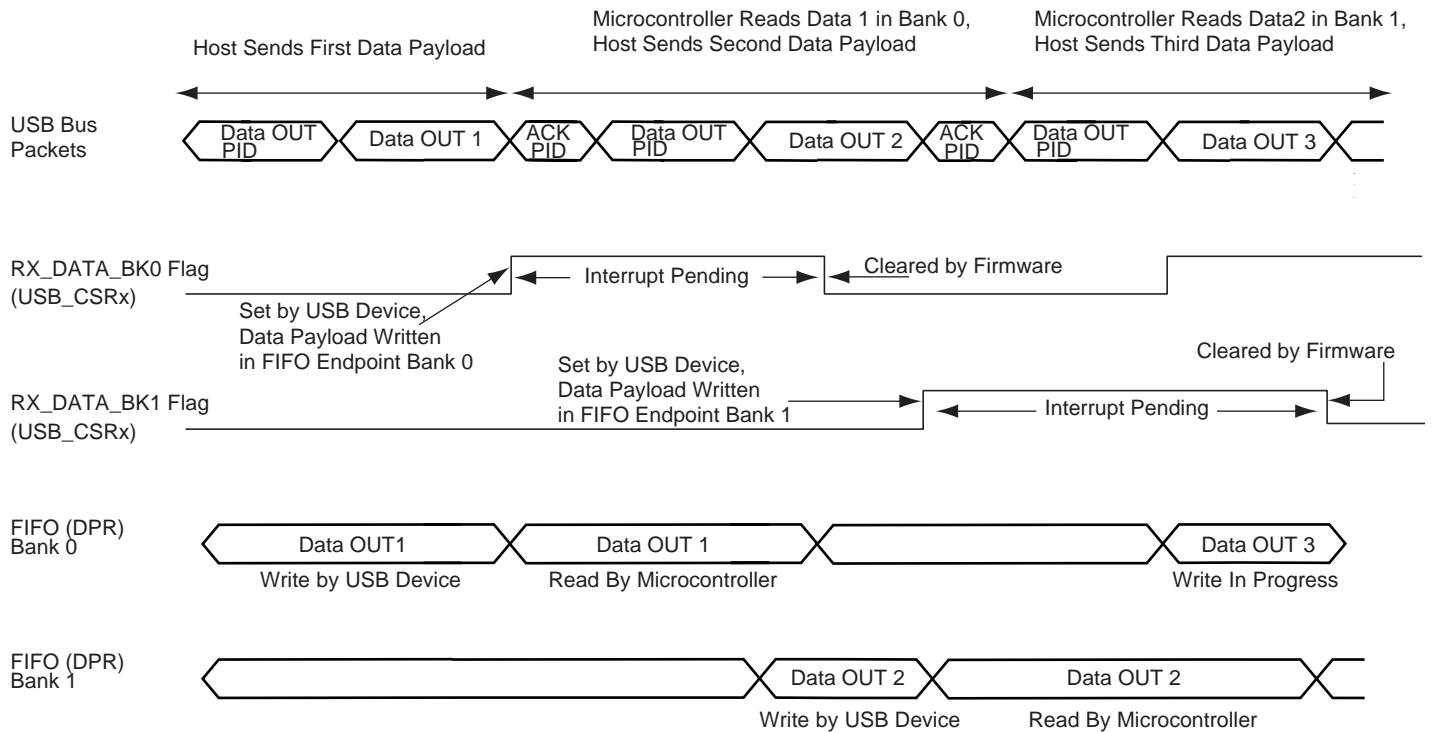


When using a ping-pong endpoint, the following procedures are required to perform Data OUT transactions:

1. The host generates a Data OUT packet.
2. This packet is received by the USB device endpoint. It is written in the endpoint's FIFO Bank 0.
3. The USB device sends an ACK PID packet to the host. The host can immediately send a second Data OUT packet. It is accepted by the device and copied to FIFO Bank 1.
4. The microcontroller is notified that the USB device has received a data payload, polling RX\_DATA\_BK0 in the endpoint's USB\_CSRx register. An interrupt is pending for this endpoint while RX\_DATA\_BK0 is set.
5. The number of bytes available in the FIFO is made available by reading RXBYTECNT in the endpoint's USB\_CSRx register.
6. The microcontroller transfers out data received from the endpoint's memory to the microcontroller's memory. Data received is made available by reading the endpoint's USB\_FDRx register.
7. The microcontroller notifies the USB peripheral device that it has finished the transfer by clearing RX\_DATA\_BK0 in the endpoint's USB\_CSRx register.
8. A third Data OUT packet can be accepted by the USB peripheral device and copied in the FIFO Bank 0.
9. If a second Data OUT packet has been received, the microcontroller is notified by the flag RX\_DATA\_BK1 set in the endpoint's USB\_CSRx register. An interrupt is pending for this endpoint while RX\_DATA\_BK1 is set.

10. The microcontroller transfers out data received from the endpoint's memory to the microcontroller's memory. Data received is available by reading the endpoint's USB\_FDRx register.
11. The microcontroller notifies the USB device it has finished the transfer by clearing RX\_DATA\_BK1 in the endpoint's USB\_CSRx register.
12. A fourth Data OUT packet can be accepted by the USB device and copied in the FIFO Bank 0.

**Figure 252.** Data OUT Transfer for Ping-pong Endpoint



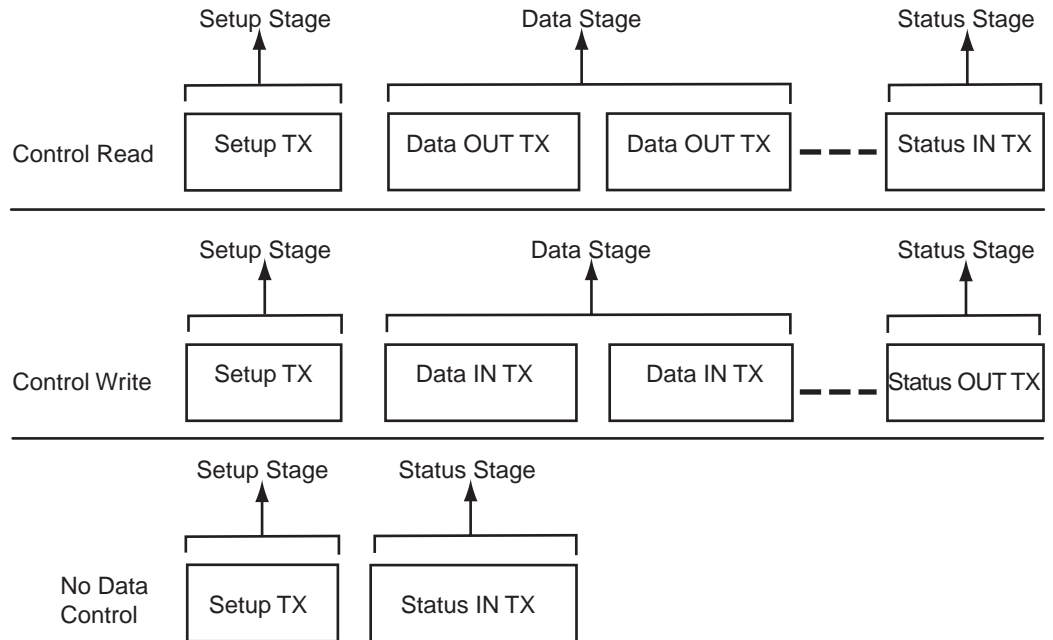
Note: An interrupt is pending while the RX\_DATA\_BK0 or RX\_DATA\_BK1 flag is set.

**Warning:** When RX\_DATA\_BK0 and RX\_DATA\_BK1 are both set, there is no way to determine which one to clear first. Thus the software must keep an internal counter to be sure to clear alternatively RX\_DATA\_BK0 then RX\_DATA\_BK1. This situation may occur when the software application is busy elsewhere and the two banks are filled by the USB host. Once the application comes back to the USB driver, the two flags are set.

## Status Transaction

A status transaction is a special type of host to device transaction used only in a control transfer. The control transfer must be performed using endpoints with no ping-pong attributes. According to the control sequence (read or write), the USB device sends or receives a status transaction.

**Figure 253.** Control Read and Write Sequences



- Notes:
1. During the Status IN stage, the host waits for a zero length packet (Data IN transaction with no data) from the device using DATA1 PID. Please refer to Chapter 8 of the *Universal Serial Bus Specification, Rev. 1.1*, to get more information on the protocol layer.
  2. During the Status OUT stage, the host emits a zero length packet to the device (Data OUT transaction with no data).

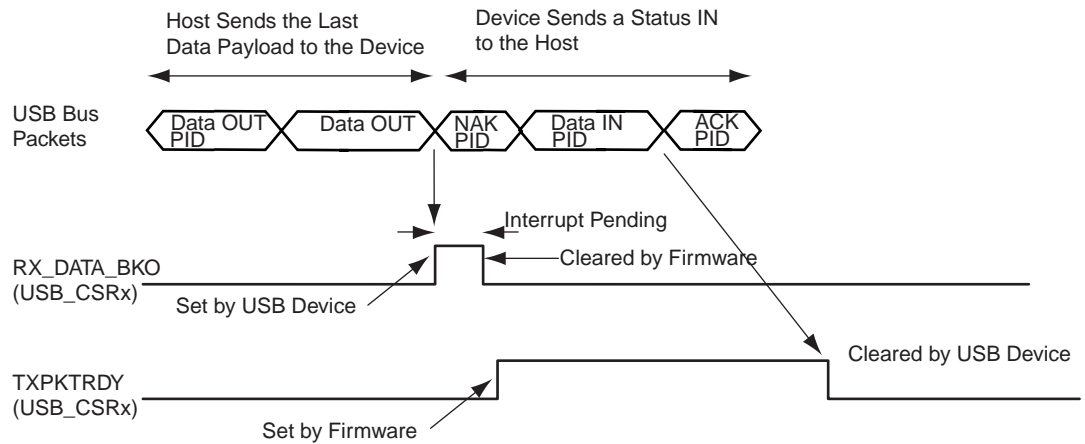


## Status IN Transfer

Once a control request has been processed, the device returns a status to the host. This is a zero length Data IN transaction.

1. The microcontroller waits for TXPKTRDY in the USB\_CSRx endpoint's register to be cleared. (At this step, TXPKTRDY must be cleared because the previous transaction was a setup transaction or a Data OUT transaction.)
2. Without writing anything to the USB\_FDRx endpoint's register, the microcontroller sets TXPKTRDY. The USB device generates a Data IN packet using DATA1 PID.
3. This packet is acknowledged by the host and TXPKTRDY is set in the USB\_CSRx endpoint's register.

**Figure 254.** Data Out Followed by Status IN Transfer.

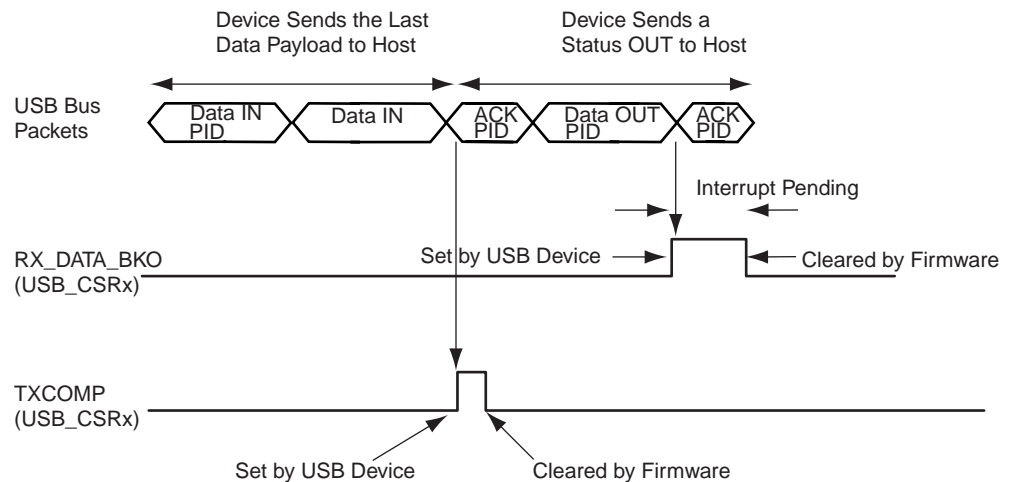


## Status OUT Transfer

Once a control request has been processed and the requested data returned, the host acknowledges by sending a zero length packet. This is a zero length Data OUT transaction.

1. The USB device receives a zero length packet. It sets RX\_DATA\_BK0 flag in the USB\_CSRx register and acknowledges the zero length packet.
2. The microcontroller is notified that the USB device has received a zero length packet sent by the host polling RX\_DATA\_BK0 in the USB\_CSRx register. An interrupt is pending while RX\_DATA\_BK0 is set. The number of bytes received in the endpoint's USB\_BCR register is equal to zero.
3. The microcontroller must clear RX\_DATA\_BK0.

**Figure 255.** Data IN Followed by Status OUT Transfer



**Stall Handshake**

A stall handshake can be used in one of two distinct occasions. (For more information on the stall handshake, refer to Chapter 8 of the *Universal Serial Bus Specification, Rev 1.1*.)

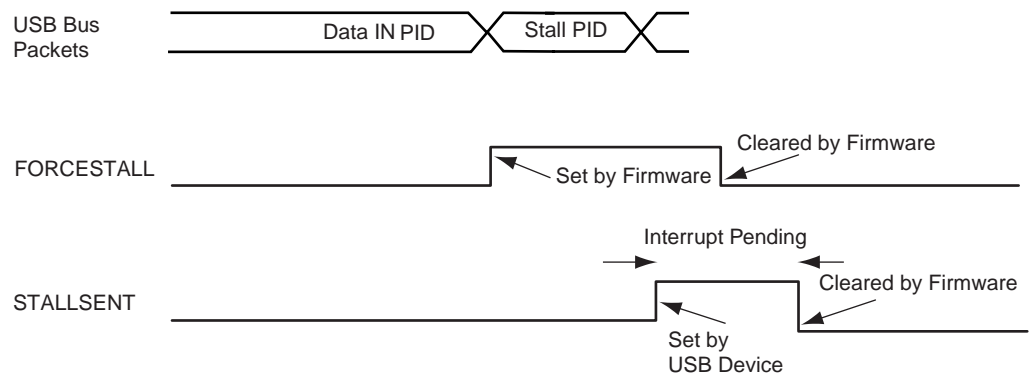
- A functional stall is used when the halt feature associated with the endpoint is set. (Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev 1.1*, for more information on the halt feature.)
- To abort the current request, a protocol stall is used, but uniquely with control transfer.

The following procedure generates a stall packet:

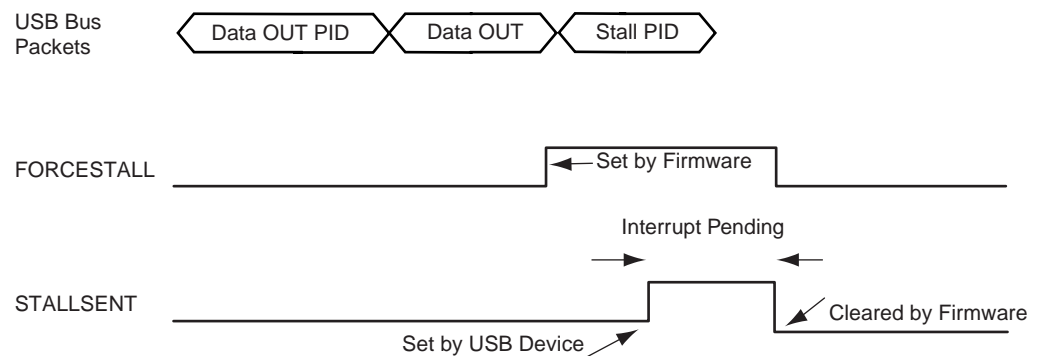
1. The microcontroller sets the FORCESTALL flag in the USB\_CSRx endpoint's register.
2. The host receives the stall packet.
3. The microcontroller is notified that the device has sent the stall by polling the STALLSENT to be set. An endpoint interrupt is pending while STALLSENT is set. The microcontroller must clear STALLSENT to clear the interrupt.

When a setup transaction is received after a stall handshake, STALLSENT must be cleared in order to prevent interrupts due to STALLSENT being set.

**Figure 256.** Stall Handshake (Data IN Transfer)



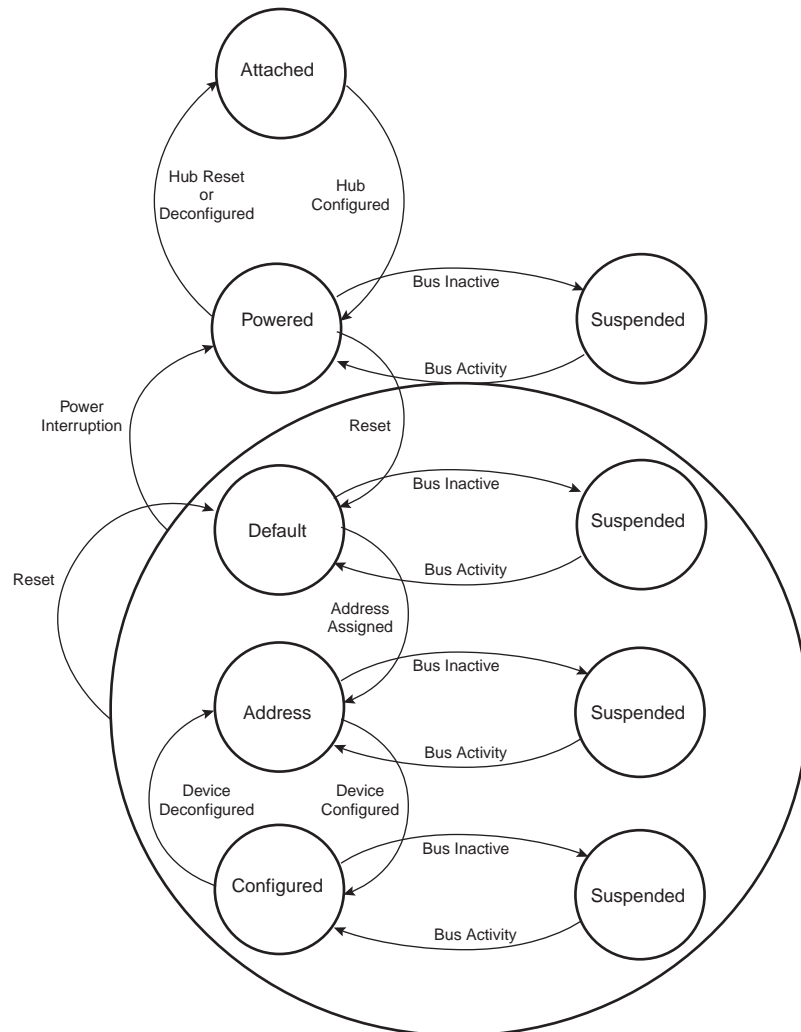
**Figure 257.** Stall Handshake (Data OUT Transfer)



## Controlling Device States

A USB device has several possible states. Please refer to Chapter 9 of the *Universal Serial Bus Specification, Rev 1.1*.

**Figure 258.** USB Device State Diagram



Movement from one state to another depends on the USB bus state or on standard requests sent through control transactions via the default endpoint (endpoint 0).

After a period of bus inactivity, the UDP device enters Suspend Mode. Accepting Suspend/Resume requests from the USB host is mandatory. Constraints in Suspend Mode are very strict for bus-powered applications; devices may not consume more than 500 uA on the USB bus.

While in Suspend Mode, the host may wake up a device by sending a resume signal (bus activity) or a USB device may send a wake-up request to the host, e.g., waking up a PC by moving a USB mouse.

The wake-up feature is not mandatory for all devices and must be negotiated with the host.

## From Powered State to Default State

After its connection to a USB host, the USB device waits for an end-of-bus reset. The USB host stops driving a reset state once it has detected the device's pull-up on DP. The unmasked flag ENDBURST is set in the register UDP\_ISR and an interrupt is triggered. The UDP software enables the default endpoint, setting the EPEDS flag in the UDP\_CSR[0] register and, optionally, enabling the interrupt for endpoint 0 by writing 1 to the UDP\_IER register. The enumeration then begins by a control transfer.

## From Default State to Address State

After a set address standard device request, the USB host peripheral enters the address state. Before this, it achieves the Status IN transaction of the control transfer, i.e., the UDP device sets its new address once the TXCOMP flag in the UDP\_CSR[0] register has been received and cleared.

To move to address state, the driver software sets the FADDEN flag in the UDP\_GLB\_STATE, sets its new address, and sets the FEN bit in the UDP\_FADDR register.

## From Address State to Configured State

Once a valid Set Configuration standard request has been received and acknowledged, the device enables endpoints corresponding to the current configuration. This is done by setting the EPEDS and EPTYPE fields in the UDP\_CSRx registers and, optionally, enabling corresponding interrupts in the UDP\_IER register.

## Enabling Suspend

When a Suspend (no bus activity on the USB bus) is detected, the RXSUSP signal in the UDP\_ISR register is set. This triggers an interrupt if the corresponding bit is set in the UDP\_IMR register.

This flag is cleared by writing to the UDP\_ICR register. Then the device enters Suspend Mode. As an example, the microcontroller switches to slow clock, disables the PLL and main oscillator, and goes into Idle Mode. It may also switch off other devices on the board.

The USB device peripheral clocks may be switched off. However, the transceiver and the USB peripheral must not be switched off, otherwise the resume is not detected.

## Receiving a Host Resume

In suspend mode, the USB transceiver and the USB peripheral must be powered to detect the RESUME. However, the USB device peripheral may not be clocked as the WAKEUP signal is asynchronous.

Once the resume is detected on the bus, the signal WAKEUP in the UDP\_ISR is set. It may generate an interrupt if the corresponding bit in the UDP\_IMR register is set. This interrupt may be used to wake-up the core, enable PLL and main oscillators and configure clocks. The WAKEUP bit must be cleared as soon as possible by setting WAKEUP in the UDP\_ICR register.

## Sending an External Resume

The External Resume is negotiated with the host and enabled by setting the ESR bit in the USB\_GLB\_STATE. An asynchronous event on the ext\_resume\_pin of the peripheral generates a WAKEUP interrupt. On early versions of the USP peripheral, the K-state on the USB line is generated immediately. This means that the USB device must be able to answer to the host very quickly. On recent versions, the software sets the RMWUPE bit in the UDP\_GLB\_STATE register once it is ready to communicate with the host. The K-state on the bus is then generated.

The WAKEUP bit must be cleared as soon as possible by setting WAKEUP in the UDP\_ICR register.

## USB Device Port (UDP) User Interface

**Table 101.** USB Device Port Memory Map

| Offset | Register                               | Name         | Access     | Reset State |
|--------|--|--------------|------------|-------------|
| 0x000  | Frame Number Register                  | USB_FRM_NUM  | Read       | 0x0000_0000 |
| 0x004  | Global State Register                  | USB_GLB_STAT | Read/write | 0x0000_0010 |
| 0x008  | Function Address Register              | USB_FADDR    | Read/write | 0x0000_0100 |
| 0x00C  | Reserved                               | –            | –          | –           |
| 0x010  | Interrupt Enable Register              | USB_IER      | Write      |             |
| 0x014  | Interrupt Disable Register             | USB_IDR      | Write      |             |
| 0x018  | Interrupt Mask Register                | USB_IMR      | Read       | 0x0000_1200 |
| 0x01C  | Interrupt Status Register              | USB_ISR      | Read       | 0x0000_0000 |
| 0x020  | Interrupt Clear Register               | USB_ICR      | Write      |             |
| 0x024  | Reserved                               | –            | –          | –           |
| 0x028  | Reset Endpoint Register                | USB_RST_EP   | Read/write |             |
| 0x02C  | Reserved                               | –            | –          | –           |
| 0x030  | Endpoint 0 Control and Status Register | USB_CSR0     | Read/write | 0x0000_0000 |
| 0x034  | Endpoint 1 Control and Status Register | USB_CSR1     | Read/write | 0x0000_0000 |
| 0x038  | Endpoint 2 Control and Status Register | USB_CSR2     | Read/write | 0x0000_0000 |
| 0x03C  | Endpoint 3 Control and Status Register | USB_CSR3     | Read/write | 0x0000_0000 |
| 0x040  | Endpoint 4 Control and Status Register | USB_CSR4     | Read/write | 0x0000_0000 |
| 0x044  | Endpoint 5 Control and Status Register | USB_CSR5     | Read/write | 0x0000_0000 |
| 0x048  | Endpoint 6 Control and Status Register | USB_CSR6     | Read/write | 0x0000_0000 |
| 0x04C  | Endpoint 7 Control and Status Register | USB_CSR7     | Read/write | 0x0000_0000 |
| 0x050  | Endpoint 0 FIFO Data Register          | USB_FDR0     | Read/write | 0x0000_0000 |
| 0x054  | Endpoint 1 FIFO Data Register          | USB_FDR1     | Read/write | 0x0000_0000 |
| 0x058  | Endpoint 2 FIFO Data Register          | USB_FDR2     | Read/write | 0x0000_0000 |
| 0x05C  | Endpoint 3 FIFO Data Register          | USB_FDR3     | Read/write | 0x0000_0000 |
| 0x060  | Endpoint 4 FIFO Data Register          | USB_FDR4     | Read/write | 0x0000_0000 |
| 0x064  | Endpoint 5 FIFO Data Register          | USB_FDR5     | Read/write | 0x0000_0000 |
| 0x068  | Endpoint 6 FIFO Data Register          | USB_FDR6     | Read/write | 0x0000_0000 |
| 0x06C  | Endpoint 7 FIFO Data Register          | USB_FDR7     | Read/write | 0x0000_0000 |
| 0x070  | Reserved                               | –            | –          | –           |
| 0x074  | Reserved                               | –            | –          | –           |

**USB Frame Number Register**

**Register Name:** USB\_FRM\_NUM

**Access Type:** Read-only

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

• **FRM\_NUM[10:0]: Frame Number as Defined in the Packet Field Formats**

This 11-bit value is incremented by the host on a per frame basis. This value is updated at each start of frame. Value Updated at the SOF\_EOP (Start of Frame End of Packet).

• **FRM\_ERR: Frame Error**

This bit is set at SOF\_EOP when the SOF packet is received containing an error.

This bit is reset upon receipt of SOF\_PID.

• **FRM\_OK: Frame OK**

This bit is set at SOF\_EOP when the SOF packet is received without any error.

This bit is reset upon receipt of SOF\_PID (Packet Identification).

In the Interrupt Status Register, the SOF interrupt is updated upon receiving SOF\_PID. This bit is set without waiting for EOP.

Note: In the 8-bit Register Interface, FRM\_OK is bit 4 of FRM\_NUM\_H and FRM\_ERR is bit 3 of FRM\_NUM\_L.

## USB Global State Register

**Register Name:** USB\_GLB\_STAT

**Access Type:** Read/Write

|    |    |    |        |         |     |        |        |
|----|----|----|--------|---------|-----|--------|--------|
| 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      |
| –  | –  | –  | RMWUPE | RSMINPR | ESR | CONFIG | FADDEN |

This register is used to get and set the device state as specified in Chapter 9 of the *USB Serial Bus Specification, Rev. 1.1*.

- **FADDEN: Function Address Enable**

Read:

0 = Device is not in address state.

1 = Device is in address state.

Write:

0 = No effect, only a reset can bring back a device to the default state.

1 = Set device in address state. This occurs after a successful Set Address request. Beforehand, the USB\_FADDR register must have been initialized with Set Address parameters. Set Address must complete the Status Stage before setting FADDEN. Please refer to chapter 9 of the *Universal Serial Bus Specification, Rev. 1.1* to get more details.

- **CONFIG: Configured**

Read:

0 = Device is not in configured state.

1 = Device is in configured state.

Write:

0 = Set device in a nonconfigured state

1 = Set device in configured state.

The device is set in configured state when it is in address state and receives a successful Set Configuration request. Please refer to Chapter 9 of the *Universal Serial Bus Specification, Rev. 1.1* to get more details.

- **ESR: Enable Send Resume**

0 = Disable the Remote Wake Up sequence.

1 = Remote Wake Up can be processed and the pin send\_resume is enabled.

- **RSMINPR: A Resume Has Been Sent to the Host**

Read:

0 = No effect.

1 = A Resume has been received from the host during Remote Wake Up feature.

- **RMWUPE: Remote Wake Up Enable**

0 = Must be cleared after receiving any HOST packet or SOF interrupt.

1 = Enables the K-state on the USB cable if ESR is enabled.



**USB Function Address Register**

**Register Name:** USB\_FADDR

**Access Type:** Read/Write

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

• **FADD[6:0]: Function Address Value**

The Function Address Value must be programmed by firmware once the device receives a set address request from the host, and has achieved the status stage of the no-data control sequence. Please refer to the *Universal Serial Bus Specification, Rev. 1.1* to get more information. After power up, or reset, the function address value is set to 0.

• **FEN: Function Enable**

Read:

0 = Function endpoint disabled.

1 = Function endpoint enabled.

Write:

0 = Disable function endpoint.

1 = Default value.

The Function Enable bit (FEN) allows the microcontroller to enable or disable the function endpoints. The microcontroller sets this bit after receipt of a reset from the host. Once this bit is set, the USB device is able to accept and transfer data packets from and to the host.

## USB Interrupt Enable Register

**Register Name:** USB\_IER

**Access Type:** Write-only

|        |        |        |        |        |        |        |        |
|--------|--------|--------|--------|--------|--------|--------|--------|
| 31     | 30     | 29     | 28     | 27     | 26     | 25     | 24     |
| –      | –      | –      | –      | –      | –      | –      | –      |
| 23     | 22     | 21     | 20     | 19     | 18     | 17     | 16     |
| –      | –      | –      | –      | –      | –      | –      | –      |
| 15     | 14     | 13     | 12     | 11     | 10     | 9      | 8      |
| –      | –      | WAKEUP | –      | SOFINT | EXTRSM | RXRSM  | RXSUSP |
| 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
| EP7INT | EP6INT | EP5INT | EP4INT | EP3INT | EP2INT | EP1INT | EP0INT |

- **EP0INT: Enable Endpoint 0 Interrupt**
- **EP1INT: Enable Endpoint 1 Interrupt**
- **EP2INT: Enable Endpoint 2 Interrupt**
- **EP3INT: Enable Endpoint 3 Interrupt**
- **EP4INT: Enable Endpoint 4 Interrupt**
- **EP5INT: Enable Endpoint 5 Interrupt**
- **EP6INT: Enable Endpoint 6 Interrupt**
- **EP7INT: Enable Endpoint 7 Interrupt**

0 = No effect.

1 = Enable corresponding Endpoint Interrupt.

- **RXSUSP: Enable USB Suspend Interrupt**

0 = No effect.

1 = Enable USB Suspend Interrupt.

- **RXRSM: Enable USB Resume Interrupt**

0 = No effect.

1 = Enable USB Resume Interrupt.

- **EXTRSM: Enable External Resume Interrupt**

0 = No effect.

1 = Enable External Resume Interrupt.

- **SOFINT: Enable Start Of Frame Interrupt**

0 = No effect.

1 = Enable Start Of Frame Interrupt.

- **WAKEUP: Enable USB bus Wakeup Interrupt**

0 = No effect.

1 = Enable USB bus Interrupt.

## USB Interrupt Disable Register

**Register Name:** USB\_IDR

**Access Type:** Write-only

|        |        |        |        |        |        |        |        |
|--------|--------|--------|--------|--------|--------|--------|--------|
| 31     | 30     | 29     | 28     | 27     | 26     | 25     | 24     |
| –      | –      | –      | –      | –      | –      | –      | –      |
| 23     | 22     | 21     | 20     | 19     | 18     | 17     | 16     |
| –      | –      | –      | –      | –      | –      | –      | –      |
| 15     | 14     | 13     | 12     | 11     | 10     | 9      | 8      |
| –      | –      | WAKEUP | –      | SOFINT | EXTRSM | RXRSM  | RXSUSP |
| 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
| EP7INT | EP6INT | EP5INT | EP4INT | EP3INT | EP2INT | EP1INT | EP0INT |

- **EP0INT: Disable Endpoint 0 Interrupt**
- **EP1INT: Disable Endpoint 1 Interrupt**
- **EP2INT: Disable Endpoint 2 Interrupt**
- **EP3INT: Disable Endpoint 3 Interrupt**
- **EP4INT: Disable Endpoint 4 Interrupt**
- **EP5INT: Disable Endpoint 5 Interrupt**
- **EP6INT: Disable Endpoint 6 Interrupt**
- **EP7INT: Disable Endpoint 7 Interrupt**

0 = No effect.

1 = Disable corresponding Endpoint Interrupt.

- **RXSUSP: Disable USB Suspend Interrupt**

0 = No effect.

1 = Disable USB Suspend Interrupt.

- **RXRSM: Disable USB Resume Interrupt**

0 = No effect.

1 = Disable USB Resume Interrupt.

- **EXTRSM: Disable External Resume Interrupt**

0 = No effect.

1 = Disable External Resume Interrupt.

- **SOFINT: Disable Start Of Frame Interrupt**

0 = No effect.

1 = Disable Start Of Frame Interrupt

- **WAKEUP: Disable USB Bus Interrupt**

0 = No effect.

1 = Disable USB Bus Wakeup Interrupt.

## USB Interrupt Mask Register

**Register Name:** USB\_IMR

**Access Type:** Read-only

|        |        |        |        |        |        |        |        |
|--------|--------|--------|--------|--------|--------|--------|--------|
| 31     | 30     | 29     | 28     | 27     | 26     | 25     | 24     |
| –      | –      | –      | –      | –      | –      | –      | –      |
| 23     | 22     | 21     | 20     | 19     | 18     | 17     | 16     |
| –      | –      | –      | –      | –      | –      | –      | –      |
| 15     | 14     | 13     | 12     | 11     | 10     | 9      | 8      |
| –      | –      | WAKEUP | –      | SOFINT | EXTRSM | RXRSM  | RXSUSP |
| 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
| EP7INT | EP6INT | EP5INT | EP4INT | EP3INT | EP2INT | EP1INT | EP0INT |

- **EP0INT: Mask Endpoint 0 Interrupt**
- **EP1INT: Mask Endpoint 1 Interrupt**
- **EP2INT: Mask Endpoint 2 Interrupt**
- **EP3INT: Mask Endpoint 3 Interrupt**
- **EP4INT: Mask Endpoint 4 Interrupt**
- **EP5INT: Mask Endpoint 5 Interrupt**
- **EP6INT: Mask Endpoint 6 Interrupt**
- **EP7INT: Mask Endpoint 7 Interrupt**

0 = Corresponding Endpoint Interrupt is disabled.

1 = Corresponding Endpoint Interrupt is enabled.

- **RXSUSP: Mask USB Suspend Interrupt**

0 = USB Suspend Interrupt is disabled.

1 = USB Suspend Interrupt is enabled.

- **RXRSM: Mask USB Resume Interrupt.**

0 = USB Resume Interrupt is disabled.

1 = USB Resume Interrupt is enabled.

- **EXTRSM: Mask External Resume Interrupt**

0 = External Resume Interrupt is disabled.

1 = External Resume Interrupt is enabled.

- **SOFINT: Mask Start Of Frame Interrupt**

0 = Start of Frame Interrupt is disabled.

1 = Start of Frame Interrupt is enabled.

- **WAKEUP: USB Bus WAKEUP Interrupt**

0 = USB Bus Wakeup Interrupt is disabled.

1 = USB Bus Wakeup Interrupt is enabled.

**Note:** When the USB block is in suspend mode, the application may power down the USB logic. In this case, any USB HOST resume request that is made must be taken into account and, thus, the reset value of the RXRSM bit of the register USB\_IMR is enabled.

## USB Interrupt Status Register

**Register Name:** USB\_ISR

**Access Type:** Read-only

|        |        |        |           |        |        |        |        |
|--------|--------|--------|-----------|--------|--------|--------|--------|
| 31     | 30     | 29     | 28        | 27     | 26     | 25     | 24     |
| –      | –      | –      | –         | –      | –      | –      | –      |
| 23     | 22     | 21     | 20        | 19     | 18     | 17     | 16     |
| –      | –      | –      | –         | –      | –      | –      | –      |
| 15     | 14     | 13     | 12        | 11     | 10     | 9      | 8      |
| –      | –      | WAKEUP | ENDBUSRES | SOFINT | EXTRSM | RXRSM  | RXSUSP |
| 7      | 6      | 5      | 4         | 3      | 2      | 1      | 0      |
| EP7INT | EP6INT | EP5INT | EP4INT    | EP3INT | EP2INT | EP1INT | EP0INT |

### • EP0INT: Endpoint 0 Interrupt Status

0 = No Endpoint0 Interrupt pending.

1 = Endpoint0 Interrupt has been raised.

Several signals can generate this interrupt. The reason can be found by reading USB\_CSR0:

RXSETUP set to 1

RX\_DATA\_BK0 set to 1

RX\_DATA\_BK1 set to 1

TXCOMP set to 1

STALLSENT set to 1

EP0INT is a sticky bit. Interrupt remains valid until EP0INT is cleared by writing in the corresponding USB\_CSR0 bit.

### • EP1INT: Endpoint 1 Interrupt Status

0 = No Endpoint1 Interrupt pending.

1 = Endpoint1 Interrupt has been raised.

Several signals can generate this interrupt. The reason can be found by reading USB\_CSR1:

RXSETUP set to 1

RX\_DATA\_BK0 set to 1

RX\_DATA\_BK1 set to 1

TXCOMP set to 1

STALLSENT set to 1

EP1INT is a sticky bit. Interrupt remains valid until EP1INT is cleared by writing in the corresponding USB\_CSR1 bit.

### • EP2INT: Endpoint 2 Interrupt Status

0 = No Endpoint2 Interrupt pending.

1 = Endpoint2 Interrupt has been raised.

Several signals can generate this interrupt. The reason can be found by reading USB\_CSR2:

RXSETUP set to 1

RX\_DATA\_BK0 set to 1

RX\_DATA\_BK1 set to 1

TXCOMP set to 1

STALLSENT set to 1

EP2INT is a sticky bit. Interrupt remains valid until EP2INT is cleared by writing in the corresponding USB\_CSR2 bit.

- **EP3INT: Endpoint 3 Interrupt Status**

0 = No Endpoint3 Interrupt pending.

1 = Endpoint3 Interrupt has been raised.

Several signals can generate this interrupt. The reason can be found by reading USB\_CSR3:

- RXSETUP set to 1

- RX\_DATA\_BK0 set to 1

- RX\_DATA\_BK1 set to 1

- TXCOMP set to 1

- STALLSENT set to 1

EP3INT is a sticky bit. Interrupt remains valid until EP3INT is cleared by writing in the corresponding USB\_CSR3 bit.

- **EP4INT: Endpoint 4 Interrupt Status**

0 = No Endpoint4 Interrupt pending.

1 = Endpoint4 Interrupt has been raised.

Several signals can generate this interrupt. The reason can be found by reading USB\_CSR4:

- RXSETUP set to 1

- RX\_DATA\_BK0 set to 1

- RX\_DATA\_BK1 set to 1

- TXCOMP set to 1

- STALLSENT set to 1

EP4INT is a sticky bit. Interrupt remains valid until EP4INT is cleared by writing in the corresponding USB\_CSR4 bit.

- **EP5INT: Endpoint 5 Interrupt Status**

0 = No Endpoint5 Interrupt pending.

1 = Endpoint5 Interrupt has been raised.

Several signals can generate this interrupt. The reason can be found by reading USB\_CSR5:

- RXSETUP set to 1

- RX\_DATA\_BK0 set to 1

- RX\_DATA\_BK1 set to 1

- TXCOMP set to 1

- STALLSENT set to 1

EP5INT is a sticky bit. Interrupt remains valid until EP5INT is cleared by writing in the corresponding USB\_CSR5 bit.

- **EP6INT: Endpoint 6 Interrupt Status**

0 = No Endpoint6 Interrupt pending.

1 = Endpoint6 Interrupt has been raised.

Several signals can generate this interrupt. The reason can be found by reading USB\_CSR6:

- RXSETUP set to 1

- RX\_DATA\_BK0 set to 1

- RX\_DATA\_BK1 set to 1

- TXCOMP set to 1

- STALLSENT set to 1

EP6INT is a sticky bit. Interrupt remains valid until EP6INT is cleared by writing in the corresponding USB\_CSR6 bit.

- **EP7INT: Endpoint 7 Interrupt Status**

0 = No Endpoint7 Interrupt pending.

1 = Endpoint7 Interrupt has been raised.

Several signals can generate this interrupt. The reason can be found by reading USB\_CSR7:

RXSETUP set to 1

RX\_DATA\_BK0 set to 1

RX\_DATA\_BK1 set to 1

TXCOMP set to 1

STALLSENT set to 1

EP7INT is a sticky bit. Interrupt remains valid until EP7INT is cleared by writing in the corresponding USB\_CSR7 bit.

- **RXSUSP: USB Suspend Interrupt Status**

0 = No USB Suspend Interrupt pending.

1 = USB Suspend Interrupt has been raised.

The USB device sets this bit when it detects no activity for 3ms. The USB device enters Suspend mode.

- **RXRSM: USB Resume Interrupt Status**

0 = No USB Resume Interrupt pending.

1 = USB Resume Interrupt has been raised.

The USB device sets this bit when a USB resume signal is detected at its port.

- **EXTRSM: External Resume Interrupt Status**

0 = No External Resume Interrupt pending.

1 = External Resume Interrupt has been raised.

This interrupt is raised when, in suspend mode, an asynchronous rising edge on the send\_resume is detected.

If RMWUPE = 1, a resume state is sent in the USB bus.

- **SOFINT: Start of Frame Interrupt Status**

0 = No Start of Frame Interrupt pending.

1 = Start of Frame Interrupt has been raised.

This interrupt is raised each time a SOF token has been detected. It can be used as a synchronization signal by using isochronous endpoints.

- **ENDBUSRES: End of BUS Reset Interrupt Status**

0 = No End of Bus Reset Interrupt pending.

1 = End of Bus Reset Interrupt has been raised.

This interrupt is raised at the end of a USB reset sequence. The USB device must prepare to receive requests on the endpoint 0. The host starts the enumeration, then performs the configuration.

- **WAKEUP: USB Resume Interrupt Status**

0 = No Wakeup Interrupt pending.

1 = A Wakeup Interrupt (USB Host Sent a RESUME or RESET) occurred since the last clear.

## USB Interrupt Clear Register

**Register Name:** USB\_ICR

**Access Type:** Write-only

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

- **RXSUSP: Clear USB Suspend Interrupt**

0 = No effect.

1 = Clear USB Suspend Interrupt.

- **RXRSM: Clear USB Resume Interrupt**

0 = No effect.

1 = Clear USB Resume Interrupt.

- **EXTRSM: Clear External Resume Interrupt**

0 = No effect.

1 = Clear External Resume Interrupt.

- **SOFINT: Clear Start Of Frame Interrupt**

0 = No effect.

1 = Clear Start Of Frame Interrupt.

- **ENDBURST: Clear End of Bus Reset Interrupt**

0 = No effect.

1 = Clear Start Of Frame Interrupt.

- **WAKEUP: Clear Wakeup Interrupt**

0 = No effect.

1 = Clear Wakeup Interrupt.



**USB Reset Endpoint Register**

**Register Name:** USB\_RST\_EP

**Access Type:** Read/write

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 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   |
| EP7 | EP6 | EP5 | EP4 | EP3 | EP2 | EP1 | EP0 |

- **EP0: Reset Endpoint 0**
- **EP1: Reset Endpoint 1**
- **EP2: Reset Endpoint 2**
- **EP3: Reset Endpoint 3**
- **EP4: Reset Endpoint 4**
- **EP5: Reset Endpoint 5**
- **EP6: Reset Endpoint 6**
- **EP7: Reset Endpoint 7**

This flag is used to reset the FIFO associated with the endpoint and the bit RXBYTECOUNT in the register UDP\_CSRx. It also resets the data toggle to DATA0. It is useful after removing a HALT condition on a BULK endpoint. Refer to Chapter 5.8.5 in the *USB Serial Bus Specification, Rev.1.1*.

Warning: This flag must be cleared at the end of the reset. It does not clear USB\_CSRx flags.

0 = No reset.

1 = Forces the corresponding endpoint FIFO pointers to 0, therefore RXBYTECNT field is read at 0 in USB\_CSRx register.

## USB Endpoint Control and Status Register

**Register Name:** USB\_CSRx [x = 0.. 7]

**Access Type:** Read/Write

|           |              |             |          |                    |           |              |        |
|-----------|--------------|-------------|----------|--------------------|-----------|--------------|--------|
| 31        | 30           | 29          | 28       | 27                 | 26        | 25           | 24     |
| –         | –            | –           | –        | –                  | RXBYTECNT |              |        |
| 23        | 22           | 21          | 20       | 19                 | 18        | 17           | 16     |
| RXBYTECNT |              |             |          |                    |           |              |        |
| 15        | 14           | 13          | 12       | 11                 | 10        | 9            | 8      |
| EPEDS     | –            | –           | –        | DTGLE              | EPTYPE    |              |        |
| 7         | 6            | 5           | 4        | 3                  | 2         | 1            | 0      |
| DIR       | RX_DATA_ BK1 | FORCE STALL | TXPKTRDY | STALLSENT ISOERROR | RXSETUP   | RX_DATA_ BK0 | TXCOMP |

- **TXCOMP: Generates an IN packet with data previously written in the DPR**

This flag generates an interrupt while it is set to one.

Write (Cleared by the firmware)

0 = Clear the flag, clear the interrupt.

1 = No effect.

Read (Set by the USB peripheral)

0 = Data IN transaction has not been acknowledged by the Host.

1 = Data IN transaction is achieved, acknowledged by the Host.

After having issued a Data IN transaction setting TXPKTRDY, the device firmware waits for TXCOMP to be sure that the host has acknowledged the transaction.

- **RX\_DATA\_BK0: Receive Data Bank 0**

This flag generates an interrupt while it is set to one.

Write (Cleared by the firmware)

0 = Notify USB peripheral device that data have been read in the FIFO's Bank 0.

1 = No effect.

Read (Set by the USB peripheral)

0 = No data packet has been received in the FIFO's Bank 0

1 = A data packet has been received, it has been stored in the FIFO's Bank 0.

When the device firmware has polled this bit or has been interrupted by this signal, it must transfer data from the FIFO to the microcontroller memory. The number of bytes received is available in RXBYTCENT field. Bank 0 FIFO values are read through the USB\_FDRx register. Once a transfer is done, the device firmware must release Bank 0 to the USB peripheral device by clearing RX\_DATA\_BK0.

- **RXSETUP: Sends STALL to the Host (Control endpoints)**

This flag generates an interrupt while it is set to one.

Read

0 = No setup packet available.

1 = A setup data packet has been sent by the host and is available in the FIFO.

Write

0 = Device firmware notifies the USB peripheral device that it has read the setup data in the FIFO.

1 = No effect.

This flag is used to notify the USB device firmware that a valid Setup data packet has been sent by the host and successfully received by the USB device. The USB device firmware may transfer Setup data from the FIFO by reading the USB\_FDRx register to the microcontroller memory. Once a transfer has been done, RXSETUP must be cleared by the device firmware.

Ensuing Data OUT transactions is not accepted while RXSETUP is set.

- **STALLSENT: Stall sent (Control, Bulk Interrupt endpoints)/ ISOERROR (Isochronous endpoints)**

This flag generates an interrupt while it is set to one.

STALLSENT: this ends a STALL handshake

Read

0 = the host has not acknowledged a STALL.

1 = host has acknowledge the stall.

Write

0 = reset the STALLSENT flag, clear the interrupt.

1 = No effect.

This is mandatory for the device firmware to clear this flag. Otherwise the interrupt remains.

Please refer to chapters 8.4.4 and 9.4.5 of the *Universal Serial Bus Specification, Rev. 1.1* to get more information on the STALL handshake.

ISOERROR: a CRC error has been detected in an isochronous transfer

Read

0 = No error in the previous isochronous transfer.

1 = CRC error has been detected, data available in the FIFO are corrupted.

Write

0 = reset the ISOERROR flag, clear the interrupt.

1 = No effect.

- **TXPKTRDY: Transmit Packet Ready**

This flag is cleared by the USB device.

This flag is set by the USB device firmware.

Read

0 = Data values can be written in the FIFO.

1 = Data values can not be written in the FIFO.

Write

0 = No effect.

1 = A new data payload is has been written in the FIFO by the firmware and is ready to be sent.

This flag is used to generate a Data IN transaction (device to host). Device firmware checks that it can write a data payload in the FIFO, checking that TXPKTRDY is cleared. Transfer to the FIFO is done by writing in the USB\_FDRx register. Once the data payload has been transferred to the FIFO, the firmware notifies the USB device setting TXPKTRDY to one. USB bus transactions can start. TXCOMP is set once the data payload has been received by the host.

- **FORCESTALL: Force Stall (used by Control, Bulk and Isochronous endpoints)**

Write-only

0 = No effect.

1 = Send STALL to the host.

Please refer to chapters 8.4.4 and 9.4.5 of the *Universal Serial Bus Specification, Rev. 1.1* to get more information on the STALL handshake.

Control endpoints: during the data stage and status stage, this indicates that the microcontroller can not complete the request.

Bulk and interrupt endpoints: notify the host that the endpoint is halted.

The host acknowledges the STALL, device firmware is notified by the STALLSENT flag.

- **RX\_DATA\_BK1: Receive Data Bank 1 (only used by endpoints with ping-pong attributes)**

This flag generates an interrupt while it is set to one.

Write (Cleared by the firmware)

0 = Notify USB device that data have been read in the FIFO's Bank 1.

1 = No effect.

Read (Set by the USB peripheral)

0 = No data packet has been received in the FIFO's Bank 1.

1 = A data packet has been received, it has been stored in FIFO's Bank 1.

When the device firmware has polled this bit or has been interrupted by this signal, it must transfer data from the FIFO to microcontroller memory. The number of bytes received is available in RXBYTECNT field. Bank 1 FIFO values are read through USB\_FDRx register. Once a transfer is done, the device firmware must release Bank 1 to the USB device by clearing RX\_DATA\_BK1.

- **DIR: Transfer Direction (only available for control endpoints)**

Read/Write

0 = Allow Data OUT transactions in the control data stage.

1 = Enable Data IN transactions in the control data stage.

Please refer to Chapter 8.5.2 of the *Universal Serial Bus Specification, Rev. 1.1* to get more information on the control data stage.

This bit must be set before USB\_CSRx/RXSETUP is cleared at the end of the setup stage. According to the request sent in the setup data packet, the data stage is either a device to host (DIR = 1) or host to device (DIR = 0) data transfer. It is not necessary to check this bit to reverse direction for the status stage.

- **EPTYPE[2:0]: Endpoint Type**

Read/Write

|     |                 |
|-----|-----------------|
| 000 | Control         |
| 001 | Isochronous OUT |
| 101 | Isochronous IN  |
| 010 | Bulk OUT        |
| 110 | Bulk IN         |
| 011 | Interrupt OUT   |
| 111 | Interrupt IN    |

- **DTGLE: Data Toggle**

Read-only

0 = Identifies DATA0 packet.

1 = Identifies DATA1 packet.

Please refer to Chapter 8 of the *Universal Serial Bus Specification, Rev. 1.1* to get more information on DATA0, DATA1 packet definitions.

- **EPEDS: Endpoint Enable Disable**

Read

0 = Endpoint disabled.

1 = Endpoint enabled.

Write

0 = Disable endpoint.

1 = Enable endpoint.

- **RXBYTECNT[10:0]: Number of Bytes Available in the FIFO**

Read-only.

When the host sends a data packet to the device, the USB device stores the data in the FIFO and notifies the microcontroller. The microcontroller can load the data from the FIFO by reading RXBYTECENT bytes in the USB\_FDRx register.

## USB FIFO Data Register

**Register Name:** USB\_FDRx [x = 0.. 7]

**Access Type:** Read/Write

|           |    |    |    |    |    |    |    |
|-----------|----|----|----|----|----|----|----|
| 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  |
| FIFO_DATA |    |    |    |    |    |    |    |

- **FIFO\_DATA[7:0]: FIFO Data Value**

The microcontroller can push or pop values in the FIFO through this register.

RXBYTECNT in the corresponding USB\_CSRx register is the number of bytes to be read from the FIFO (sent by the host).

The maximum number of bytes to write is fixed by the Max Packet Size in the Standard Endpoint Descriptor. It can not be more than the physical memory size associated to the endpoint. Please refer to the *Universal Serial Bus Specification, Rev. 1.1* to get more information.

## USB Host Port (UHP)

### Overview

The USB Host Port interfaces the USB with the host application. It handles Open HCI protocol (Open Host Controller Interface) as well as USB v2.0 Full-speed and Low-speed protocols. It also provides a simple read/write protocol on the ASB.

The USB Host Port integrates a root hub and transceivers on downstream ports. It provides several high-speed half-duplex serial communication ports at a baud rate of 12 Mbit/s. Up to 127 USB devices (printer, camera, mouse, keyboard, disk, etc.) and the USB hub can be connected to the USB host in the USB “tiered star” topology.

The USB Host Port controller is fully compliant with the Open HCI specification. The standard OHCI USB stack driver can be easily ported to ATMEL’s architecture in the same way all existing class drivers run without hardware specialization.

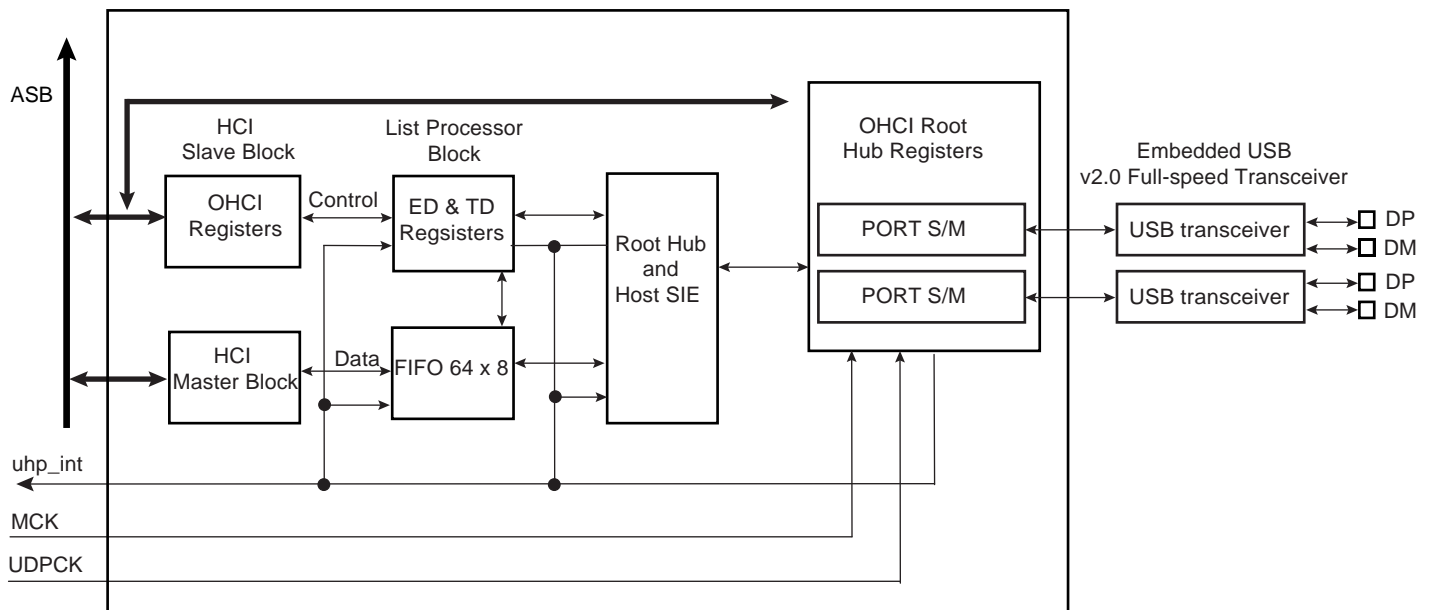
This means that all standard class devices are automatically detected and available to the user application. As an example, integrating an HID (Human Interface Device) class driver provides a plug & play feature for all USB keyboards and mice.

Key features of the USB Host Port are:

- Compliance with Open HCI Rev 1.0 Specification
- Compliance with USB V2.0 Full Speed and Low Speed Specification
- Supports Both Low-speed 1.5 Mbps and Full-speed 12 Mbps USB devices
- Root Hub Integrated with Two Downstream USB Ports
- Embedded USB Transceivers (Number of Transceivers is Product Dependant)
- Supports Power Management
- Operates as a Master on the ASB Bus

### Block Diagram

Figure 259. USB Host Port Block Diagram



Access to the USB host operational registers is achieved through the ASB bus interface. The Open HCI host controller initializes master DMA transfers with the ASB bus as follows:

- Fetches endpoint descriptors and transfer descriptors
- Access to endpoint data from system memory
- Access to the HC communication area
- Write status and retire transfer Descriptor

All of the ASB memory map is accessible to the USB host master DMA. Thus there is no need to define a dedicated physical memory area to the USB host.

The USB root hub is integrated in the USB host. Several USB downstream ports are available. The number of downstream ports can be determined by the software driver reading the root hub's operational registers. Device connection is automatically detected by the USB host port logic.

Warning: a pull-down must be connected to DP on the board. Otherwise The USB host will permanently detect a device connection on this port.

USB physical transceivers are integrated in the product and driven by the root hub's ports.

Over current protection on ports can be activated by the USB host controller. Atmel's standard product does not dedicate pads to external over current protection.

## Product Dependencies

### I/O Lines

DPs and DMs are not controlled by any PIO controllers. The embedded USB physical transceivers are controlled by the USB host controller.

### Power Management

The USB host controller requires a 48 MHz clock. This clock must be generated by a PLL with a correct accuracy of  $\pm 0.25\%$ .

Thus the USB device peripheral receives two clocks from the Power Management Controller (PMC): the master clock MCK used to drive the peripheral user interface (MCK domain) and the UHPCLK 48 MHz clock used to interface with the bus USB signals (Recovered 12 MHz domain).

### Interrupt

The USB host interface has an interrupt line connected to the Advanced Interrupt Controller (AIC).

Handling USB host interrupts requires programming the AIC before configuring the UHP.

## Functional Description

Please refer to the Open Host Controller Interface Specification for USB Release 1.0.a.

### Host Controller Interface

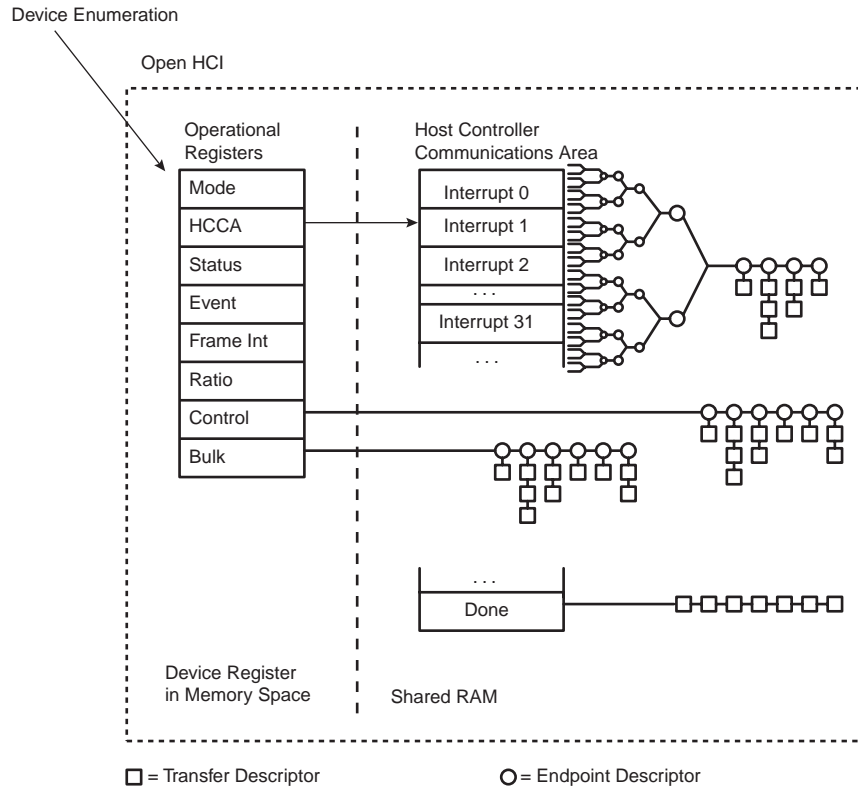
There are two communication channels between the Host Controller and the Host Controller Driver. The first channel uses a set of operational registers located on the USB Host Controller. The Host Controller is the target for all communications on this channel. The operational registers contain control, status and list pointer registers. They are mapped in the ASB memory mapped area. Within the operational register set there is a pointer to a location in the processor address space named the Host Controller Communication Area (HCCA). The HCCA is the second communication channel. The host controller is the master for all communication on this channel. The HCCA contains the head pointers to the interrupt Endpoint



Descriptor lists, the head pointer to the done queue and status information associated with start-of-frame processing.

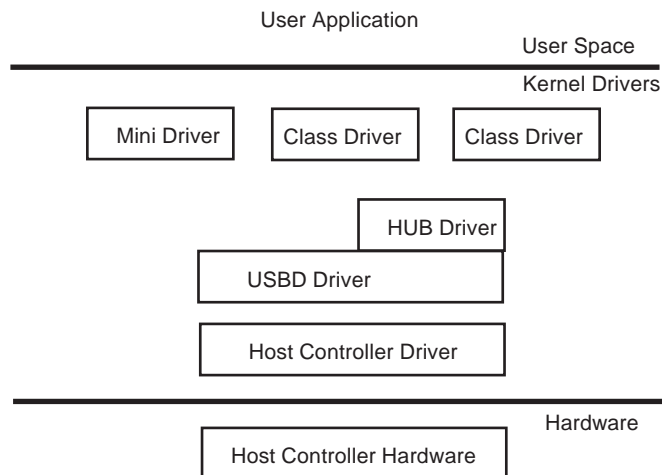
The basic building blocks for communication across the interface are Endpoint Descriptors (ED, 4 double words) and Transfer Descriptors (TD, 4 or 8 double words). The host controller assigns an Endpoint Descriptor to each endpoint in the system. A queue of Transfer Descriptors is linked to the Endpoint Descriptor for the specific endpoint.

**Figure 260. USB Host Communication Channels**



**Host Controller Driver**

**Figure 261. USB Host Drivers**

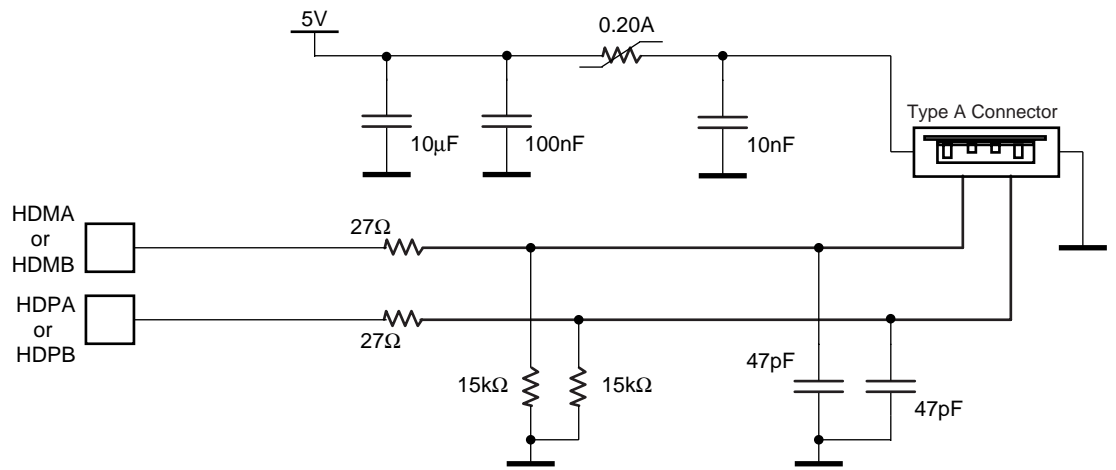


USB Handling is done through several layers as follows:

- Host controller hardware and serial engine: Transmit and receive USB data on the bus.
- Host controller driver: Drives the Host controller hardware and handle the USB protocol
- USB Bus driver and hub driver: Handles USB commands and enumeration. Offers a hardware independent interface.
- Mini driver: Handles device specific commands.
- Class driver: handles standard devices. This acts as a generic driver for a class of devices, for example the HID driver.

## Typical Connection

**Figure 262.** Board Schematic to Interface UHP Device Controller



As device connection is automatically detected by the USB host port logic, a pull-down must be connected on DP and DM on the board. Otherwise the USB host will permanently detect a device connection on this port.

## Ethernet MAC (EMAC)

### Overview

The Ethernet MAC is the hardware implementation of the MAC sub-layer OSI reference model between the physical layer (PHY) and the logical link layer (LLC). It controls the data exchange between a host and a PHY layer according to Ethernet IEEE 802.3u data frame format. The Ethernet MAC contains the required logic and transmit and receive FIFOs for DMA management. In addition, it is interfaced through MDIO/MDC pins for PHY layer management.

The Ethernet MAC can transfer data in media-independent interface (MII) or reduced media-independent interface (RMII) modes depending on the pinout configuration.

The aim of the reduced interface is to lower the pin count for a switch product that can be connected to multiple PHY interfaces. The characteristics specific to RMII mode are:

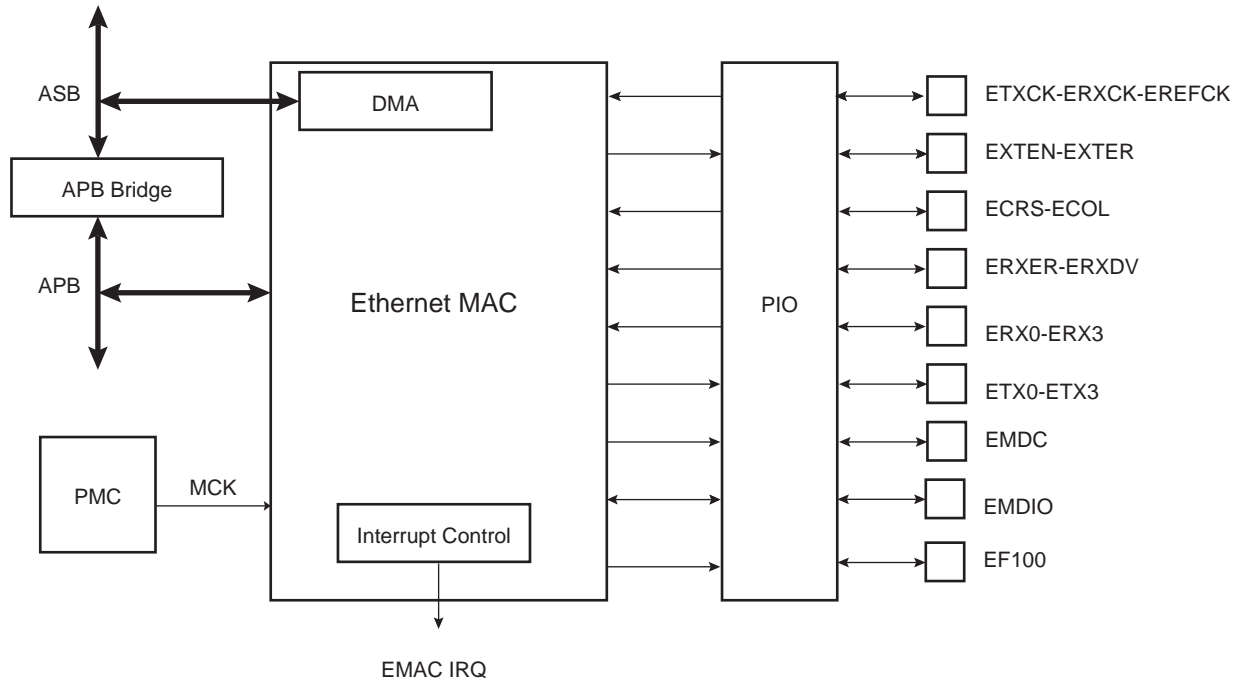
- Single clock at 50 MHz frequency
- Reduction of required control pins
- Reduction of data paths to di-bit (2-bit wide) by doubling clock frequency
- 10 Mbits/sec. and 100 Mbits/sec. data capability

The major features of the EMAC are:

- Compatibility with IEEE Standard 802.3
- 10 and 100 Mbits per second data throughput capability
- Full- and half-duplex operation
- MII or RMII interface to the physical layer
- Register interface to address, status and control registers
- DMA interface
- Interrupt generation to signal receive and transmit completion
- 28-byte transmit and 28-byte receive FIFOs
- Automatic pad and CRC generation on transmitted frames
- Address checking logic to recognize four 48-bit addresses
- Supports promiscuous mode where all valid frames are copied to memory
- Supports physical layer management through MDIO interface control of alarm and update time/calendar data in

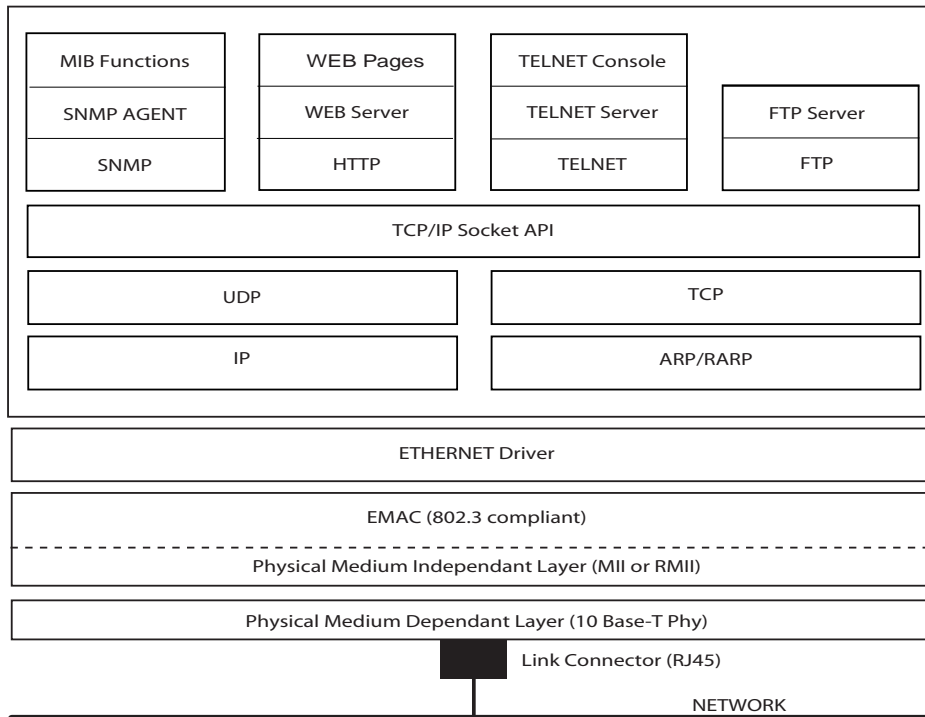
## Block Diagram

Figure 263. Block Diagram



## Application Block Diagram

Figure 264. Ethernet MAC Application Block Diagram



## Product Dependencies

### I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the EMAC pins to their peripheral functions. In RMII mode, unused pins (see Table 102: MII/RMII Signal Mapping) can be used as general I/O lines.

### Power Management

The EMAC may be clocked through the Power Management Controller (PMC), so the programmer must first configure the PMC to enable the EMAC clock.

### Interrupt

The EMAC has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling the EMAC interrupt requires programming the AIC before configuring the EMAC.

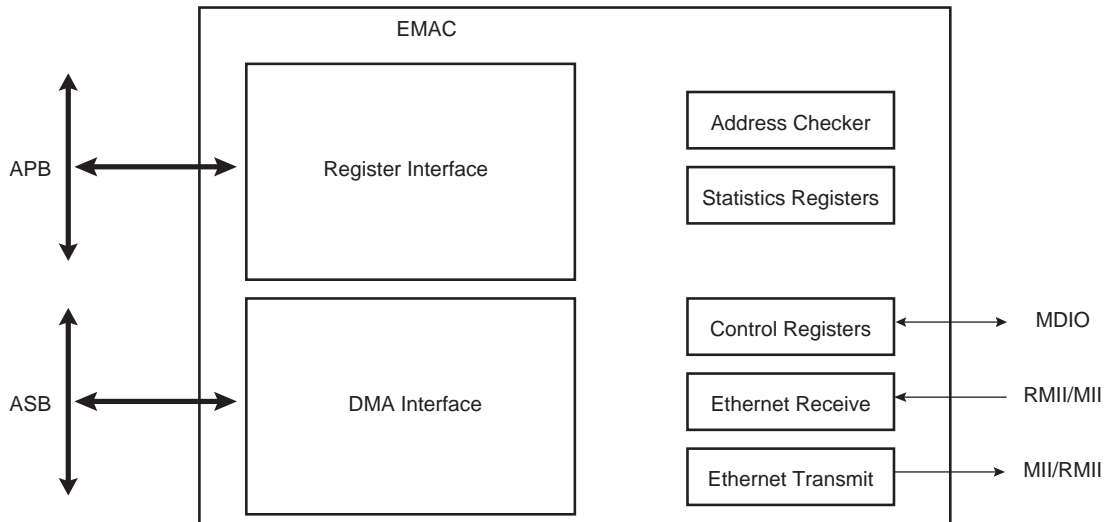
## Functional Description

The Ethernet Media Access Control (EMAC) engine is fully compatible with the IEEE 802.3 Ethernet standard. It manages frame transmission and reception including collision detection, preamble generation and detection, CRC control and generation and transmitted frame padding.

The MAC functions are:

- Frame encapsulation and decapsulation
- Error detection
- Media access management (MII, RMII)

**Figure 265.** EMAC Functional Block Diagram



## Media Independent Interface

### General

The Ethernet MAC is capable of interfacing to both RMII and MII Interfaces. The RMII bit in the ETH\_CFG register controls the interface that is selected. When this bit is set, the RMII interface is selected, else the MII interface is selected.

The MII and RMII interface are capable of both 10Mb/s and 100Mb/s data rates as described in the IEEE 802.3u standard. The signals used by the MII and RMII interfaces are described in the Table 102.

**Table 102.** Pin Configurations

| Pin Name    | MII                              | RMII                             |
|-------------|----------------------------------|----------------------------------|
| ETXCK_REFCK | ETXCK: Transmit Clock            | REFCK: Reference Clock           |
| ECRS_ECRSDV | ECRS: Carrier Sense              | ECRSDV: Carrier Sense/Data Valid |
| ECOL        | ECOL: Collision Detect           |                                  |
| ERXDV       | ERXDV: Data Valid                |                                  |
| ERX0 - ERX3 | ERX0 - ERX3: 4-bit Receive Data  | ERX0 - ERX1: 2-bit Receive Data  |
| ERXER       | ERXER: Receive Error             | ERXER: Receive Error             |
| ERXCK       | ERXCK: Receive Clock             |                                  |
| ETXEN       | ETXEN: Transmit Enable           | ETXEN: Transmit Enable           |
| ETX0-ETX3   | ETX0 - ETX3: 4-bit Transmit Data | ETX0 - ETX1: 2-bit Transmit Data |
| ETXER       | ETXER: Transmit Error            |                                  |

The intent of the RMII is to provide a reduced pin count alternative to the IEEE 802.3u MII. It uses 2 bits for transmit (ETX0 and ETX1) and two bits for receive (ERX0 and ERX1). There is a Transmit Enable (ETXEN), a Receive Error (ERXER), a Carrier Sense (ECRS\_DV), and a 50 MHz Reference Clock (ETXCK\_REFCK) for 100Mb/s data rate.

### RMII Transmit and Receive Operation

The same signals are used internally for both the RMII and the MII operations. The RMII maps these signals in a more pin-efficient manner. The transmit and receive bits are converted from a 4-bit parallel format to a 2-bit parallel scheme that is clocked at twice the rate. The carrier sense and data valid signals are combined into the ECRS\_ECRSDV signal. This signal contains information on carrier sense, FIFO status, and validity of the data. Transmit error bit (ETXER) and collision detect (ECOL) are not used in RMII mode.

## Transmit/Receive Operation

A standard IEEE 802.3 packet consists of the following fields: preamble, start of frame delimiter (SFD), destination address (DA), source address (SA), length, data (Logical Link Control Data) and frame check sequence CRC32 (FCS).

**Table 103.** Packet Format

| Preamble          |        | Frame <sup>(1)</sup> |         |             |          |     |         |
|-------------------|--------|----------------------|---------|-------------|----------|-----|---------|
| Alternating 1s/0s | SFD    | DA                   | SA      | Length/type | LLC Data | PAD | FCS     |
| Up to 7 bytes     | 1 byte | 6 bytes              | 6 bytes | 2 bytes     |          |     | 4 bytes |

Note: Frame Length between 64 bytes and 1518 bytes.

The packets are Manchester-encoded and -decoded and transferred serially using NRZ data with a clock. All fields are of fixed length except for the data field. The MAC generates and appends the preamble, SFD and CRC fields during transmission.

The preamble and SFD fields are stripped during reception.

### Preamble and Start of Frame Delimiter (SFD)

The preamble field is used to acquire bit synchronization with an incoming packet. When transmitted, each packet contains 62 bits of alternating 1,0 preamble. Some of this preamble is lost as the packet travels through the network. Byte alignment is performed with the Start of Frame Delimiter (SFD) pattern that consists of two consecutive 1's.

### Destination Address

The destination address (DA) indicates the destination of the packet on the network and is used to filter unwanted packets. There are three types of address formats: physical, multicast and broadcast. The physical address is a unique address that corresponds only to a single node. All physical addresses have an MSB of 0.

Multicast addresses begin with an MSB of 1. The MAC filters multicast addresses using a standard hashing algorithm that maps all multicast addresses into a 6-bit value. This 6-bit value indexes a 64-bit array that filters the value. If the address consists of all ones, it is a broadcast address, indicating that the packet is intended for all nodes.

### Source Address

The source address (SA) is the physical address of the node that sent the packet. Source addresses cannot be multicast or broadcast addresses. This field is passed to buffer memory.

### Length/Type

If the value of this field is less than or equal to 1500, then the Length/Type field indicates the number of bytes in the subsequent LLC Data field. If the value of this field is greater than or equal to 1536, then the Length/Type field indicates the nature of the MAC client protocol (protocol type).

### LLC Data

The data field consists of anywhere from 46 to 1500 bytes. Messages longer than 1500 bytes need to be broken into multiple packets. Messages shorter than 46 bytes require appending a pad to bring the data field to the minimum length of 46 bytes. If the data field is padded, the number of valid data bytes is indicated in the length field.

### FCS Field

The Frame Check Sequence (FCS) is a 32-bit CRC field, calculated and appended to a packet during transmission to allow detection of errors when a packet is received. During reception, error free packets result in a specific pattern in the CRC generator. Packets with improper CRC will be rejected.



## Frame Format Extensions

The original Ethernet standards defined the minimum frame size as 64 bytes and the maximum as 1518 bytes. These numbers include all bytes from the Destination MAC Address field through the Frame Check Sequence field. The Preamble and Start Frame Delimiter fields are not included when quoting the size of a frame. The IEEE 802.3ac standard extended the maximum allowable frame size to 1522 bytes to allow a VLAN tag to be inserted into the Ethernet frame format. The bit BIG defined in the ETH\_CFG register aims to process packet with VLAN tag.

The VLAN protocol permits insertion of an identifier, or tag, into the Ethernet frame format to identify the VLAN to which the frame belongs. It allows frames from stations to be assigned to logical groups. This provides various benefits, such as easing network administration, allowing formation of work groups, enhancing network security, and providing a means of limiting broadcast domains (refer to IEEE standard 802.1Q for definition of the VLAN protocol). The 802.3ac standard defines only the implementation details of the VLAN protocol that are specific to Ethernet.

If present, the 4-byte VLAN tag is inserted into the Ethernet frame between the Source MAC Address field and the Length field. The first 2-bytes of the VLAN tag consist of the “802.1Q Tag Type” and are always set to a value of 0x8100. The 0x8100 value is a reserved Length/Type field assignment that indicates the presence of the VLAN tag, and signals that the traditional Length/Type field can be found at an offset of four bytes further into the frame. The last two bytes of the VLAN tag contain the following information:

- The first three bits are a User Priority Field that may be used to assign a priority level to the Ethernet frame.
- The following one bit is a Canonical Format Indicator (CFI) used in Ethernet frames to indicate the presence of a Routing Information Field (RIF).
- The last twelve bits are the VLAN Identifier (VID) that uniquely identifies the VLAN to which the Ethernet frame belongs.

With the addition of VLAN tagging, the 802.3ac standard permits the maximum length of an Ethernet frame to be extended from 1518 bytes to 1522 bytes. Table 104 illustrates the format of an Ethernet frame that has been “tagged” with a VLAN identifier according to the IEEE 802.3ac standard.

**Table 104.** Ethernet Frame with VLAN Tagging

|                               |             |
|-------------------------------|-------------|
| Preamble                      | 7 bytes     |
| Start Frame Delimiter         | 1 byte      |
| Dest. MAC Address             | 6 bytes     |
| Source MAC Address            | 6 bytes     |
| Length/Type = 802.1Q Tag Type | 2 byte      |
| Tag Control Information       | 2 bytes     |
| Length / Type                 | 2 bytes     |
| MAC Client Data               | 0 - n bytes |
| Pad                           | 0 - p bytes |
| Frame Check Sequence          | 4 bytes     |

## DMA Operations

Frame data is transferred to and from the Ethernet MAC via the DMA interface. All transfers are 32-bit words and may be single accesses or bursts of two, three or four words. Burst accesses do not cross 16-byte boundaries.

The DMA controller performs four types of operations on the ASB bus. In order of priority, these operations are receive buffer manager read, receive buffer manager write, transmit data DMA read and receive data DMA write.

## Transmitter Mode

Transmit frame data needs to be stored in contiguous memory locations. It does not need to be word-aligned.

The transmit address register is written with the address of the first byte to be transmitted.

Transmit is initiated by writing the number of bytes to transfer (length) to the transmit control register.

The transmit channel then reads data from memory 32 bits at a time and places them in the transmit FIFO.

The transmit block starts frame transmission when three words have been loaded into the FIFO.

The transmit address register must be written before the transmit control register. While a frame is being transmitted, it is possible to set up one other frame for transmission by writing new values to the transmit address and control registers. Reading the transmit address register returns the address of the buffer currently being accessed by the transmit FIFO.

Reading the transmit control register returns the total number of bytes to be transmitted. The BNQ bit in the Transmit Status Register indicates whether another buffer can be safely queued. An interrupt is generated whenever this bit is set.

Frame assembly starts by adding preamble and the start frame delimiter. Data is taken from the transmit FIFO word-by-word. If necessary, padding is added to make the frame length 60 bytes. The CRC is calculated as a 32-bit polynomial. This is inverted and appended to the end of the frame, making the frame length a minimum of 64 bytes. The CRC is not appended if the NCRC bit is set in the transmit control register.

In full-duplex mode, frames are transmitted immediately. Back-to-back frames are transmitted at least 96 bit times apart to guarantee the inter-frame gap.

In half-duplex mode, the transmitter checks carrier sense. If asserted, it waits for it to de-assert and then starts transmission after the inter-frame gap of 96 bit-times.

If the collision signal is asserted during transmission, the transmitter transmits a jam sequence of 32 bits taken from the data register and then retries transmission after the backoff time has elapsed. An error is indicated and any further attempts aborted if 16 attempts cause collisions.

If transmit DMA underruns, bad CRC is automatically appended using the same mechanism as jam insertion. Underrun also causes TXER to be asserted.

## Receiver Mode

When a packet is received, it is checked for valid preamble, CRC, alignment, length and address. If all these criteria are met, the packet is stored successfully in a receive buffer. If at the end of reception the CRC is bad, then the received buffer is recovered. Each received frame including CRC is written to a single receive buffer.

Receive buffers are word-aligned and are capable of containing 1518 or 1522 bytes (BIG = 1 in ETH\_CFG) of data (the maximum length of an Ethernet frame).

The start location for each received frame is stored in memory in a list of receive buffer descriptors at a location pointed to by the receive buffer queue pointer register. Each entry in

the list consists of two words. The first word is the address of the received buffer; the second is the receive status. Table 105 defines an entry in the received buffer descriptor list.

To receive frames, the buffer queue must be initialized by writing an appropriate address to bits [31:2] in the first word of each list entry. Bit zero of word zero must be written with zero.

After a frame is received, bit zero becomes set and the second word indicates what caused the frame to be copied to memory. The start location of the received buffer descriptor list should be written to the received buffer queue pointer register before receive is enabled (by setting the receive enable bit in the network control register). As soon as the received block starts writing received frame data to the receive FIFO, the received buffer manager reads the first receive buffer location pointed to by the received buffer queue pointer register. If the filter block is active, the frame should be copied to memory; the receive data DMA operation starts writing data into the receive buffer. If an error occurs, the buffer is recovered. If the frame is received without error, the queue entry is updated. The buffer pointer is rewritten to memory with its low-order bit set to indicate successful frame reception and a used buffer. The next word is written with the length of the frame and how the destination address was recognized. The next receive buffer location is then read from the following word or, if the current buffer pointer had its wrap bit set, the beginning of the table. The maximum number of buffer pointers before a wrap bit is seen is 1024. If a wrap bit is not seen by then, a wrap bit is assumed in that entry. The received buffer queue pointer register must be written with zero in its lower-order bit positions to enable the wrap function to work correctly.

If bit zero is set when the receive buffer manager reads the location of the receive buffer, then the buffer has already been used and cannot be used again until software has processed the frame and cleared bit zero. In this case, the DMA block sets the buffer unavailable bit in the received status register and triggers an interrupt. The frame is discarded and the queue entry is reread on reception of the next frame to see if the buffer is now available. Each discarded frame increments a statistics register that is cleared on being read. When there is network congestion, it is possible for the MAC to be programmed to apply backpressure.

This is when half-duplex mode collisions are forced on all received frames by transmitting 64 bits of data (a default pattern).

Reading the received buffer queue register returns the location of the queue entry currently being accessed. The queue wraps around to the start after either 1024 entries (i.e., 2048 words) or when the wrap bit is found to be set in bit 1 of the first word of an entry.

**Table 105.** Received Buffer Descriptor List

| Bit           | Function  |
|---------------|---|
| <b>Word 0</b> |   |
| 31:2          | Base address of receive buffer  |
| 1             | Wrap bit. If this bit is set, the counter that is ORed with the received buffer queue pointer register to give the pointer to entries in this table is cleared after the buffer is used.  |
| 0             | Ownership bit. 1 indicates software owns the pointer, 0 indicates that the DMA owns the buffer. If this bit is not zero when the entry is read by the receiver, the buffer unavailable bit is set in the received status register and the receiver goes inactive. |
| <b>Word 1</b> |   |
| 31            | Global all ones broadcast address detected  |
| 30            | Multicast hash match  |
| 29            | Unicast hash match  |

**Table 105.** Received Buffer Descriptor List

| Bit   | Function   |
|-------|--|
| 28    | External address (optional)                      |
| 27    | Unknown source address (reserved for future use) |
| 26    | Local address match (Specific address 1 match)   |
| 25    | Local address match (Specific address 2 match)   |
| 24    | Local address match (Specific address 3 match)   |
| 23    | Local address match (Specific address 4 match)   |
| 22:11 | Reserved; written to 0                           |
| 10:0  | Length of frame including FCS                    |

## Address Checking

Whether or not a frame is stored depends on what is enabled in the network configuration register, the contents of the specific address and hash registers and the frame destination address. In this implementation of the MAC the frame source address is not checked.

A frame is not copied to memory if the MAC is transmitting in half-duplex mode at the time a destination address is received.

The hash register is 64 bits long and takes up two locations in the memory map.

There are four 48-bit specific address registers, each taking up two memory locations. The first location contains the first four bytes of the address; the second location contains the last two bytes of the address stored in its least significant byte positions. The addresses stored can be specific, group, local or universal.

Ethernet frames are transmitted a byte at a time, LSB first. The first bit (i.e., the LSB of the first byte) of the destination address is the group/individual bit and is set one for multicast addresses and zero for unicast. This bit corresponds to bit 24 of the first word of the specific address register. The MSB of the first byte of the destination address corresponds to bit 31 of the specific address register.

The specific address registers are compared to the destination address of received frames once they have been activated. Addresses are deactivated at reset or when the first byte [47:40] is written and activated or when the last byte [7:0] is written. If a receive frame address matches an active address, the local match signal is set and the store frame pulse signal is sent to the DMA block via the HCLK synchronization block.

A frame can also be copied if a unicast or multicast hash match occurs, it has the broadcast address of all ones, or the copy all frames bit in the network configuration register is set.

The broadcast address of 0xFFFFFFFF is recognized if the no broadcast bit in the network configuration register is zero. This sets the broadcast match signal and triggers the store frame signal.

The unicast hash enable and the multicast hash enable bits in the network configuration register enable the reception of hash matched frames. So all multicast frames can be received by setting all bits in the hash register.

The CRC algorithm reduces the destination address to a 6-bit index into a 64-bit hash register. If the equivalent bit in the register is set, the frame is matched depending on whether the frame is multicast or unicast and the appropriate match signals are sent to the DMA block. If the copy all frames bit is set in the network configuration register, the store frame pulse is always sent to the DMA block as soon as any destination address is received.

## Ethernet MAC (EMAC) User Interface

**Table 106.** EMAC Register Mapping

| Offset                              | Register                             | Register Name | Read/Write | Reset  |
|-------------------------------------|--------------------------------------|---------------|------------|--------|
| 0x00                                | EMAC Control Register                | ETH_CTL       | Read/write | 0x0    |
| 0x04                                | EMAC Configuration Register          | ETH_CFG       | Read/write | 0x800  |
| 0x08                                | EMAC Status Register                 | ETH_SR        | Read-only  | 0x6    |
| 0x0C                                | EMAC Transmit Address Register       | ETH_TAR       | Read/Write | 0x0    |
| 0x10                                | EMAC Transmit Control Register       | ETH_TCR       | Read/write | 0x0    |
| 0x14                                | EMAC Transmit Status Register        | ETH_TSR       | Read/write | 0x18   |
| 0x18                                | EMAC Receive Buffer Queue Pointer    | ETH_RBQP      | Read/write | 0x0    |
| 0x1C                                | Reserved                             | –             | Read-only  | 0x0    |
| 0x20                                | EMAC Receive Status Register         | ETH_RSR       | Read/write | 0x0    |
| 0x24                                | EMAC Interrupt Status Register       | ETH_ISR       | Read/write | 0x0    |
| 0x28                                | EMAC Interrupt Enable Register       | ETH_IER       | Write-only | –      |
| 0x2C                                | EMAC Interrupt Disable Register      | ETH_IDR       | Write-only | –      |
| 0x30                                | EMAC Interrupt Mask Register         | ETH_IMR       | Read-only  | 0xFFFF |
| 0x34                                | EMAC PHY Maintenance Register        | ETH_MAN       | Read/write | 0x0    |
| Statistics Registers <sup>(1)</sup> |                                      |               |            |        |
| 0x40                                | Frames Transmitted OK Register       | ETH_FRA       | Read/write | 0x0    |
| 0x44                                | Single Collision Frame Register      | ETH_SCOL      | Read/write | 0x0    |
| 0x48                                | Multiple Collision Frame Register    | ETH_MCOL      | Read/write | 0x0    |
| 0x4C                                | Frames Received OK Register          | ETH_OK        | Read/write | 0x0    |
| 0x50                                | Frame Check Sequence Error Register  | ETH_SEQE      | Read/write | 0x0    |
| 0x54                                | Alignment Error Register             | ETH_ALE       | Read/write | 0x0    |
| 0x58                                | Deferred Transmission Frame Register | ETH_DTE       | Read/write | 0x0    |
| 0x5C                                | Late Collision Register              | ETH_LCOL      | Read/write | 0x0    |
| 0x60                                | Excessive Collision Register         | ETH_ECOL      | Read/write | 0x0    |
| 0x64                                | Carrier Sense Error Register         | ETH_CSE       | Read/write | 0x0    |
| 0x68                                | Transmit Underrun Error Register     | ETH_TUE       | Read/write | 0x0    |
| 0x6C                                | Code Error Register                  | ETH_CDE       | Read/write | 0x0    |
| 0x70                                | Excessive Length Error Register      | ETH_ELR       | Read/write | 0x0    |
| 0x74                                | Receive Jabber Register              | ETH_RJB       | Read/write | 0x0    |
| 0x78                                | Undersize Frame Register             | ETH_USF       | Read/write | 0x0    |
| 0x7C                                | SQE Test Error Register              | ETH_SQEE      | Read/write | 0x0    |
| 0x80                                | Discarded RX Frame Register          | ETH_DRFC      | Read/write | 0x0    |
| Address Registers                   |                                      |               |            |        |
| 0x90                                | EMAC Hash Address High [63:32]       | ETH_HSH       | Read/write | 0x0    |

**Table 106.** EMAC Register Mapping

| Offset | Register                                   | Register Name | Read/Write | Reset |
|--------|--|---------------|------------|-------|
| 0x94   | EMAC Hash Address Low [31:0]               | ETH_HSL       | Read/write | 0x0   |
| 0x98   | EMAC Specific Address 1 Low, First 4 Bytes | ETH_SA1L      | Read/write | 0x0   |
| 0x9C   | EMAC Specific Address 1 High, Last 2 Bytes | ETH_SA1H      | Read/write | 0x0   |
| 0xA0   | EMAC Specific Address 2 Low, First 4 Bytes | ETH_SA2L      | Read/write | 0x0   |
| 0xA4   | EMAC Specific Address 2 High, Last 2 Bytes | ETH_SA2H      | Read/write | 0x0   |
| 0xA8   | EMAC Specific Address 3 Low, First 4 Bytes | ETH_SA3L      | Read/write | 0x0   |
| 0xAC   | EMAC Specific Address 3 High, Last 2 Bytes | ETH_SA3H      | Read/write | 0x0   |
| 0xB0   | EMAC Specific Address 4 Low, First 4 Bytes | ETH_SA4L      | Read/write | 0x0   |
| 0xB4   | EMAC Specific Address 4 High, Last 2 Bytes | ETH_SA4H      | Read/write | 0x0   |

Note: For further details on the statistics registers, see Table 107 on page 593.

## EMAC Control Register

**Name:** ETH\_CTL  
**Access Type:** Read/write

|     |     |     |     |    |    |     |    |
|-----|-----|-----|-----|----|----|-----|----|
| 31  | 30  | 29  | 28  | 27 | 26 | 25  | 24 |
| –   | –   | –   | –   | –  | –  | –   | –  |
| 23  | 22  | 21  | 20  | 19 | 18 | 17  | 16 |
| –   | –   | –   | –   | –  | –  | –   | –  |
| 15  | 14  | 13  | 12  | 11 | 10 | 9   | 8  |
| –   | –   | –   | –   | –  | –  | –   | BP |
| 7   | 6   | 5   | 4   | 3  | 2  | 1   | 0  |
| WES | ISR | CSR | MPE | TE | RE | LBL | LB |

- **LB: Loopback**

Optional. When set, loopback signal is at high level.

- **LBL: Loopback Local**

When set, connects ETX[3:0] to ERX[3:0], ETXEN to ERXDV, forces full duplex and drives ERXCK and ETXCK\_REFCK with MCK divided by 4.

- **RE: Receive Enable**

When set, enables the Ethernet MAC to receive data.

- **TE: Transmit Enable**

When set, enables the Ethernet transmitter to send data.

- **MPE: Management Port Enable**

Set to one to enable the management port. When zero, forces MDIO to high impedance state.

- **CSR: Clear Statistics Registers**

This bit is write-only. Writing a one clears the statistics registers.

- **ISR: Increment Statistics Registers**

This bit is write-only. Writing a one increments all the statistics registers by one for test purposes.

- **WES: Write Enable for Statistics Registers**

Setting this bit to one makes the statistics registers writable for functional test purposes.

- **BP: Back Pressure**

If this field is set, then in half-duplex mode collisions are forced on all received frames by transmitting 64 bits of data (default pattern).



## EMAC Configuration Register

Name: ETH\_CFG

Access Type: Read/Write

|     |     |      |     |     |    |     |     |
|-----|-----|------|-----|-----|----|-----|-----|
| 31  | 30  | 29   | 28  | 27  | 26 | 25  | 24  |
| –   | –   | –    | –   | –   | –  | –   | –   |
| 23  | 22  | 21   | 20  | 19  | 18 | 17  | 16  |
| –   | –   | –    | –   | –   | –  | –   | –   |
| 15  | 14  | 13   | 12  | 11  | 10 | 9   | 8   |
| –   | –   | RMII | RTY | CLK |    | EAE | BIG |
| 7   | 6   | 5    | 4   | 3   | 2  | 1   | 0   |
| UNI | MTI | NBC  | CAF | –   | BR | FD  | SPD |

- **SPD: Speed**

Set to 1 to indicate 100 Mbit/sec, 0 for 10 Mbit/sec. Has no other functional effect.

- **FD: Full Duplex**

If set to 1, the transmit block ignores the state of collision and carrier sense and allows receive while transmitting.

- **BR: Bit Rate**

Optional.

- **CAF: Copy All Frames**

When set to 1, all valid frames are received.

- **NBC: No Broadcast**

When set to 1, frames addressed to the broadcast address of all ones are not received.

- **MTI: Multicast Hash Enable**

When set multicast frames are received when six bits of the CRC of the destination address point to a bit that is set in the hash register.

- **UNI: Unicast Hash Enable**

When set, unicast frames are received when six bits of the CRC of the destination address point to a bit that is set in the hash register.

- **BIG: Receive 1522 Bytes**

When set, the MAC receives up to 1522 bytes. Normally the MAC receives frames up to 1518 bytes in length.

This bit allows to receive extended Ethernet frame with "VLAN tag" (IEEE 802.3ac)

- **EAE: External Address Match Enable**

Optional.

- **CLK**

The system clock (MCK) is divided down to generate MDC (the clock for the MDIO). To conform with IEEE standard 802.3 MDC must not exceed 2.5 MHz. At reset this field is set to 10 so that MCK is divided by 32.

| CLK | MDC               |
|-----|-------------------|
| 00  | MCK divided by 8  |
| 01  | MCK divided by 16 |
| 10  | MCK divided by 32 |
| 11  | MCK divided by 64 |



- **RTY: Retry Test**

When set, the time between frames is always one time slot. For test purposes only. Must be cleared for normal operation.

- **RMII: Reduce MII**

When set, this bit enables the RMII operation mode. When reset, it selects the MII mode.

## EMAC Status Register

Name: ETH\_SR

Access Type: Read only

|    |    |    |    |    |      |      |      |
|----|----|----|----|----|------|------|------|
| 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    |
| –  | –  | –  | –  | –  | IDLE | MDIO | LINK |

- **LINK**

Reserved.

- **MDIO**

0 = MDIO pin not set.

1 = MDIO pin set.

- **IDLE**

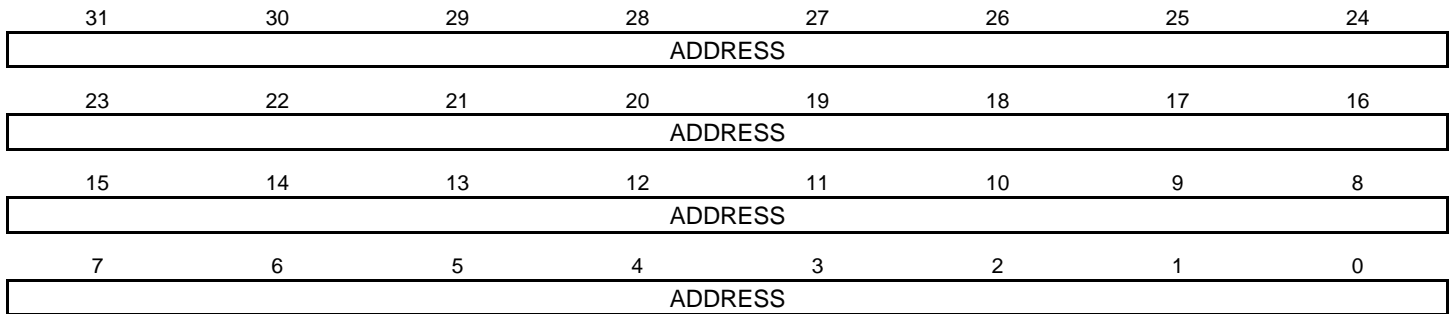
0 = PHY logic is idle.

1 = PHY logic is running.

**EMAC Transmit Address Register**

**Name:** ETH\_TAR

**Access Type:** Read/Write



- **ADDRESS: Transmit Address Register**

Written with the address of the frame to be transmitted, read as the base address of the buffer being accessed by the transmit FIFO. Note that if the two least significant bits are not zero, transmit starts at the byte indicated.

## EMAC Transmit Control Register

**Name:** ETH\_TCR  
**Access Type:** Read/Write

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

- **LEN: Transmit Frame Length**

This register is written to the number of bytes to be transmitted excluding the four CRC bytes unless the no CRC bit is asserted. Writing these bits to any non-zero value initiates a transmission. If the value is greater than 1514 (1518 if no CRC is being generated), an oversize frame is transmitted. This field is buffered so that a new frame can be queued while the previous frame is still being transmitted. Must always be written in address-then-length order. Reads as the total number of bytes to be transmitted (i.e., this value does not change as the frame is transmitted.) Frame transmission does not start until two 32-bit words have been loaded into the transmit FIFO. The length must be great enough to ensure two words are loaded.

- **NCRC: No CRC**

If this bit is set, it is assumed that the CRC is included in the length being written in the low-order bits and the MAC does not append CRC to the transmitted frame. If the buffer is not at least 64 bytes long, a short frame is sent. This field is buffered so that a new frame can be queued while the previous frame is still being transmitted. Reads as the value of the frame currently being transmitted.

**EMAC Transmit Status Register**

**Name:** ETH\_TSR

**Access Type:** Read/Write

|    |     |      |     |      |     |     |     |
|----|-----|------|-----|------|-----|-----|-----|
| 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   |
| –  | UND | COMP | BNQ | IDLE | RLE | COL | OVR |

• **OVR: Ethernet Transmit Buffer Overrun**

Software has written to the Transmit Address Register (ETH\_TAR) or Transmit Control Register (ETH\_TCR) when bit BNQ was not set. Cleared by writing a one to this bit.

• **COL: Collision Occurred**

Set by the assertion of collision. Cleared by writing a one to this bit.

• **RLE: Retry Limit Exceeded**

Cleared by writing a one to this bit.

• **IDLE: Transmitter Idle**

Asserted when the transmitter has no frame to transmit. Cleared when a length is written to transmit frame length portion of the Transmit Control register. This bit is read-only.

• **BNQ: Ethernet Transmit Buffer not Queued**

Software may write a new buffer address and length to the transmit DMA controller when set. Cleared by having one frame ready to transmit and another in the process of being transmitted. This bit is read-only.

• **COMP: Transmit Complete**

Set when a frame has been transmitted. Cleared by writing a one to this bit.

• **UND: Transmit Underrun**

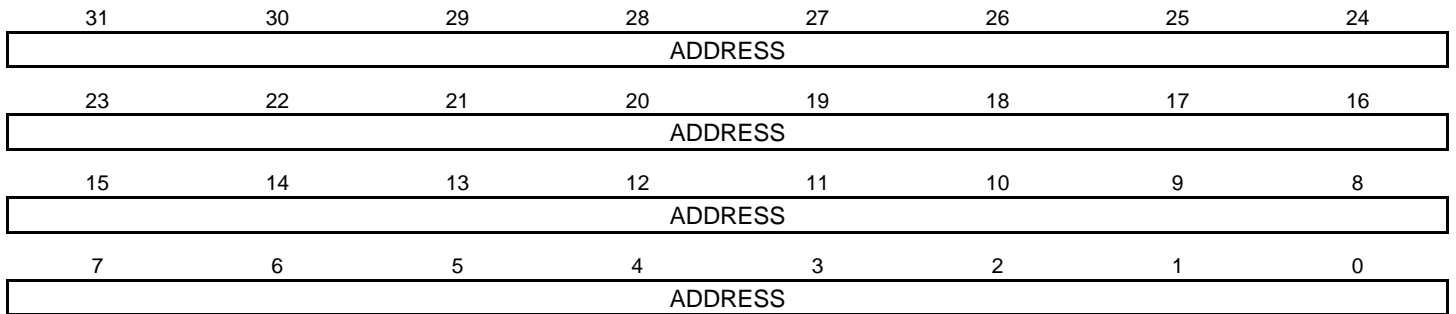
Set when transmit DMA was not able to read data from memory in time. If this happens, the transmitter forces bad CRC. Cleared by writing a one to this bit.



## EMAC Receive Buffer Queue Pointer Register

Name: ETH\_RBQP

Access Type: Read/Write



- **ADDRESS: Receive Buffer Queue Pointer**

Written with the address of the start of the receive queue, reads as a pointer to the current buffer being used. The receive buffer is forced to word alignment.

**EMAC Receive Status Register**

**Name:** ETH\_RSR

**Access Type:** Read/Write

|    |    |    |    |    |     |     |     |
|----|----|----|----|----|-----|-----|-----|
| 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   |
| –  | –  | –  | –  | –  | OVR | REC | BNA |

- **BNA: Buffer Not Available**

An attempt was made to get a new buffer and the pointer indicated that it was owned by the processor. The DMA rereads the pointer each time a new frame starts until a valid pointer is found. This bit is set at each attempt that fails even if it has not had a successful pointer read since it has been cleared. Cleared by writing a one to this bit.

- **REC: Frame Received**

One or more frames have been received and placed in memory. Cleared by writing a one to this bit.

- **OVR: RX Overrun**

The DMA block was unable to store the receive frame to memory, either because the ASB bus was not granted in time or because a not OK HRESP was returned. The buffer is recovered if this happens. Cleared by writing a one to this bit.

## EMAC Interrupt Status Register

**Name:** ETH\_ISR

**Access Type:** Read/Write

|      |      |      |      |      |      |      |       |
|------|------|------|------|------|------|------|-------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24    |
| –    | –    | –    | –    | –    | –    | –    | –     |
| 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16    |
| –    | –    | –    | –    | –    | –    | –    | –     |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8     |
| –    | –    | –    | –    | ABT  | ROVR | LINK | TIDLE |
| 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0     |
| TCOM | TBRE | RTRY | TUND | TOVR | RBNA | RCOM | DONE  |

- **DONE: Management Done**

The PHY maintenance register has completed its operation. Cleared on read.

- **RCOM: Receive Complete**

A frame has been stored in memory. Cleared on read.

- **RBNA: Receive Buffer Not Available**

Cleared on read.

- **TOVR: Transmit Buffer Overrun**

Software has written to the Transmit Address Register (ETH\_TAR) or Transmit Control Register (ETH\_TCR) when BNQ of the Transmit Status Register (ETH\_TSR) was not set. Cleared on read.

- **TUND: Transmit Buffer Underrun**

Ethernet transmit buffer underrun. The transmit DMA did not complete fetch frame data in time for it to be transmitted. Cleared on read.

- **RTRY: Retry Limit**

Retry limit exceeded. Cleared on read.

- **TBRE: Transmit Buffer Register Empty**

Software may write a new buffer address and length to the transmit DMA controller. Cleared by having one frame ready to transmit and another in the process of being transmitted. Cleared on read.

- **TCOM: Transmit Complete**

Set when a frame has been transmitted. Cleared on read.

- **TIDLE: Transmit Idle**

Set when all frames have been transmitted. Cleared on read.

- **LINK**

Set when LINK pin changes value. Optional.

- **ROVR: RX Overrun**

Set when the RX overrun status bit is set. Cleared on read.

- **ABT: Abort**

Set when an abort occurs during a DMA transfer. Cleared on read.



**EMAC Interrupt Enable Register**

**Name:** ETH\_IER

**Access Type:** Write only

|      |      |      |      |      |      |      |       |
|------|------|------|------|------|------|------|-------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24    |
| –    | –    | –    | –    | –    | –    | –    | –     |
| 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16    |
| –    | –    | –    | –    | –    | –    | –    | –     |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8     |
| –    | –    | –    | –    | ABT  | ROVR | LINK | TIDLE |
| 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0     |
| TCOM | TBRE | RTRY | TUND | TOVR | RBNA | RCOM | DONE  |

- **DONE: Management Done Interrupt Enable**
- **RCOM: Receive Complete Interrupt Enable**
- **RBNA: Receive Buffer Not Available Interrupt Enable**
- **TOVR: Transmit Buffer Overrun Interrupt Enable**
- **TUND: Transmit Buffer Underrun Interrupt Enable**
- **RTRY: Retry Limit Interrupt Enable**
- **TBRE: Transmit Buffer Register Empty Interrupt Enable**
- **TCOM: Transmit Complete Interrupt Enable**
- **TIDLE: Transmit Idle Interrupt Enable**
- **LINK: LINK Interrupt Enable**
- **ROVR: RX Overrun Interrupt Enable**
- **ABT: Abort Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

## EMAC Interrupt Disable Register

Name: ETH\_IDR

Access Type: Write only

|      |      |      |      |      |      |      |       |
|------|------|------|------|------|------|------|-------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24    |
| –    | –    | –    | –    | –    | –    | –    | –     |
| 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16    |
| –    | –    | –    | –    | –    | –    | –    | –     |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8     |
| –    | –    | –    | –    | ABT  | ROVR | LINK | TIDLE |
| 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0     |
| TCOM | TBRE | RTRY | TUND | TOVR | RBNA | RCOM | DONE  |

- **DONE: Management Done Interrupt Disable**
- **RCOM: Receive Complete Interrupt Disable**
- **RBNA: Receive Buffer Not Available Interrupt Disable**
- **TOVR: Transmit Buffer Overrun Interrupt Disable**
- **TUND: Transmit Buffer Underrun Interrupt Disable**
- **RTRY: Retry Limit Interrupt Disable**
- **TBRE: Transmit Buffer Register Empty Interrupt Disable**
- **TCOM: Transmit Complete Interrupt Disable**
- **TIDLE: Transmit Idle Interrupt Disable**
- **LINK: LINK Interrupt Disable**
- **ROVR: RX Overrun Interrupt Disable**
- **ABT: Abort Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

### EMAC Interrupt Mask Register

Name: ETH\_IMR

Access Type: Read only

|      |      |      |      |      |      |      |       |
|------|------|------|------|------|------|------|-------|
| 31   | 30   | 29   | 28   | 27   | 26   | 25   | 24    |
| –    | –    | –    | –    | –    | –    | –    | –     |
| 23   | 22   | 21   | 20   | 19   | 18   | 17   | 16    |
| –    | –    | –    | –    | –    | –    | –    | –     |
| 15   | 14   | 13   | 12   | 11   | 10   | 9    | 8     |
| –    | –    | –    | –    | ABT  | ROVR | LINK | TIDLE |
| 7    | 6    | 5    | 4    | 3    | 2    | 1    | 0     |
| TCOM | TBRE | RTRY | TUND | TOVR | RBNA | RCOM | DONE  |

- **DONE: Management Done Interrupt Mask**
- **RCOM: Receive Complete Interrupt Mask**
- **RBNA: Receive Buffer Not Available Interrupt Mask**
- **TOVR: Transmit Buffer Overrun Interrupt Mask**
- **TUND: Transmit Buffer Underrun Interrupt Mask**
- **RTRY: Retry Limit Interrupt Mask**
- **TBRE: Transmit Buffer Register Empty Interrupt Mask**
- **TCOM: Transmit Complete Interrupt Mask**
- **TIDLE: Transmit Idle Interrupt Mask**
- **LINK: LINK Interrupt Mask**
- **ROVR: RX Overrun Interrupt Mask**
- **ABT: Abort Interrupt Mask**

0: The corresponding interrupt is enabled.

1: The corresponding interrupt is not enabled.

**Important Note:** The interrupt is disabled when the corresponding bit is set. This is non-standard for AT91 products as generally a mask bit set enables the interrupt.



## EMAC PHY Maintenance Register

Name: ETH\_MAN

Access Type: Read/Write

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

Writing to this register starts the shift register that controls the serial connection to the PHY. On each shift cycle the MDIO pin becomes equal to the MSB of the shift register and LSB of the shift register becomes equal to the value of the MDIO pin. When the shifting is complete an interrupt is generated and the IDLE field is set in the Network Status register.

When read, gives current shifted value.

- **DATA**

For a write operation this is written with the data to be written to the PHY. After a read operation this contains the data read from the PHY.

- **CODE**

Must be written to 10 in accordance with IEEE standard 802.3. Reads as written.

- **REGA**

Register address. Specifies the register in the PHY to access.

- **PHYA**

PHY address. Normally is 0.

- **RW**

Read/write Operation. 10 is read. 01 is write. Any other value is an invalid PHY management frame.

- **HIGH**

Must be written with 1 to make a valid PHY management frame. Conforms with IEEE standard 802.3.

- **LOW**

Must be written with 0 to make a valid PHY management frame. Conforms with IEEE standard 802.3.

**EMAC Hash Address High Register**

**Name:** ETH\_HSH  
**Access Type:** Read/Write

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

• **ADDR**

Hash address bits 63 to 32.

**EMAC Hash Address Low Register**

**Name:** ETH\_HSL  
**Access Type:** Read/Write

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

• **ADDR**

Hash address bits 31 to 0.

### EMAC Specific Address (1, 2, 3 and 4) High Register

Name: ETH\_SA1H,...ETH\_SA4H

Access Type: Read/Write

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

- ADDR

Unicast addresses (1, 2, 3 and 4), Bits 47:32.

### EMAC Specific Address (1, 2, 3 and 4) Low Register

Name: ETH\_SA1L,...ETH\_SA4L

Access Type: Read/Write

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

- ADDR

Unicast addresses (1, 2, 3 and 4), Bits 31:0.

## EMAC Statistics Register Block Registers

These registers reset to zero on a read and remain at all ones when they count to their maximum value. They should be read frequently enough to prevent loss of data.

The statistics register block contains the registers found in Table 107.

**Table 107.** Statistics Register Block

| Register                             | Register Name | Description  |
|--------------------------------------|---------------|--|
| Frames Transmitted OK Register       | ETH_FRA       | A 24-bit register counting the number of frames successfully transmitted.  |
| Single Collision Frame Register      | ETH_SCOL      | A 16-bit register counting the number of frames experiencing a single collision before being transmitted and experiencing no carrier loss nor underrun.  |
| Multiple Collision Frame Register    | ETH_MCOL      | A 16-bit register counting the number of frames experiencing between two and fifteen collisions prior to being transmitted (62 - 1518 bytes, no carrier loss, no underrun).  |
| Frames Received OK Register          | ETH_OK        | A 24-bit register counting the number of good frames received, i.e., address recognized. A good frame is of length 64 to 1518 bytes and has no FCS, alignment or code errors.  |
| Frame Check Sequence Error Register  | ETH_SEQE      | An 8-bit register counting address-recognized frames that are an integral number of bytes long, that have bad CRC and that are 64 to 1518 bytes long.  |
| Alignment Error Register             | ETH_ALE       | An 8-bit register counting frames that: <ul style="list-style-type: none"> <li>- are address-recognized,</li> <li>- are not an integral number of bytes long,</li> <li>- have bad CRC when their length is truncated to an integral number of bytes,</li> <li>- are between 64 and 1518 bytes long.</li> </ul> |
| Deferred Transmission Frame Register | ETH_DTE       | A 16-bit register counting the number of frames experiencing deferral due to carrier sense active on their first attempt at transmission (no underrun or collision).   |
| Late Collision Register              | ETH_LCOL      | An 8-bit register counting the number of frames that experience a collision after the slot time (512 bits) has expired. No carrier loss or underrun. A late collision is counted twice, i.e., both as a collision and a late collision.  |
| Excessive Collision Register         | ETH_ECOL      | An 8-bit register counting the number of frames that failed to be transmitted because they experienced 16 collisions (64 - 1518 bytes, no carrier loss or underrun).   |
| Carrier Sense Error Register         | ETH_CSE       | An 8-bit register counting the number of frames for which carrier sense was not detected and that were maintained in half-duplex mode one slot time (512 bits) after the start of transmission (no excessive collision).   |
| Transmit Underrun Error Register     | ETH_TUE       | An 8-bit register counting the number of frames not transmitted due to a transmit DMA underrun. If this register is incremented, then no other register is incremented.  |
| Code Error Register                  | ETH_CDE       | An 8-bit register counting the number of frames that are address-recognized, had RXER asserted during reception. If this counter is incremented, then no other counters are incremented.   |
| Excessive Length Error Register      | ETH_ELR       | An 8-bit register counting the number of frames received exceeding 1518 bytes in length but that do not have either a CRC error, an alignment error or a code error.   |
| Receive Jabber Register              | ETH_RJB       | An 8-bit register counting the number of frames received exceeding 1518 bytes in length and having either a CRC error, an alignment error or a code error.   |

**Table 107.** Statistics Register Block (Continued)

| Register                    | Register Name | Description  |
|-----------------------------|---------------|--|
| Undersize Frame Register    | ETH_USF       | An 8-bit register counting the number of frames received less than 64 bytes in length but that do not have either a CRC error, an alignment error or a code error. |
| SQE Test Error Register     | ETH_SQEE      | An 8-bit register counting the number of frames where pin ECOL was not asserted within a slot time of pin ETXEN being deasserted.                                  |
| Discarded RX Frame Register | ETH_DRFC      | This 16-bit counter is incremented every time an address-recognized frame is received but cannot be copied to memory because the receive buffer is available.      |



## AT91RM9200 Electrical Characteristics

### Absolute Maximum Ratings

**Table 108.** Absolute Maximum Ratings\*

|   |                 |
|---|-----------------|
| Operating Temperature (Industrial) .....  | -40°C to +85°C  |
| Storage Temperature .....   | -60°C to +150°C |
| Voltage on Input Pins<br>with Respect to Ground .....   | -0.3V to +3.6V  |
| Maximum Operating Voltage<br>(V <sub>DDCORE</sub> , V <sub>DDPLL</sub> and V <sub>DDOSC</sub> ) ..... | 1.95V           |
| Maximum Operating Voltage<br>(V <sub>DDIOM</sub> and V <sub>DDIOP</sub> ) .....                       | 3.6V            |
| DC Output Current<br>(SDA10, SDCKE, SDWE, RAS, CAS) .....   | 16 mA           |
| DC Output Current<br>(Any other pin) .....  | 8 mA            |

\*NOTICE: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

## DC Characteristics

The following characteristics are applicable to the operating temperature range:  $T_A = -40^{\circ}\text{C}$  to  $85^{\circ}\text{C}$ , unless otherwise specified and are certified for a junction temperature up to  $T_J = 100^{\circ}\text{C}$ .

**Table 109.** DC Characteristics

| Symbol       | Parameter                 | Conditions  | Min  | Typ  | Max                          | Units         |
|--------------|---------------------------|---|--|------|------------------------------|---------------|
| $V_{DDCORE}$ | DC Supply Core            |   | 1.65   |      | 1.95                         | V             |
| $V_{DDOSC}$  | DC Supply Oscillator      |   | 1.65   |      | 1.95                         | V             |
| $V_{DDPLL}$  | DC Supply PLL             |   | 1.65   |      | 1.95                         | V             |
| $V_{DDIOM}$  | DC Supply Memory I/Os     |   | $V_{DDCORE}$                                 |      | $V_{DDCORE} + 1.5$<br>or 3.6 | V             |
| $V_{DDIOP}$  | DC Supply Peripheral I/Os |   | $V_{DDCORE}$                                 |      | $V_{DDCORE} + 1.5$<br>or 3.6 | V             |
| $V_{IL}$     | Input Low-level Voltage   |   | -0.3   |      | 0.8                          | V             |
| $V_{IH}$     | Input High-level Voltage  |   | 2  |      | $V_{DD} + 0.3^{(1)}$         | V             |
| $V_{OL}$     | Output Low-level Voltage  | SDA10, SDCKE, SDWE, RAS, CAS pins:<br>$I_{OL} = 16 \text{ mA}^{(2)}$<br>$I_{OL} = 0 \text{ mA}^{(2)}$ |  |      | 0.4<br>0.2                   | V             |
|              |                           | Other pins:<br>$I_{OL} = 8 \text{ mA}^{(2)}$<br>$I_{OL} = 0 \text{ mA}^{(2)}$                         |  |      | 0.4<br>0.2                   |               |
| $V_{OH}$     | Output High-level Voltage | SDA10, SDCKE, SDWE, RAS, CAS pins:<br>$I_{OH} = 16 \text{ mA}^{(2)}$<br>$I_{OH} = 0 \text{ mA}^{(2)}$ | $V_{DD} - 0.4^{(1)}$<br>$V_{DD} - 0.2^{(1)}$ |      |                              | V             |
|              |                           | Other pins:<br>$I_{OH} = 8 \text{ mA}^{(2)}$<br>$I_{OH} = 0 \text{ mA}^{(2)}$                         | $V_{DD} - 0.4^{(1)}$<br>$V_{DD} - 0.2^{(1)}$ |      |                              |               |
| $I_{LEAK}$   | Input Leakage Current     | Pullup resistors disabled   |  |      | 1                            | $\mu\text{A}$ |
| $I_{PULL}$   | Input Pull-up Current     | $V_{DD} = 3.0\text{V}^{(1)}$ , $V_{IN} = 0$   | 129  |      |                              | $\mu\text{A}$ |
|              |                           | $V_{DD} = 3.6\text{V}^{(1)}$ , $V_{IN} = 0$   |  |      | 322                          |               |
| $C_{IN}$     | Input Capacitance         | 208-PQFP Package  |  |      | 8.8                          | pF            |
|              |                           | 256-LFBGA Package   |  |      | 7.6                          |               |
| $I_{SC}$     | Static Current            | On $V_{DDCORE} = 2\text{V}$ ,<br>MCK = 0 Hz   | $T_A = 25^{\circ}\text{C}$                   | 179  | 1157                         | $\mu\text{A}$ |
|              |                           | All inputs driven TMS,<br>TDI, TCK, NRST = 1  | $T_A = 85^{\circ}\text{C}$                   | 1610 | 7989                         |               |

Notes: 1.  $V_{DD}$  is applicable to  $V_{DDIOM}$ ,  $V_{DDIOP}$ ,  $V_{DDPLL}$  and  $V_{DDOSC}$   
2.  $I_O$  = Output Current.

## Clocks Characteristics

These parameters are given in the following conditions:

- $V_{DDCORE} = 1.8V$
- Ambient Temperature = 25°C

The Temperature Derating Factor described in the section “Temperature Derating Factor” on page 604 and  $V_{DDCORE}$  Voltage Derating Factor described in the section “VDDCORE Voltage Derating Factor” on page 604 are both applicable to these characteristics.

## Processor Clock Characteristics

**Table 110.** Processor Clock Waveform Parameters

| Symbol          | Parameter                     | Conditions | Min | Max   | Units |
|-----------------|-------------------------------|------------|-----|-------|-------|
| $1/(t_{CPPCK})$ | Processor Clock Frequency     |            |     | 209.0 | MHz   |
| $t_{CPPCK}$     | Processor Clock Period        |            | 4.8 |       | ns    |
| $t_{CHMCK}$     | Master Clock High Half-period |            | 2.2 |       | ns    |
| $t_{CLMCK}$     | Master Clock Low Half-period  |            | 2.2 |       | ns    |

## Master Clock Characteristics

**Table 111.** Master Clock Waveform Parameters

| Symbol          | Parameter                     | Conditions | Min  | Max  | Units |
|-----------------|-------------------------------|------------|------|------|-------|
| $1/(t_{CPMCK})$ | Master Clock Frequency        |            |      | 80.0 | MHz   |
| $t_{CPMCK}$     | Master Clock Period           |            | 12.5 |      | ns    |
| $t_{CHMCK}$     | Master Clock High Half-period |            | 6.3  |      | ns    |
| $t_{CLMCK}$     | Master Clock Low Half-period  |            | 6.3  |      | ns    |

## XIN Clock Characteristics <sup>(1)</sup>

**Table 112.** XIN Clock Electrical Characteristics

| Symbol          | Parameter                  | Conditions | Min                    | Max                    | Units |
|-----------------|----------------------------|------------|------------------------|------------------------|-------|
| $1/(t_{CPXIN})$ | XIN Clock Frequency        |            |                        | 50.0                   | MHz   |
| $t_{CPXIN}$     | XIN Clock Period           |            | 20.0                   |                        | ns    |
| $t_{CHXIN}$     | XIN Clock High Half-period |            | $0.4 \times t_{CPXIN}$ | $0.6 \times t_{CPXIN}$ |       |
| $t_{CLXIN}$     | XIN Clock Low Half-period  |            | $0.4 \times t_{CPXIN}$ | $0.6 \times t_{CPXIN}$ |       |
| $C_{IN}$        | XIN Input Capacitance      | Note (1)   |                        | 25                     | pF    |
| $R_{IN}$        | XIN Pulldown Resistor      | Note (1)   |                        | 500                    | kOhm  |

Notes: 1. These characteristics apply only when the Main Oscillator is in bypass mode (i.e., when MOSCEN = 0 in the CKGR\_MOR register. See “PMC Clock Generator Main Oscillator Register” on page 276.)

## Power Consumption

The values in Table 113 and Table 114 are measured values on the AT91RM9200DK Evaluation Board with operating conditions as follows:

- $V_{DDIO} = 3.3V$
- $V_{DDCORE} = V_{DDPLL} = V_{DDOSC} = 1.8V$
- $T_A = 25^{\circ}C$
- MCK = 60 MHz
- PCK = 180 Mhz
- SLCK = 32.768 kHz

These figures represent the power consumption measured on the  $V_{DDCORE}$  power supply.

**Table 113.** Power Consumption for PMC Modes<sup>(1)</sup>

| Mode       | Conditions  | Consumption | Unit |
|------------|---|-------------|------|
| Normal     | ARM Core clock enabled.<br>All peripheral clocks deactivated.                                     | 31.7        | mA   |
| Idle       | ARM Core clock disabled and waiting for the next interrupt.<br>All peripheral clocks deactivated. | 15.0        |      |
| Slow Clock | Main oscillator and PLLs are switched off.<br>Processor and all peripherals run at slow clock.    | 1.7         |      |
| Standby    | Combination of Idle and Slow Clock Modes.   | 1.7         |      |

Note: 1. Code in internal SRAM.

**Table 114.** Power Consumption by Peripheral<sup>(1)</sup>

| Peripheral                           | Consumption | Unit |
|--------------------------------------|-------------|------|
| PIO Controller                       | 0.6         | mA   |
| USART                                | 1.6         |      |
| MCI                                  | 1.9         |      |
| UDP                                  | 1.5         |      |
| TWI                                  | 0.4         |      |
| SPI                                  | 1.4         |      |
| SSC                                  | 1.8         |      |
| Timer Counter Channel                | 0.4         |      |
| UHP                                  | 3.4         |      |
| EMAC                                 | 4.3         |      |
| PMC                                  |             |      |
| PLL <sup>(2)</sup>                   | 3144        | uA   |
| Slow Clock Oscillator <sup>(3)</sup> | 858         | nA   |
| Main Oscillator <sup>(3)</sup>       | 350         | uA   |

- Notes: 1. Code in internal SRAM.  
2. Power consumption on the  $V_{DDPLL}$  power supply.  
3. Power consumption on the  $V_{DDOSC}$  power supply.

## Crystal Oscillators Characteristics

### 32 kHz Oscillator Characteristics

Table 115. 32 kHz Oscillator Characteristics

| Symbol            | Parameter                    | Conditions   | Min | Typ    | Max | Unit |
|-------------------|------------------------------|--|-----|--------|-----|------|
| $1/(t_{CP32KHz})$ | Crystal Oscillator Frequency |  |     | 32.768 |     | kHz  |
|                   | Duty Cycle                   | Measured at the PCK output pin   | 40  | 50     | 60  | %    |
| $t_{ST}$          | Startup Time                 | $V_{DDOSC} = 1.8V$<br>$R_s = 50\text{ k}\Omega$ , $C_L = 12.5\text{ pF}^{(1)}$ |     |        | 900 | ms   |

Note: 1.  $R_s$  is the equivalent series resistance,  $C_L$  is the equivalent load capacitance

## Main Oscillator Characteristics

Table 116. Main Oscillator Characteristics

| Symbol              | Parameter                                    | Conditions  | Min | Typ  | Max  | Unit |
|---------------------|--|---|-----|------|------|------|
| $1/(t_{CPMAIN})$    | Crystal Oscillator Frequency                 |   | 3   | 16   | 20   | MHz  |
| $C_{L1}$ , $C_{L2}$ | Internal Load Capacitance<br>( $CL1 = CL2$ ) |   |     | 25   |      | pF   |
| $C_L$               | Equivalent Load Capacitance                  | $C_{L1} = C_{L2} = 25\text{ pF}$  |     | 12.5 |      | pF   |
|                     | Duty Cycle                                   | Measured at the PCK output pin  | 40  | 50   | 60   | %    |
| $t_{ST}$            | Startup Time                                 | $V_{DDPLL} = 1.8V$<br>$1/(t_{CPMAIN}) = 3\text{ MHz}$<br>Without any capacitor connected to the main oscillator pins (XIN and XOUT) |     |      | 14.5 | ms   |

## PLL Characteristics

Table 117. Phase Lock Loop Characteristics

| Symbol    | Parameter        | Conditions | Min | Typ | Max | Unit    |
|-----------|------------------|------------|-----|-----|-----|---------|
| $F_{OUT}$ | Output Frequency |            | 80  |     | 240 | MHz     |
| $F_{IN}$  | Input Frequency  |            | 1   |     | 32  | MHz     |
| $K_O$     | VCO Gain         |            | 120 | 190 | 300 | MHz/V   |
| $I_P$     | Pump Current     |            | 36  | 44  | 60  | $\mu A$ |

## Transceiver Characteristics

### Electrical Characteristics

**Table 118.** Electrical Parameters

| Symbol        | Parameter                                | Conditions                                  | Min | Typ | Max | Unit    |
|---------------|--|---|-----|-----|-----|---------|
| Input Levels  |  |   |     |     |     |         |
| $V_{IL}$      | Low Level                                |   |     |     | 0.8 | V       |
| $V_{IH}$      | High Level                               |   | 2.0 |     |     | V       |
| $V_{DI}$      | Differential Input Sensivity             | $ (D+) - (D-) $                             | 0.2 |     |     | V       |
| $V_{CM}$      | Differential Input Common Mode Range     |   | 0.8 |     | 2.5 | V       |
| $C_{IN}$      | Transceiver capacitance                  | Capacitance to ground on each line          |     |     | 20  | pF      |
| I             | Hi-Z State Data Line Leakage             | $0V < V_{IN} < 3.3V$                        | -5  |     | +5  | $\mu A$ |
| $R_{EXT}$     | Recommended External USB Series Resistor | In series with each USB pin with $\pm 5\%$  |     | 27  |     |         |
| Output Levels |  |   |     |     |     |         |
| $V_{OL}$      | Low Level Output                         | Measured with RL of 1.425 kOhm tied to 3.6V |     |     | 0.3 | V       |
| $V_{OH}$      | High Level Output                        | Measured with RL of 14.25 kOhm tied to GND  | 2.8 |     |     | V       |
| $V_{CRS}$     | Output Signal Crossover Voltage          | Measure conditions described in Figure 266  | 1.3 |     | 2.0 | V       |

### Switching Characteristics

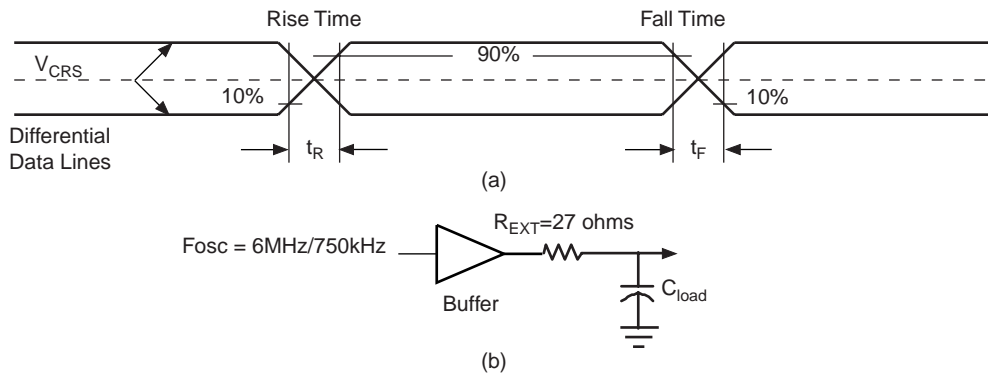
**Table 119.** In Slow Mode

| Symbol     | Parameter               | Conditions                  | Min | Typ | Max | Unit |
|------------|-------------------------|-----------------------------|-----|-----|-----|------|
| $t_{FR}$   | Transition Rise Time    | $C_{LOAD} = 400 \text{ pF}$ | 75  |     | 300 | ns   |
| $t_{FE}$   | Transition Fall Time    | $C_{LOAD} = 400 \text{ pF}$ | 75  |     | 300 | ns   |
| $t_{FRFM}$ | Rise/Fall time Matching | $C_{LOAD} = 400 \text{ pF}$ | 80  |     | 120 | %    |

**Table 120.** In Full Speed

| Symbol     | Parameter               | Conditions                 | Min | Typ | Max    | Unit |
|------------|-------------------------|----------------------------|-----|-----|--------|------|
| $t_{FR}$   | Transition Rise Time    | $C_{LOAD} = 50 \text{ pF}$ | 4   |     | 20     | ns   |
| $t_{FE}$   | Transition Fall Time    | $C_{LOAD} = 50 \text{ pF}$ | 4   |     | 20     | ns   |
| $t_{FRFM}$ | Rise/Fall Time Matching |                            | 90  |     | 111.11 | %    |

**Figure 266.** USB Data Signal Rise and Fall Times







## AT91RM9200 AC Characteristics

### Applicable Conditions and Derating Data

#### Conditions and Timings Computation

The delays are given as typical values in the following conditions:

- $V_{DDIOM} = 3.3V$
- $V_{DDCORE} = 1.8V$
- Ambient Temperature = 25°C
- Load Capacitance = 0 pF
- The output level change detection is  $(0.5 \times V_{DDIOM})$ .
- The input level is  $(0.3 \times V_{DDIOM})$  for a low-level detection and is  $(0.7 \times V_{DDIOM})$  for a high-level detection.

The minimum and maximum values given in the AC characteristics tables of this datasheet take into account process variation and design. In order to obtain the timing for other conditions, the following equation should be used:

$$t = \delta_{T^{\circ}} \times ((\delta_{VDDCORE} \times t_{DATASHEET}) + (\delta_{VDDIOM} \times \sum (C_{SIGNAL} \times \delta_{CSIGNAL})))$$

where:

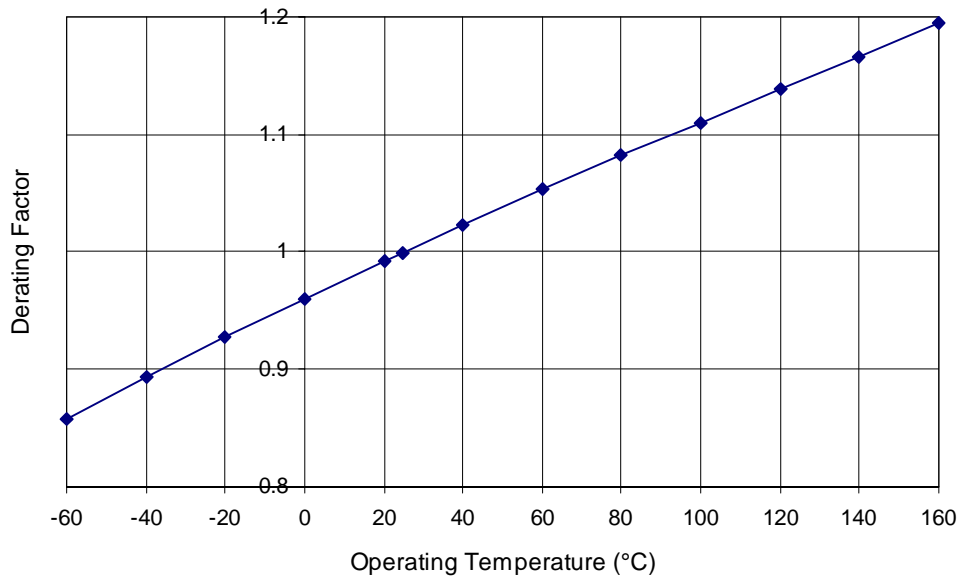
- $\delta_{T^{\circ}}$  is the derating factor in temperature given in Figure 267 on page 604.
- $\delta_{VDDCORE}$  is the derating factor for the Core Power Supply given in Figure 268 on page 604.
- $t_{DATASHEET}$  is the minimum or maximum timing value given in this datasheet for a load capacitance of 0 pF.
- $\delta_{VDDIOM}$  is the derating factor for the IOM Power Supply given in Figure 269 on page 605.
- $C_{SIGNAL}$  is the capacitance load on the considered output pin<sup>(1)</sup>.
- $\delta_{CSIGNAL}$  is the load derating factor depending on the capacitance load on the related output pins given in Min and Max in this datasheet.

The input delays are given as typical values.

Note: 1. The user must take into account the package capacitance load contribution ( $C_{IN}$ ) described in Table 109, "DC Characteristics," on page 596.

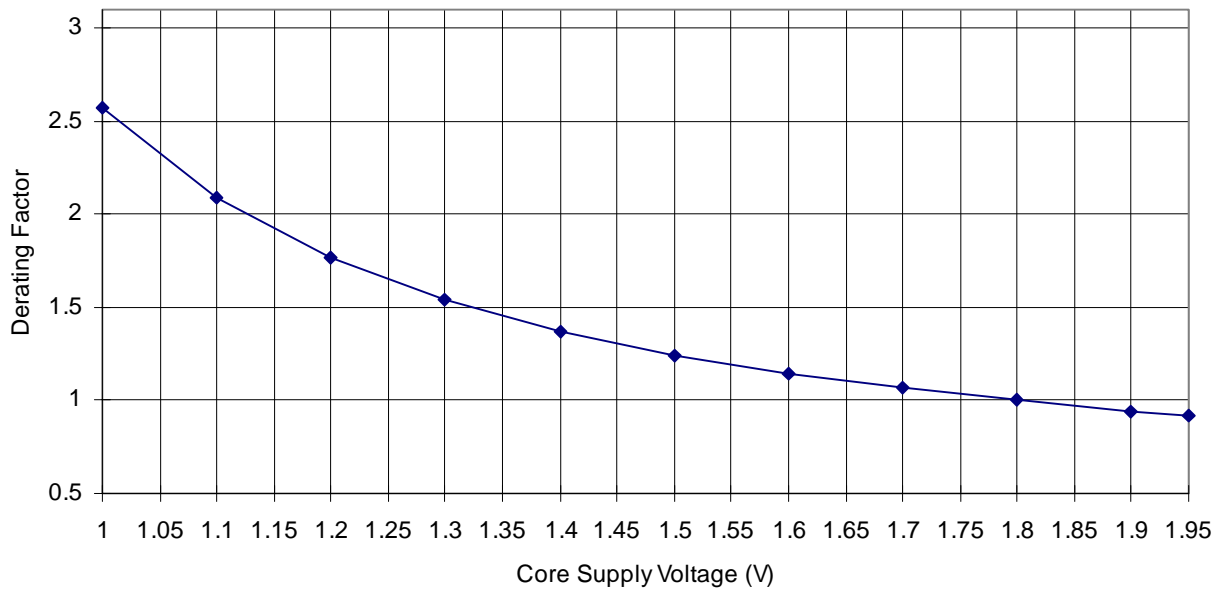
## Temperature Derating Factor

Figure 267. Derating Curve for Different Operating Temperatures



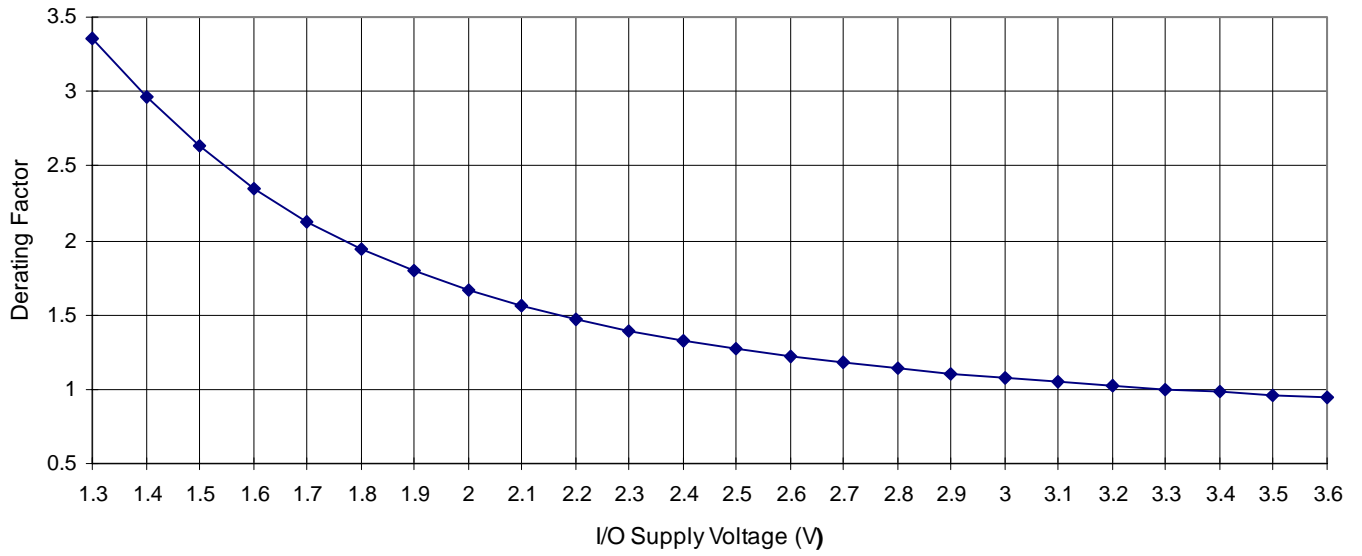
## V<sub>DDCORE</sub> Voltage Derating Factor

Figure 268. Derating Curve for Different Core Supply Voltages



**V<sub>DDIOM</sub> Voltage Derating Factor**

**Figure 269.** Derating Curve for Different IO Supply Voltages



Note: The derating factor in this example is applicable only to timings related to output pins.

## EBI Timings

### SMC Signals Relative to MCK

Table 121, Table 122 and Table 123 show timings relative to operating condition limits defined in the section “Conditions and Timings Computation” on page 603.

**Table 121.** General-purpose SMC Signals

| Symbol           | Parameter   | Conditions                | Min  | Max   | Units |
|------------------|---|---------------------------|--|---|-------|
| SMC <sub>1</sub> | MCK Falling to NUB Valid  | C <sub>NUB</sub> = 0 pF   | 5.0  | 7.5   | ns    |
|                  |   | C <sub>NUB</sub> derating | 0.028  | 0.045   | ns/pF |
| SMC <sub>2</sub> | MCK Falling to NLB/A0 Valid   | C <sub>NLB</sub> = 0 pF   | 4.9  | 7.5   | ns    |
|                  |   | C <sub>NLB</sub> derating | 0.028  | 0.045   | ns/pF |
| SMC <sub>3</sub> | MCK Falling to A1 - A25 Valid   | C <sub>ADD</sub> = 0 pF   | 4.9  | 7.4   | ns    |
|                  |   | C <sub>ADD</sub> derating | 0.028  | 0.045   | ns/pF |
| SMC <sub>4</sub> | MCK Falling to Chip Select Change (No Address to Chip Select Setup)               | C <sub>NCS</sub> = 0 pF   | 4.3  | 6.5   | ns    |
|                  |   | C <sub>NCS</sub> derating | 0.028  | 0.045   | ns/pF |
| SMC <sub>5</sub> | MCK Falling to Chip Select Active (Address to Chip Select Setup) <sup>(1)</sup>   | C <sub>NCS</sub> = 0 pF   | $(nacss \times t_{CPMCK}) + 4.3$ <sup>(2)</sup>                    | $(nacss \times t_{CPMCK}) + 6.5$ <sup>(2)</sup> | ns    |
|                  |   | C <sub>NCS</sub> derating | 0.028  | 0.045   | ns/pF |
| SMC <sub>6</sub> | Chip Select Inactive to MCK Falling (Address to Chip Select Setup) <sup>(1)</sup> | C <sub>NCS</sub> = 0 pF   | $(nacss \times t_{CPMCK}) + 4.4$ <sup>(2)</sup>                    | $(nacss \times t_{CPMCK}) + 6.5$ <sup>(2)</sup> | ns    |
|                  |   | C <sub>NCS</sub> derating | 0.028  | 0.045   | ns/pF |
| SMC <sub>7</sub> | NCS Minimum Pulse Width (Address to Chip Select Setup) <sup>(1)</sup>             | C <sub>NCS</sub> = 0 pF   | $((n + 2) - (2 \times nacss)) \times t_{CPMCK}$ <sup>(2) (3)</sup> |   | ns    |
| SMC <sub>8</sub> | NWAIT Minimum Pulse Width <sup>(1)</sup>  |                           | t <sub>CPMCK</sub>   |   | ns    |

- Notes:
1. The derating factor is not to be applied to t<sub>CPMCK</sub>.
  2. nacss = Number of Address to Chip Select Setup Cycles inserted.
  3. n = Number of standard Wait States inserted.

**Table 122. SMC Write Signals**

| Symbol            | Parameter  | Conditions                 | Min   | Max   | Units |
|-------------------|--|----------------------------|---|-------|-------|
| SMC <sub>10</sub> | MCK Rising to NWR Active (No Wait States) <sup>(5)</sup>                           | C <sub>NWR</sub> = 0 pF    | 4.8   | 7.2   | ns    |
|                   |  | C <sub>NWR</sub> derating  | 0.028                                       | 0.045 | ns/pF |
| SMC <sub>11</sub> | MCK Rising to NWR Active (Wait States)   | C <sub>NWR</sub> = 0 pF    | 4.8   | 7.2   | ns    |
|                   |  | C <sub>NWR</sub> derating  | 0.028                                       | 0.045 | ns/pF |
| SMC <sub>12</sub> | MCK Falling to NWR Inactive (No Wait States) <sup>(5)</sup>                        | C <sub>NWR</sub> = 0 pF    | 4.8   | 7.2   | ns    |
|                   |  | C <sub>NWR</sub> derating  | 0.028                                       | 0.045 | ns/pF |
| SMC <sub>13</sub> | MCK Rising to NWR Inactive (Wait States)   | C <sub>NWR</sub> = 0 pF    | 4.8   | 7.2   | ns    |
|                   |  | C <sub>NWR</sub> derating  | 0.028                                       | 0.045 | ns/pF |
| SMC <sub>14</sub> | MCK Rising to D0 - D15 Out Valid   | C <sub>DATA</sub> = 0 pF   | 4.1   | 7.9   | ns    |
|                   |  | C <sub>DATA</sub> derating | 0.028                                       | 0.044 | ns/pF |
| SMC <sub>15</sub> | NWR High to NUB Change <sup>(5)</sup>  | C <sub>NUB</sub> = 0 pF    | 3.4   |       | ns    |
|                   |  | C <sub>NUB</sub> derating  | 0.028                                       |       | ns/pF |
| SMC <sub>16</sub> | NWR High to NLB/A0 Change <sup>(5)</sup>   | C <sub>NLB</sub> = 0 pF    | 3.7   |       | ns    |
|                   |  | C <sub>NLB</sub> derating  | 0.028                                       |       | ns/pF |
| SMC <sub>17</sub> | NWR High to A1 - A25 Change <sup>(5)</sup>   | C <sub>ADD</sub> = 0 pF    | 3.3   |       | ns    |
|                   |  | C <sub>ADD</sub> derating  | 0.028                                       |       | ns/pF |
| SMC <sub>18</sub> | NWR High to Chip Select Inactive <sup>(5)</sup>                                    | C <sub>NCS</sub> = 0 pF    | 3.3   |       | ns    |
|                   |  | C <sub>NCS</sub> derating  | 0.028                                       |       | ns/pF |
| SMC <sub>19</sub> | Data Out Valid before NWR High (No Wait States) <sup>(1) (5)</sup>                 | C = 0 pF                   | t <sub>CHMCK</sub> - 0.8                    |       | ns    |
|                   |  | C <sub>DATA</sub> derating | - 0.044                                     |       | ns/pF |
|                   |  | C <sub>NWR</sub> derating  | 0.045                                       |       | ns/pF |
| SMC <sub>20</sub> | Data Out Valid before NWR High (Wait States) <sup>(1) (5)</sup>                    | C = 0 pF                   | n × t <sub>CPMCK</sub> - 0.6 <sup>(2)</sup> |       | ns    |
|                   |  | C <sub>DATA</sub> derating | - 0.044                                     |       | ns/pF |
|                   |  | C <sub>NWR</sub> derating  | 0.045                                       |       | ns/pF |
| SMC <sub>21</sub> | Data Out Valid after NWR High (No Wait States) <sup>(1) (5)</sup>                  | C = 0 pF                   | t <sub>CLMCK</sub> - 1.0                    |       | ns    |
|                   |  | C <sub>DATA</sub> derating | - 0.044                                     |       | ns/pF |
|                   |  | C <sub>NWR</sub> derating  | 0.045                                       |       | ns/pF |
| SMC <sub>22</sub> | Data Out Valid after NWR High (Wait States without Hold Cycles) <sup>(1) (5)</sup> | C = 0 pF                   | t <sub>CHMCK</sub> - 1.2                    |       | ns    |
|                   |  | C <sub>DATA</sub> derating | - 0.044                                     |       | ns/pF |
|                   |  | C <sub>NWR</sub> derating  | 0.045                                       |       | ns/pF |
| SMC <sub>23</sub> | Data Out Valid after NWR High (Wait States with Hold Cycles) <sup>(1) (5)</sup>    | C = 0 pF                   | h × t <sub>CPMCK</sub> - 1.1 <sup>(4)</sup> |       | ns    |
|                   |  | C <sub>DATA</sub> derating | - 0.044                                     |       | ns/pF |
|                   |  | C <sub>NWR</sub> derating  | 0.045                                       |       | ns/pF |

**Table 122. SMC Write Signals (Continued)**

| Symbol            | Parameter  | Conditions                 | Min   | Max | Units |
|-------------------|--|----------------------------|---|-----|-------|
| SMC <sub>24</sub> | Data Out Valid before NCS High<br>(Address to Chip Select Setup Cycles) <sup>(1)</sup> | C = 0 pF                   | $((n + 1) - nacss) \times t_{CPMCK} + t_{CHMCK} - 1.4$ <sup>(2) (3)</sup> |     | ns    |
|                   |  | C <sub>DATA</sub> derating | - 0.044   |     | ns/pF |
|                   |  | C <sub>NCS</sub> derating  | 0.045   |     | ns/pF |
| SMC <sub>25</sub> | Data Out Valid after NCS High<br>(Address to Chip Select Setup Cycles) <sup>(1)</sup>  | C = 0 pF                   | $nacss \times t_{CPMCK} - 0.4$ <sup>(3)</sup>                             |     | ns    |
|                   |  | C <sub>DATA</sub> derating | - 0.044   |     | ns/pF |
|                   |  | C <sub>NCS</sub> derating  | 0.045   |     | ns/pF |
| SMC <sub>26</sub> | NWR Minimum Pulse Width<br>(No Wait States) <sup>(1) (5)</sup>                         | C <sub>NWR</sub> = 0 pF    | $t_{CHMCK} - 0.1$   |     | ns    |
|                   |  | C <sub>NWR</sub> derating  | 0.002   |     | ns/pF |
| SMC <sub>27</sub> | NWR Minimum Pulse Width<br>(Wait States) <sup>(1) (5)</sup>                            | C <sub>NWR</sub> = 0 pF    | $n \times t_{CPMCK}$ <sup>(2)</sup>                                       |     | ns    |
|                   |  | C <sub>NWR</sub> derating  | 0.002   |     | ns/pF |
| SMC <sub>28</sub> | NWR Minimum Pulse Width<br>(Address to Chip Select Setup Cycles) <sup>(1)</sup>        | C <sub>NWR</sub> = 0 pF    | $(n + 1) \times t_{CPMCK}$ <sup>(2)</sup>                                 |     | ns    |
|                   |  | C <sub>NWR</sub> derating  | 0.002   |     | ns/pF |

- Notes:
1. The derating factor is not to be applied to  $t_{CLMCK}$ ,  $t_{CHMCK}$  or  $t_{CPMCK}$ .
  2. n = Number of standard Wait States inserted.
  3. nacss = Number of Address to Chip Select Setup Cycles inserted.
  4. h = Number of Hold Cycles inserted.
  5. Not applicable when Address to Chip Select Setup Cycles are inserted.

**Table 123. SMC Read Signals**

| Symbol            | Parameter   | Conditions                | Min   | Max   | Units |
|-------------------|---|---------------------------|---|---|-------|
| SMC <sub>29</sub> | MCK Falling to NRD Active <sup>(1) (7)</sup>      | C <sub>NRD</sub> = 0 pF   | 4.5   | 6.8   | ns    |
|                   |   | C <sub>NRD</sub> derating | 0.028   | 0.045   | ns/pF |
| SMC <sub>30</sub> | MCK Rising to NRD Active <sup>(2)</sup>           | C <sub>NRD</sub> = 0 pF   | 4.7   | 7.0   | ns    |
|                   |   | C <sub>NRD</sub> derating | 0.028   | 0.045   | ns/pF |
| SMC <sub>31</sub> | MCK Falling to NRD Inactive <sup>(1) (7)</sup>    | C <sub>NRD</sub> = 0 pF   | 4.5   | 6.8   | ns    |
|                   |   | C <sub>NRD</sub> derating | 0.028   | 0.045   | ns/pF |
| SMC <sub>32</sub> | MCK Falling to NRD Inactive <sup>(2)</sup>        | C <sub>NRD</sub> = 0 pF   | 4.5   | 6.8   | ns    |
|                   |   | C <sub>NRD</sub> derating | 0.028   | 0.045   | ns/pF |
| SMC <sub>33</sub> | D0-D15 in Setup before MCK Falling <sup>(8)</sup> |                           | 0.8   |   | ns    |
| SMC <sub>34</sub> | D0-D15 in Hold after MCK Falling <sup>(9)</sup>   |                           | 1.7   |   | ns    |
| SMC <sub>35</sub> | NRD High to NUB Change <sup>(3)</sup>             | C <sub>NUB</sub> = 0 pF   | $(h \times t_{CPMCK}) + 0.5$ <sup>(6)</sup>     | $(h \times t_{CPMCK}) + 0.8$ <sup>(6)</sup>     | ns    |
|                   |   | C <sub>NUB</sub> derating | 0.028   | 0.045   | ns/pF |
| SMC <sub>36</sub> | NRD High to NLB/A0 Change <sup>(3)</sup>          | C <sub>NLB</sub> = 0 pF   | $(h \times t_{CPMCK}) + 0.4$ <sup>(6)</sup>     | $(h \times t_{CPMCK}) + 0.7$ <sup>(6)</sup>     | ns    |
|                   |   | C <sub>NLB</sub> derating | 0.028   | 0.045   | ns/pF |
| SMC <sub>37</sub> | NRD High to A1-A25 Change <sup>(3)</sup>          | C <sub>ADD</sub> = 0 pF   | $(h \times t_{CPMCK}) + 0.3$ <sup>(6)</sup>     | $(h \times t_{CPMCK}) + 0.6$ <sup>(6)</sup>     | ns    |
|                   |   | C <sub>ADD</sub> derating | 0.028   | 0.045   | ns/pF |
| SMC <sub>38</sub> | NRD High to Chip Select Inactive <sup>(3)</sup>   | C <sub>NCS</sub> = 0 pF   | $(h \times t_{CPMCK}) - 0.3$ <sup>(6)</sup>     | $(h \times t_{CPMCK}) - 0.2$ <sup>(6)</sup>     | ns    |
|                   |   | C <sub>NCS</sub> derating | -0.045  | -0.028  | ns/pF |
| SMC <sub>39</sub> | Chip Select Inactive to NRD High <sup>(3)</sup>   | C <sub>NCS</sub> = 0 pF   | $(nacss \times t_{CPMCK}) + 0.2$ <sup>(5)</sup> | $(nacss \times t_{CPMCK}) + 0.3$ <sup>(5)</sup> | ns    |
|                   |   | C <sub>NCS</sub> derating | 0.028   | 0.045   | ns/pF |
| SMC <sub>40</sub> | Data Setup before NRD High <sup>(8)</sup>         | C <sub>NRD</sub> = 0 pF   | 7.5   |   | ns    |
|                   |   | C <sub>NRD</sub> derating | 0.045   |   | ns/pF |
| SMC <sub>41</sub> | Data Hold after NRD High <sup>(9)</sup>           | C <sub>NRD</sub> = 0 pF   | -3.4  |   | ns    |
|                   |   | C <sub>NRD</sub> derating | -0.028  |   | ns/pF |
| SMC <sub>42</sub> | Data Setup before NCS High                        | C <sub>NRD</sub> = 0 pF   | 7.3   |   | ns    |
|                   |   | C <sub>NRD</sub> derating | 0.045   |   | ns/pF |
| SMC <sub>43</sub> | Data Hold after NCS High                          | C <sub>NRD</sub> = 0 pF   | -3.2  |   | ns    |
|                   |   | C <sub>NRD</sub> derating | -0.028  |   | ns/pF |
| SMC <sub>44</sub> | NRD Minimum Pulse Width <sup>(1) (3) (7)</sup>    | C <sub>NRD</sub> = 0 pF   | $n \times t_{CPMCK} - 0.02$ <sup>(4)</sup>      |   | ns    |
|                   |   | C <sub>NRD</sub> derating | 0.002   |   | ns/pF |

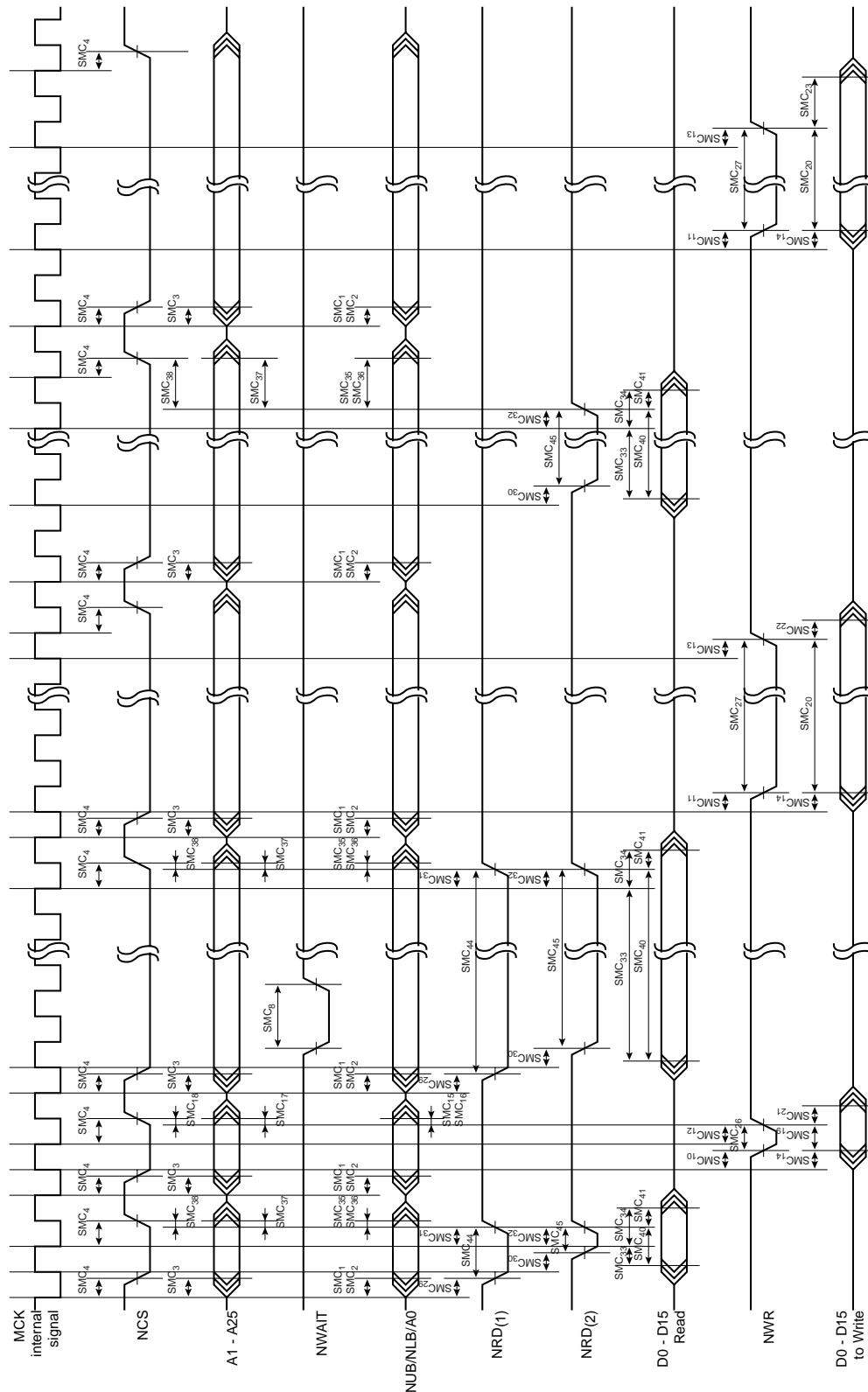
**Table 123.** SMC Read Signals (Continued)

| Symbol            | Parameter                                      | Conditions                | Min   | Max | Units |
|-------------------|--|---------------------------|---|-----|-------|
| SMC <sub>45</sub> | NRD Minimum Pulse Width <sup>(2) (3) (7)</sup> | C <sub>NRD</sub> = 0 pF   | $n \times t_{\text{CHMCK}} + t_{\text{CHMCK}} - 0.2$ <sup>(4)</sup>         |     | ns    |
|                   |  | C <sub>NRD</sub> derating | 0.002   |     | ns/pF |
| SMC <sub>46</sub> | NRD Minimum Pulse Width <sup>(2) (3)</sup>     | C <sub>NRD</sub> = 0 pF   | $((n + 1) \times t_{\text{CHMCK}}) + t_{\text{CHMCK}} - 0.2$ <sup>(4)</sup> |     | ns    |
|                   |  | C <sub>NRD</sub> derating | 0.002   |     | ns/pF |

- Notes:
1. Early Read Protocol.
  2. Standard Read Protocol.
  3. The derating factor is not to be applied to  $t_{\text{CHMCK}}$  or  $t_{\text{CPMCK}}$ .
  4.  $n$  = Number of standard Wait States inserted.
  5.  $n_{\text{acss}}$  = Number of Address to Chip Select Setup Cycles inserted.
  6.  $h$  = Number of Hold Cycles inserted.
  7. Not applicable when Address to Chip Select Setup Cycles are inserted.
  8. Only one of these two timings needs to be met.
  9. Only one of these two timings needs to be met.

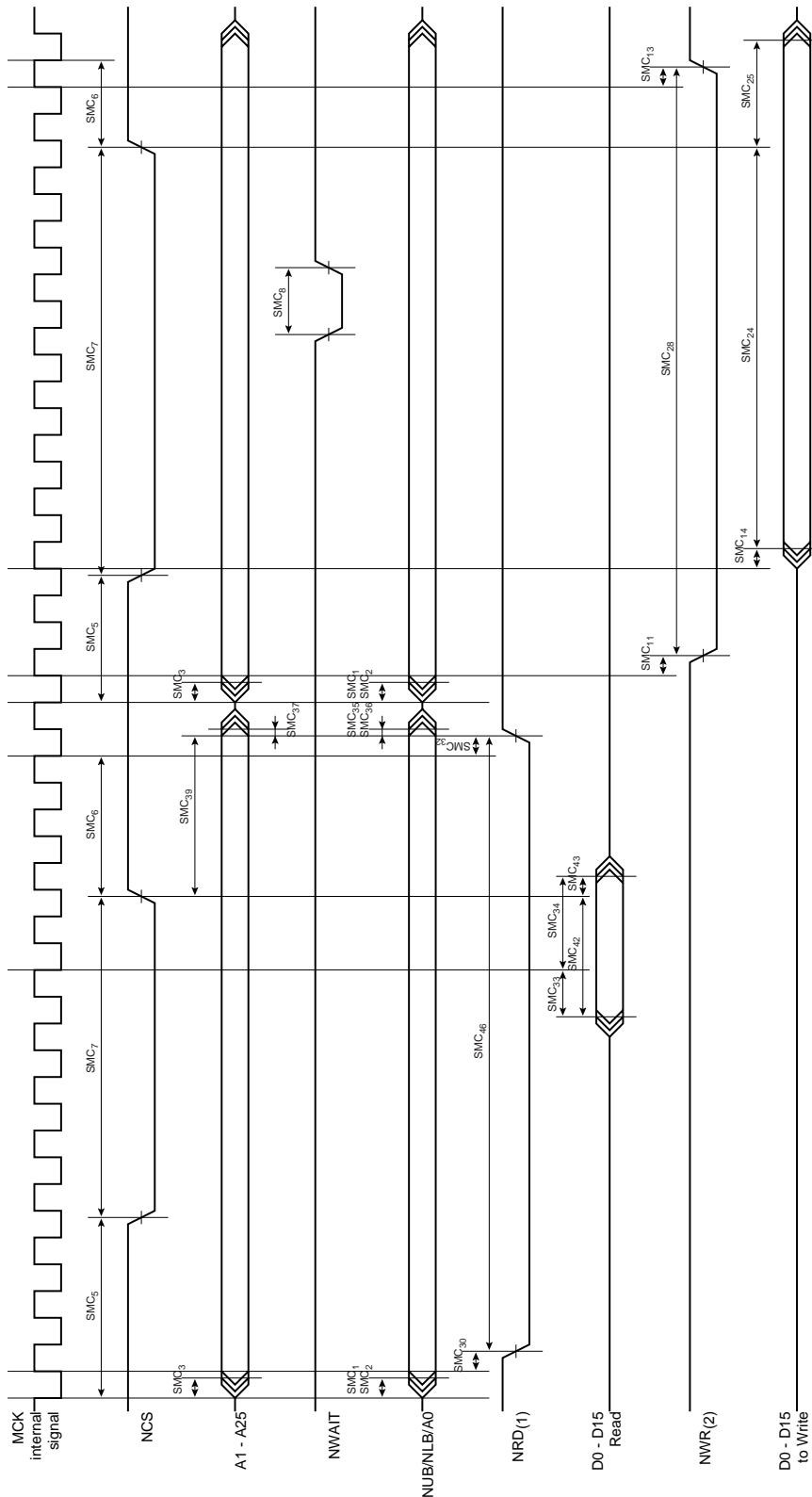


**Figure 270. SMC Signals Relative to MCK in Memory Interface Mode**



- Notes:
1. Early Read Protocol.
  2. Standard Read Protocol with or without Setup and Hold Cycles.

**Figure 271. SMC Signals Relative to MCK in LCD Interface Mode**



- Notes:
1. Standard Read Protocol only.
  2. With standard Wait States inserted only.

## SDRAMC Signals Relative to SDCK

Table 124 and Table 125 below show timings relative to operating condition limits defined in the section “Conditions and Timings Computation” on page 603.

**Table 124.** SDRAMC Clock Signal

| Symbol                  | Parameter                               | Conditions | Min  | Max  | Units |
|-------------------------|---|------------|------|------|-------|
| $1/(t_{CP\text{SDCK}})$ | SDRAM Controller Clock Frequency        |            |      | 80.0 | MHz   |
| $t_{CP\text{SDCK}}$     | SDRAM Controller Clock Period           |            | 12.5 |      | ns    |
| $t_{CH\text{SDCK}}$     | SDRAM Controller Clock High Half-Period |            | 5.6  |      | ns    |
| $t_{CL\text{SDCK}}$     | SDRAM Controller Clock Low Half-Period  |            | 6.9  |      | ns    |

**Table 125.** SDRAMC Signals

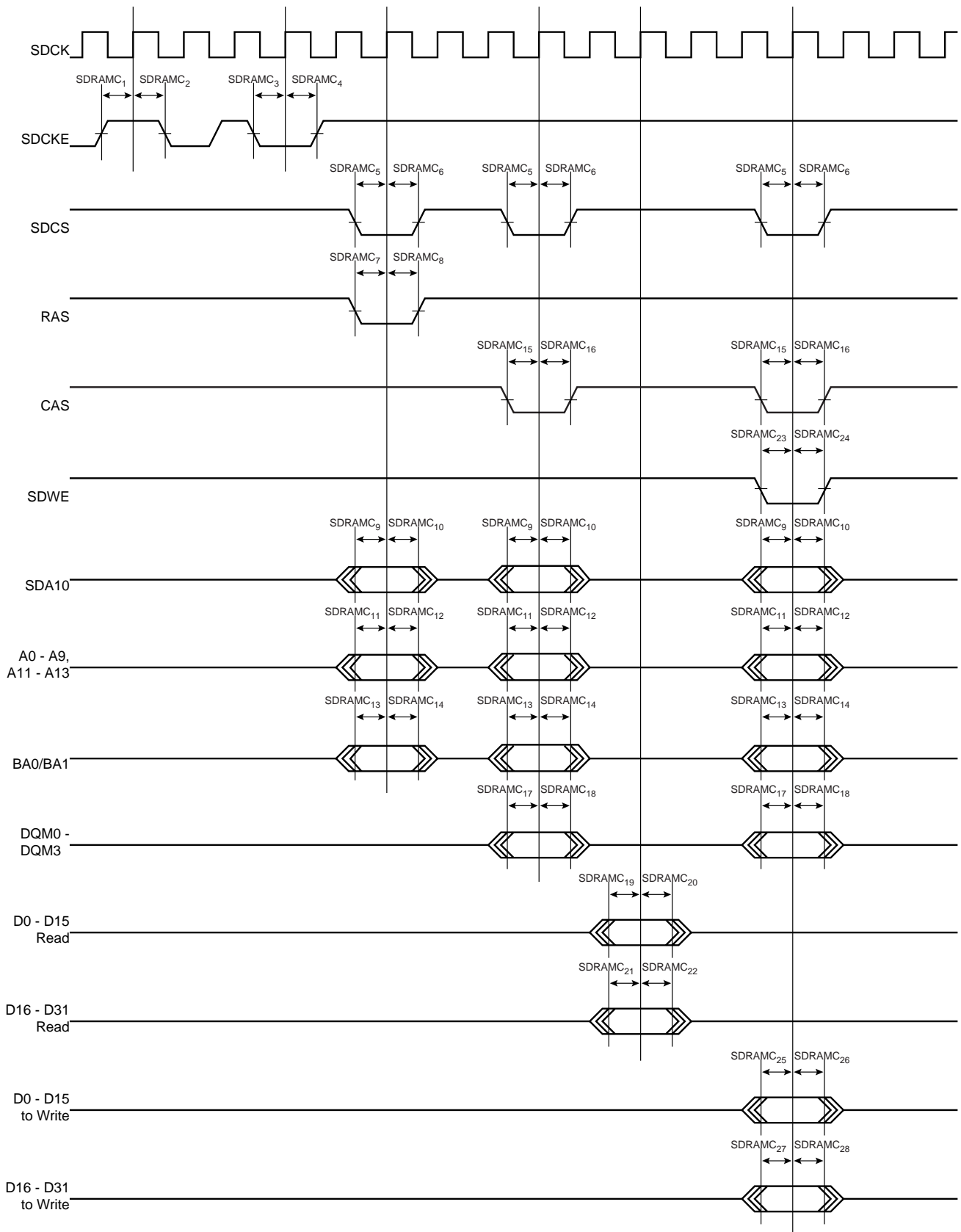
| Symbol               | Parameter   | Conditions                 | Min               | Max | Units |
|----------------------|---|----------------------------|-------------------|-----|-------|
| SDRAMC <sub>1</sub>  | SDCKE High before SDCK Rising Edge <sup>(1)</sup>     | $C_{SDCKE} = 0 \text{ pF}$ | $t_{CLMCK} + 1.2$ |     | ns    |
|                      |   | $C_{SDCKE}$ derating       | 0.015             |     | ns/pF |
| SDRAMC <sub>2</sub>  | SDCKE Low after SDCK Rising Edge <sup>(1)</sup>       | $C_{SDCKE} = 0 \text{ pF}$ | $t_{CHMCK} - 1.4$ |     | ns    |
|                      |   | $C_{SDCKE}$ derating       | - 0.023           |     | ns/pF |
| SDRAMC <sub>3</sub>  | SDCKE Low before SDCK Rising Edge <sup>(1)</sup>      | $C_{SDCKE} = 0 \text{ pF}$ | $t_{CLMCK} + 1.0$ |     | ns    |
|                      |   | $C_{SDCKE}$ derating       | 0.015             |     | ns/pF |
| SDRAMC <sub>4</sub>  | SDCKE High after SDCK Rising Edge <sup>(1)</sup>      | $C_{SDCKE} = 0 \text{ pF}$ | $t_{CHMCK} - 1.7$ |     | ns    |
|                      |   | $C_{SDCKE}$ derating       | - 0.023           |     | ns/pF |
| SDRAMC <sub>5</sub>  | SDCS Low before SDCK Rising Edge <sup>(1)</sup>       | $C_{SDCS} = 0 \text{ pF}$  | $t_{CLMCK} + 1.2$ |     | ns    |
|                      |   | $C_{SDCS}$ derating        | 0.028             |     | ns/pF |
| SDRAMC <sub>6</sub>  | SDCS High after SDCK Rising Edge <sup>(1)</sup>       | $C_{SDCS} = 0 \text{ pF}$  | $t_{CHMCK} - 1.9$ |     | ns    |
|                      |   | $C_{SDCS}$ derating        | - 0.045           |     | ns/pF |
| SDRAMC <sub>7</sub>  | RAS Low before SDCK Rising Edge <sup>(1)</sup>        | $C_{RAS} = 0 \text{ pF}$   | $t_{CLMCK} + 0.6$ |     | ns    |
|                      |   | $C_{RAS}$ derating         | 0.015             |     | ns/pF |
| SDRAMC <sub>8</sub>  | RAS High after SDCK Rising Edge <sup>(1)</sup>        | $C_{RAS} = 0 \text{ pF}$   | $t_{CHMCK} - 1.1$ |     | ns    |
|                      |   | $C_{RAS}$ derating         | - 0.023           |     | ns/pF |
| SDRAMC <sub>9</sub>  | SDA10 Change before SDCK Rising Edge <sup>(1)</sup>   | $C_{SDA10} = 0 \text{ pF}$ | $t_{CLMCK} + 0.8$ |     | ns    |
|                      |   | $C_{SDA10}$ derating       | 0.015             |     | ns/pF |
| SDRAMC <sub>10</sub> | SDA10 Change after SDCK Rising Edge <sup>(1)</sup>    | $C_{SDA10} = 0 \text{ pF}$ | $t_{CHMCK} - 1.2$ |     | ns    |
|                      |   | $C_{SDA10}$ derating       | - 0.023           |     | ns/pF |
| SDRAMC <sub>11</sub> | Address Change before SDCK Rising Edge <sup>(1)</sup> | $C_{ADD} = 0 \text{ pF}$   | $t_{CLMCK} + 0.6$ |     | ns    |
|                      |   | $C_{ADD}$ derating         | 0.028             |     | ns/pF |
| SDRAMC <sub>12</sub> | Address Change after SDCK Rising Edge <sup>(1)</sup>  | $C_{ADD} = 0 \text{ pF}$   | $t_{CHMCK} - 1.5$ |     | ns    |
|                      |   | $C_{ADD}$ derating         | - 0.045           |     | ns/pF |

**Table 125. SDRAMC Signals (Continued)**

| Symbol               | Parameter  | Conditions                 | Min                      | Max | Units |
|----------------------|--|----------------------------|--------------------------|-----|-------|
| SDRAMC <sub>13</sub> | Bank Change before SDCK Rising Edge <sup>(1)</sup> | C <sub>BA</sub> = 0 pF     | t <sub>CLMCK</sub> + 0.8 |     | ns    |
|                      |  | C <sub>BA</sub> derating   | 0.028                    |     | ns/pF |
| SDRAMC <sub>14</sub> | Bank Change after SDCK Rising Edge <sup>(1)</sup>  | C <sub>BA</sub> = 0 pF     | t <sub>CHMCK</sub> - 1.6 |     | ns    |
|                      |  | C <sub>BA</sub> derating   | - 0.045                  |     | ns/pF |
| SDRAMC <sub>15</sub> | CAS Low before SDCK Rising Edge <sup>(1)</sup>     | C <sub>CAS</sub> = 0 pF    | t <sub>CLMCK</sub> + 0.9 |     | ns    |
|                      |  | C <sub>CAS</sub> derating  | 0.015                    |     | ns/pF |
| SDRAMC <sub>16</sub> | CAS High after SDCK Rising Edge <sup>(1)</sup>     | C <sub>CAS</sub> = 0 pF    | t <sub>CHMCK</sub> - 1.5 |     | ns    |
|                      |  | C <sub>CAS</sub> derating  | - 0.023                  |     | ns/pF |
| SDRAMC <sub>17</sub> | DQM Change before SDCK Rising Edge <sup>(1)</sup>  | C <sub>DQM</sub> = 0 pF    | t <sub>CLMCK</sub> + 0.7 |     | ns    |
|                      |  | C <sub>DQM</sub> derating  | 0.028                    |     | ns/pF |
| SDRAMC <sub>18</sub> | DQM Change after SDCK Rising Edge <sup>(1)</sup>   | C <sub>DQM</sub> = 0 pF    | t <sub>CHMCK</sub> - 1.4 |     | ns    |
|                      |  | C <sub>DQM</sub> derating  | - 0.045                  |     | ns/pF |
| SDRAMC <sub>19</sub> | D0-D15 in Setup before SDCK Rising Edge            |                            | 1.3                      |     | ns    |
| SDRAMC <sub>20</sub> | D0-D15 in Hold after SDCK Rising Edge              |                            | 0.03                     |     | ns    |
| SDRAMC <sub>21</sub> | D16-D31 in Setup before SDCK Rising Edge           |                            | 2.0                      |     | ns    |
| SDRAMC <sub>22</sub> | D16-D31 in Hold after SDCK Rising Edge             |                            | -0.2                     |     | ns    |
| SDRAMC <sub>23</sub> | SDWE Low before SDCK Rising Edge                   | C <sub>SDWE</sub> = 0 pF   | t <sub>CLMCK</sub> + 1.0 |     | ns    |
|                      |  | C <sub>SDWE</sub> derating | 0.015                    |     | ns/pF |
| SDRAMC <sub>24</sub> | SDWE High after SDCK Rising Edge                   | C <sub>SDWE</sub> = 0 pF   | t <sub>CHMCK</sub> - 1.8 |     | ns    |
|                      |  | C <sub>SDWE</sub> derating | -0.023                   |     | ns/pF |
| SDRAMC <sub>25</sub> | D0-D15 Out Valid before SDCK Rising Edge           | C = 0 pF                   | t <sub>CLMCK</sub> - 2.7 |     | ns    |
|                      |  | C <sub>DATA</sub> derating | -0.044                   |     | ns/pF |
| SDRAMC <sub>26</sub> | D0-D15 Out Valid after SDCK Rising Edge            | C = 0 pF                   | t <sub>CHMCK</sub> - 2.4 |     | ns    |
|                      |  | C <sub>DATA</sub> derating | -0.044                   |     | ns/pF |
| SDRAMC <sub>27</sub> | D16-D31 Out Valid before SDCK Rising Edge          | C = 0 pF                   | t <sub>CLMCK</sub> - 3.2 |     | ns    |
|                      |  | C <sub>DATA</sub> derating | -0.044                   |     | ns/pF |
| SDRAMC <sub>28</sub> | D16-D31 Out Valid after SDCK Rising Edge           | C = 0 pF                   | t <sub>CHMCK</sub> - 2.4 |     | ns    |
|                      |  | C <sub>DATA</sub> derating | -0.044                   |     | ns/pF |

Note: 1. The derating factor is not to be applied to t<sub>CLMCK</sub> or t<sub>CHMCK</sub>.

Figure 272. SDRAMC Signals Relative to SDCK



## BFC Signals Relative to BFCK

Table 126, Table 127 and Table 128 show timings relative to operating condition limits defined in the section “Conditions and Timings Computation” on page 603.

**Table 126.** BFC Clock Signal

| Symbol           | Parameter                            | Conditions                   | Min  | Max  | Units |
|------------------|--------------------------------------|------------------------------|------|------|-------|
| $1/(t_{CPBFCK})$ | BF Controller Clock Frequency        | BFCK is MCK <sup>(1)</sup>   |      | 80.0 | MHz   |
|                  |                                      | BFCK is MCK/2 <sup>(2)</sup> |      | 40.0 | MHz   |
|                  |                                      | BFCK is MCK/4 <sup>(3)</sup> |      | 20.0 | MHz   |
| $t_{CPBFCK}$     | BF Controller Clock Period           | BFCK is MCK <sup>(1)</sup>   | 12.5 |      | ns    |
|                  |                                      | BFCK is MCK/2 <sup>(2)</sup> | 25.0 |      | ns    |
|                  |                                      | BFCK is MCK/4 <sup>(3)</sup> | 50.0 |      | ns    |
| $t_{CHBFCK}$     | BF Controller Clock High Half-Period | BFCK is MCK <sup>(1)</sup>   | 6.5  |      | ns    |
|                  |                                      | BFCK is MCK/2 <sup>(2)</sup> | 12.8 |      | ns    |
|                  |                                      | BFCK is MCK/4 <sup>(3)</sup> | 25.3 |      | ns    |
| $t_{CLBFCK}$     | BF Controller Clock Low Half-Period  | BFCK is MCK <sup>(1)</sup>   | 6.1  |      | ns    |
|                  |                                      | BFCK is MCK/2 <sup>(2)</sup> | 12.3 |      | ns    |
|                  |                                      | BFCK is MCK/4 <sup>(3)</sup> | 24.8 |      | ns    |

- Notes:
1. Field BFCC = 1 in Register BFC\_MR, see “Burst Flash Controller Mode Register” on page 221.
  2. Field BFCC = 2 in Register BFC\_MR, see “Burst Flash Controller Mode Register” on page 221.
  3. Field BFCC = 3 in Register BFC\_MR, see “Burst Flash Controller Mode Register” on page 221.

**Table 127.** BFC Signals in Asynchronous Mode

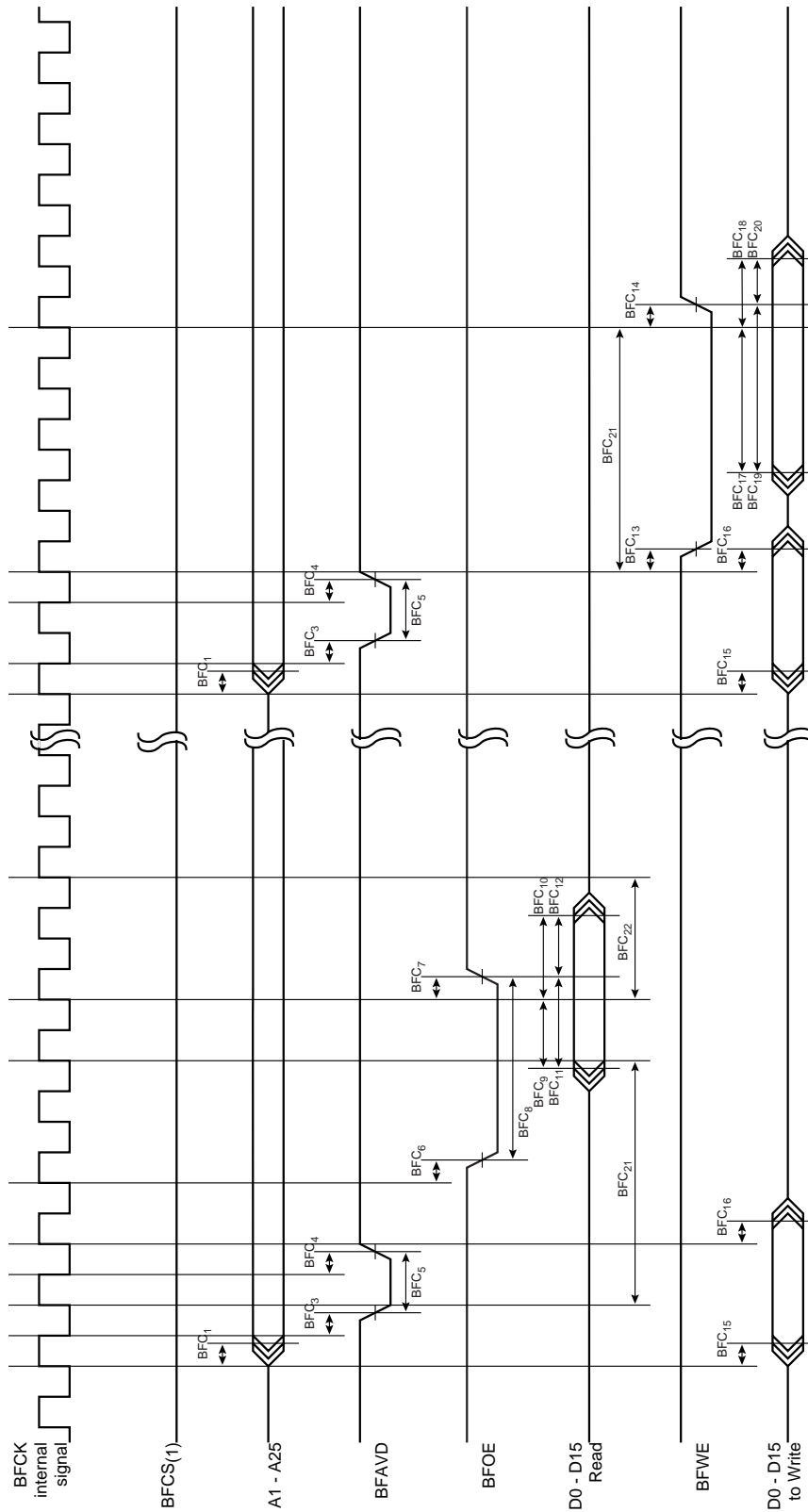
| Symbol           | Parameter                                     | Conditions           | Min                | Max                | Units |
|------------------|---|----------------------|--------------------|--------------------|-------|
| BFC <sub>1</sub> | BFCK Rising to A1-A25 Valid <sup>(1)</sup>    | $C_{ADD} = 0$ pF     |                    | $t_{CLBFCK} - 0.2$ | ns    |
|                  |   | $C_{ADD}$ derating   |                    | - 0.028            | ns/pF |
| BFC <sub>2</sub> | BFCK Rising to A1-A25 Change <sup>(1)</sup>   | $C_{ADD} = 0$ pF     | $t_{CLBFCK} - 1.0$ |                    | ns    |
|                  |   | $C_{ADD}$ derating   | - 0.045            |                    | ns/pF |
| BFC <sub>3</sub> | BFCK Falling to BFAVD Active <sup>(1)</sup>   | $C_{BFAVD} = 0$ pF   | $t_{CLBFCK} - 1.1$ | $t_{CLBFCK} - 0.3$ | ns    |
|                  |   | $C_{BFAVD}$ derating | - 0.044            | - 0.028            | ns/pF |
| BFC <sub>4</sub> | BFCK Falling to BFAVD Inactive <sup>(1)</sup> | $C_{BFAVD} = 0$ pF   | $t_{CLBFCK} - 1.8$ | $t_{CLBFCK} + 0.2$ | ns    |
|                  |   | $C_{BFAVD}$ derating | - 0.044            | 0.044              | ns/pF |
| BFC <sub>5</sub> | BFAVD Minimum Pulse Width <sup>(1)</sup>      | $C_{BFAVD} = 0$ pF   | $t_{CPBFCK} + 1.0$ |                    | ns    |
|                  |   | $C_{BFAVD}$ derating | 0.001              |                    | ns/pF |
| BFC <sub>6</sub> | BFCK Rising to BFOE Active                    | $C_{BFOE} = 0$ pF    | - 0.4              | 0.1                | ns    |
|                  |   | $C_{BFOE}$ derating  | - 0.044            | 0.044              | ns/pF |
| BFC <sub>7</sub> | BFCK Rising to BFOE Inactive                  | $C_{BFOE} = 0$ pF    | - 1.1              | 0.7                | ns    |
|                  |   | $C_{BFOE}$ derating  | - 0.044            | 0.044              | ns/pF |

**Table 127. BFC Signals in Asynchronous Mode (Continued)**

| Symbol            | Parameter  | Conditions                  | Min  | Max  | Units |
|-------------------|--|-----------------------------|--|--|-------|
| BFC <sub>8</sub>  | BFOE Minimum Pulse Width <sup>(1)</sup>                | C <sub>BFOE</sub> = 0 pF    | $(a \times t_{CPBFCK}) + 0.9$ <sup>(2)</sup> |  | ns    |
|                   |  | C <sub>BFOE</sub> derating  | 0.028  |  | ns/pF |
| BFC <sub>9</sub>  | D0-D15 in Setup before BFCK Rising Edge <sup>(5)</sup> |                             | - 0.1  |  | ns    |
| BFC <sub>10</sub> | D0-D15 in Hold after BFCK Rising Edge <sup>(6)</sup>   |                             | 1.0  |  | ns    |
| BFC <sub>11</sub> | Data Setup before BFOE High <sup>(5)</sup>             | C <sub>BFOE</sub> = 0 pF    | - 0.9  |  | ns    |
|                   |  | C <sub>BFOE</sub> derating  | - 0.044                                      |  | ns/pF |
| BFC <sub>12</sub> | Data Hold after BFOE High <sup>(6)</sup>               | C <sub>BFOE</sub> = 0 pF    | 2.0  |  | ns    |
|                   |  | C <sub>BFOE</sub> derating  | 0.028  |  | ns/pF |
| BFC <sub>13</sub> | BFCK Rising to BFW E Active                            | C <sub>BFW E</sub> = 0 pF   | - 0.6  | - 0.05                                       | ns    |
|                   |  | C <sub>BFW E</sub> derating | - 0.044                                      | - 0.028                                      | ns/pF |
| BFC <sub>14</sub> | BFCK Rising to BFW E Inactive                          | C <sub>BFW E</sub> = 0 pF   | - 1.3  | 0.5  | ns    |
|                   |  | C <sub>BFW E</sub> derating | - 0.044                                      | 0.044  | ns/pF |
| BFC <sub>15</sub> | BFCK Rising to AD0-AD15 Valid <sup>(1) (4)</sup>       | C <sub>DATA</sub> = 0 pF    |  | t <sub>CLBFCK</sub> - 0.2                    | ns    |
|                   |  | C <sub>DATA</sub> derating  |  | - 0.028                                      | ns/pF |
| BFC <sub>16</sub> | BFCK Rising to AD0-AD15 Not Valid <sup>(1) (4)</sup>   | C <sub>DATA</sub> = 0 pF    | t <sub>CLBFCK</sub> - 0.8                    |  | ns    |
|                   |  | C <sub>DATA</sub> derating  | - 0.044                                      |  | ns/pF |
| BFC <sub>17</sub> | Data Out Valid before BFCK Rising <sup>(1) (5)</sup>   | C <sub>DATA</sub> = 0 pF    | t <sub>CLBFCK</sub> + 0.5                    |  | ns    |
|                   |  | C <sub>DATA</sub> derating  | 0.028  |  | ns/pF |
| BFC <sub>18</sub> | Data Out Valid after BFCK Rising <sup>(1) (6)</sup>    | C <sub>DATA</sub> = 0 pF    | t <sub>CHBFCK</sub> + 0.7                    |  | ns    |
|                   |  | C <sub>DATA</sub> derating  | 0.028  |  | ns/pF |
| BFC <sub>19</sub> | Data Out Valid before BFW E High <sup>(1) (5)</sup>    | C = 0 pF                    | t <sub>CLBFCK</sub> - 0.5                    |  | ns    |
|                   |  | C <sub>DATA</sub> derating  | - 0.028                                      |  | ns/pF |
|                   |  | C <sub>BFW E</sub> derating | 0.044  |  | ns/pF |
| BFC <sub>20</sub> | Data Out Valid after BFW E High <sup>(1) (6)</sup>     | C = 0 pF                    | t <sub>CHBFCK</sub> + 0.3                    |  | ns    |
|                   |  | C <sub>DATA</sub> derating  | 0.028  |  | ns/pF |
|                   |  | C <sub>BFW E</sub> derating | - 0.044                                      |  | ns/pF |
| BFC <sub>21</sub> | Number of Address Valid Latency Cycles <sup>(1)</sup>  |                             | $((a + 1) \times t_{CPBFCK})$ <sup>(2)</sup> | $((a + 1) \times t_{CPBFCK})$ <sup>(2)</sup> | ns    |
| BFC <sub>22</sub> | Number of Output Enable Latency Cycles <sup>(1)</sup>  |                             | $(o \times t_{CPBFCK})$ <sup>(3)</sup>       | $(o \times t_{CPBFCK})$ <sup>(3)</sup>       | ns    |

- Notes:
1. The derating factor is not to be applied to t<sub>CPBFCK</sub>.
  2. a = Number of Address Valid Latency Cycles defined in the BFC\_MR AVL field.
  3. o = Number of Output Enable Latency Cycles defined in the BFC\_MR OEL field.
  4. Applicable only with multiplexed Address and Data Buses.
  5. Only one of these two timings needs to be met.
  6. Only one of these two timings needs to be met.

**Figure 273. BFC Signals Relative to BFCK in Asynchronous Mode**



Note: 1. BFCS is asserted as soon as the BFCOM field in BFC\_MR is different from 0.



**Table 128. BFC Signals in Burst Mode**

| Symbol            | Parameter   | Conditions                  | Min                                 | Max                                 | Units |
|-------------------|---|-----------------------------|-------------------------------------|-------------------------------------|-------|
| BFC <sub>1</sub>  | BFCK Rising to A1-A25 Valid <sup>(1)</sup>                      | C <sub>ADD</sub> = 0 pF     |                                     | t <sub>CLBFCK</sub> - 0.2           | ns    |
|                   |   | C <sub>ADD</sub> derating   |                                     | - 0.028                             | ns/pF |
| BFC <sub>2</sub>  | BFCK Rising to A1-A25 Change <sup>(1)</sup>                     | C <sub>ADD</sub> = 0 pF     | t <sub>CLBFCK</sub> - 1.0           |                                     | ns    |
|                   |   | C <sub>ADD</sub> derating   | - 0.045                             |                                     | ns/pF |
| BFC <sub>3</sub>  | BFCK Falling to BFAVD Active <sup>(1)</sup>                     | C <sub>BFAVD</sub> = 0 pF   | t <sub>CLBFCK</sub> - 1.1           | t <sub>CLBFCK</sub> - 0.3           | ns    |
|                   |   | C <sub>BFAVD</sub> derating | - 0.044                             | - 0.028                             | ns/pF |
| BFC <sub>4</sub>  | BFCK Falling to BFAVD Inactive <sup>(1)</sup>                   | C <sub>BFAVD</sub> = 0 pF   | t <sub>CLBFCK</sub> - 1.8           | t <sub>CLBFCK</sub> + 0.2           | ns    |
|                   |   | C <sub>BFAVD</sub> derating | - 0.044                             | 0.044                               | ns/pF |
| BFC <sub>5</sub>  | BFAVD Minimum Pulse Width <sup>(1)</sup>                        | C <sub>BFAVD</sub> = 0 pF   | t <sub>CPBFCK</sub> + 1.0           |                                     | ns    |
|                   |   | C <sub>BFAVD</sub> derating | 0.001                               |                                     | ns/pF |
| BFC <sub>6</sub>  | BFCK Rising to BFOE Active                                      | C <sub>BFOE</sub> = 0 pF    | - 0.4                               | 0.1                                 | ns    |
|                   |   | C <sub>BFOE</sub> derating  | - 0.044                             | 0.044                               | ns/pF |
| BFC <sub>7</sub>  | BFCK Rising to BFOE Inactive                                    | C <sub>BFOE</sub> = 0 pF    | - 1.1                               | 0.7                                 | ns    |
|                   |   | C <sub>BFOE</sub> derating  | - 0.044                             | 0.044                               | ns/pF |
| BFC <sub>9</sub>  | D0-D15 in Setup before BFCK Rising Edge                         |                             | - 0.1                               |                                     | ns    |
| BFC <sub>10</sub> | D0-D15 in Hold after BFCK Rising Edge                           |                             | 1.0                                 |                                     | ns    |
| BFC <sub>15</sub> | BFCK Rising to AD0-AD15 Valid <sup>(1)</sup> <sup>(4)</sup>     | C <sub>DATA</sub> = 0 pF    |                                     | t <sub>CLBFCK</sub> - 0.2           | ns    |
|                   |   | C <sub>DATA</sub> derating  |                                     | - 0.028                             | ns/pF |
| BFC <sub>16</sub> | BFCK Rising to AD0-AD15 Not Valid <sup>(1)</sup> <sup>(4)</sup> | C <sub>DATA</sub> = 0 pF    | t <sub>CLBFCK</sub> - 0.8           |                                     | ns    |
|                   |   | C <sub>DATA</sub> derating  | - 0.044                             |                                     | ns/pF |
| BFC <sub>21</sub> | Number of Address Valid Latency Cycles <sup>(1)</sup>           |                             | $((a + 1) \times t_{CPBFCK})^{(2)}$ | $((a + 1) \times t_{CPBFCK})^{(2)}$ | ns    |
| BFC <sub>22</sub> | Number of Output Enable Latency Cycles <sup>(1)</sup>           |                             | $(o \times t_{CPBFCK})_{(3)}$       | $(o \times t_{CPBFCK})_{(3)}$       | ns    |
| BFC <sub>23</sub> | BFCK Falling to BFBAA Active <sup>(1)</sup>                     | C <sub>BFBAA</sub> = 0 pF   | t <sub>CLBFCK</sub> - 1.0           | t <sub>CLBFCK</sub> - 0.1           | ns    |
|                   |   | C <sub>BFBAA</sub> derating | - 0.044                             | - 0.028                             | ns/pF |
| BFC <sub>24</sub> | BFCK Falling to BFBAA Inactive <sup>(1)</sup>                   | C <sub>BFBAA</sub> = 0 pF   | t <sub>CLBFCK</sub> - 1.7           | t <sub>CLBFCK</sub> + 0.1           | ns    |
|                   |   | C <sub>BFBAA</sub> derating | - 0.044                             | 0.044                               | ns/pF |
| BFC <sub>25</sub> | BFRDY Change Hold after BFCK Rising Edge                        |                             | 0.1                                 |                                     | ns    |
| BFC <sub>26</sub> | BFRDY Change Setup before BFCK Rising Edge                      |                             | 0.3                                 |                                     | ns    |

- Notes:
1. The derating factor is not to be applied to t<sub>CPBFCK</sub>.
  2. a = Number of Address Valid Latency Cycles defined in the BFC\_MR AVL field.
  3. o = Number of Output Enable Latency Cycles defined in the BFC\_MR OEL field.
  4. Applicable only with multiplexed Address and Data Buses.



## JTAG/ICE Timings

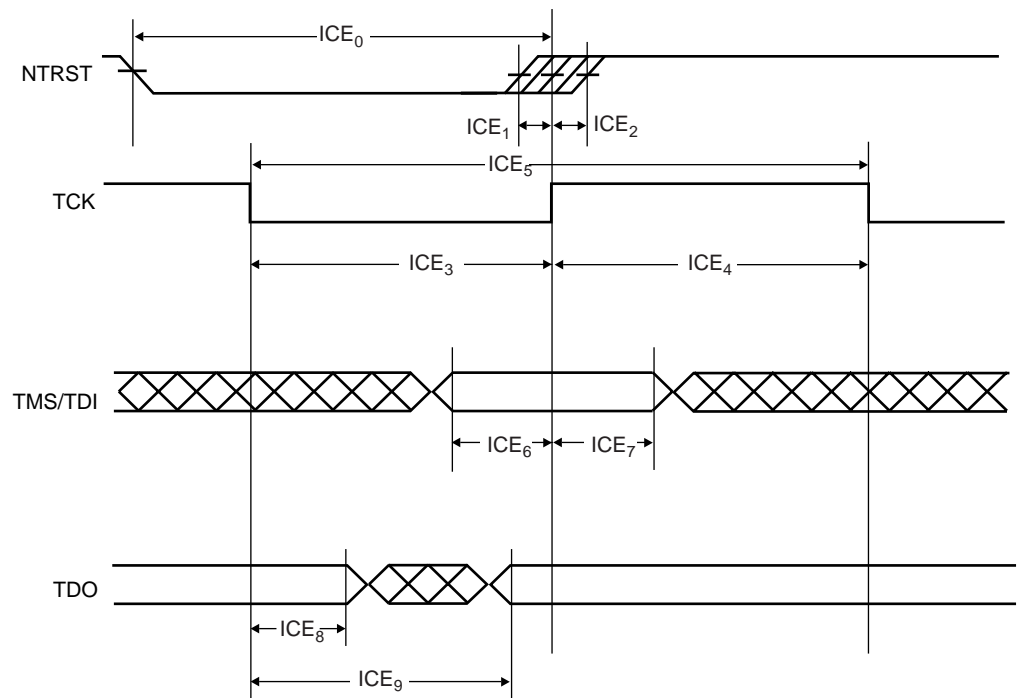
### ICE Interface Signals

Table 129 shows timings relative to operating condition limits defined in the section “Conditions and Timings Computation” on page 603

**Table 129.** ICE Interface Timing Specifications

| Symbol           | Parameter                        | Conditions                | Min   | Max   | Units |
|------------------|----------------------------------|---------------------------|-------|-------|-------|
| ICE <sub>0</sub> | NTRST Minimum Pulse Width        |                           | 20.00 |       | ns    |
| ICE <sub>1</sub> | NTRST High Recovery to TCK High  |                           | 0.86  |       | ns    |
| ICE <sub>2</sub> | NTRST High Removal from TCK High |                           | 0.90  |       | ns    |
| ICE <sub>3</sub> | TCK Low Half-period              |                           | 8.00  |       | ns    |
| ICE <sub>4</sub> | TCK High Half-period             |                           | 8.00  |       | ns    |
| ICE <sub>5</sub> | TCK Period                       |                           | 20.00 |       | ns    |
| ICE <sub>6</sub> | TDI, TMS, Setup before TCK High  |                           | -0.13 |       | ns    |
| ICE <sub>7</sub> | TDI, TMS, Hold after TCK High    |                           | 0.10  |       | ns    |
| ICE <sub>8</sub> | TDO Hold Time                    | C <sub>TDO</sub> = 0 pF   | 4.17  |       | ns    |
|                  |                                  | C <sub>TDO</sub> derating | 0     |       | ns/pF |
| ICE <sub>9</sub> | TCK Low to TDO Valid             | C <sub>TDO</sub> = 0 pF   |       | 6.49  | ns    |
|                  |                                  | C <sub>TDO</sub> derating |       | 0.028 | ns/pF |

**Figure 275.** ICE Interface Signals



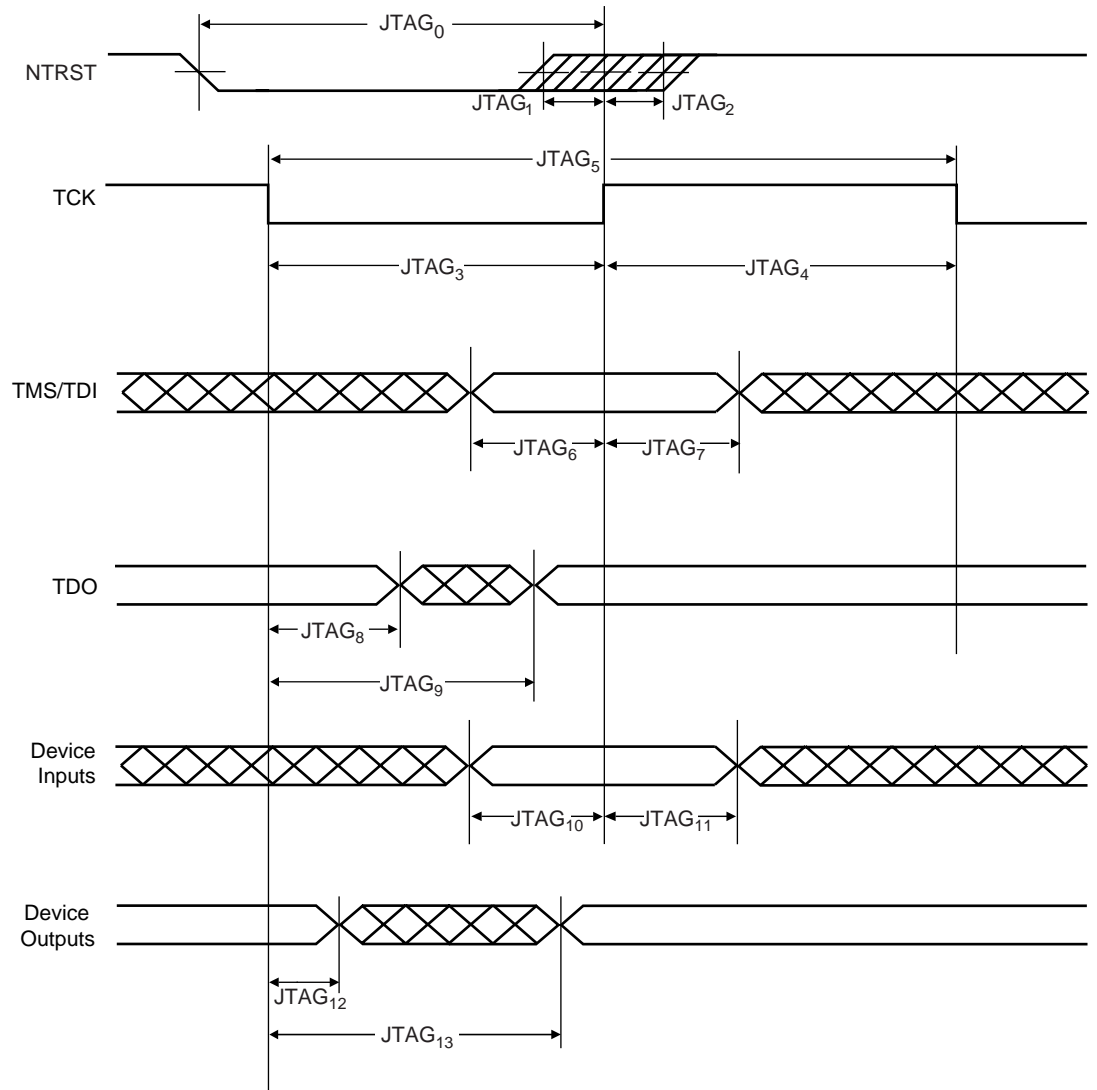
## JTAG Interface Signals

Table 130 shows timings relative to operating condition limits defined in the section “Conditions and Timings Computation” on page 603

**Table 130.** JTAG Interface Timing Specifications

| Symbol             | Parameter                        | Conditions                | Min   | Max   | Units |
|--------------------|----------------------------------|---------------------------|-------|-------|-------|
| JTAG <sub>0</sub>  | NTRST Minimum Pulse Width        |                           | 20.00 |       | ns    |
| JTAG <sub>1</sub>  | NTRST High Recovery to TCK High  |                           | -0.16 |       | ns    |
| JTAG <sub>2</sub>  | NTRST High Recovery to TCK Low   |                           | -0.16 |       | ns    |
| JTAG <sub>3</sub>  | NTRST High Removal from TCK High |                           | -0.07 |       | ns    |
| JTAG <sub>4</sub>  | NTRST High Removal from TCK Low  |                           | -0.07 |       | ns    |
| JTAG <sub>5</sub>  | TCK Low Half-period              |                           | 8.00  |       | ns    |
| JTAG <sub>6</sub>  | TCK High Half-period             |                           | 8.00  |       | ns    |
| JTAG <sub>7</sub>  | TCK Period                       |                           | 20.00 |       | ns    |
| JTAG <sub>8</sub>  | TDI, TMS Setup before TCK High   |                           | 0.01  |       | ns    |
| JTAG <sub>9</sub>  | TDI, TMS Hold after TCK High     |                           | 3.21  |       | ns    |
| JTAG <sub>10</sub> | TDO Hold Time                    | C <sub>TDO</sub> = 0 pF   | 2.38  |       | ns    |
|                    |                                  | C <sub>TDO</sub> derating | 0     |       | ns/pF |
| JTAG <sub>11</sub> | TCK Low to TDO Valid             | C <sub>TDO</sub> = 0 pF   |       | 4.66  | ns    |
|                    |                                  | C <sub>TDO</sub> derating |       | 0.028 | ns/pF |
| JTAG <sub>12</sub> | Device Inputs Setup Time         |                           | -1.23 |       | ns    |
| JTAG <sub>13</sub> | Device Inputs Hold Time          |                           | 3.81  |       | ns    |
| JTAG <sub>14</sub> | Device Outputs Hold Time         | C <sub>OUT</sub> = 0 pF   | 7.15  |       | ns    |
|                    |                                  | C <sub>OUT</sub> derating | 0     |       | ns/pF |
| JTAG <sub>15</sub> | TCK to Device Outputs Valid      | C <sub>OUT</sub> = 0 pF   |       | 7.22  | ns    |
|                    |                                  | C <sub>OUT</sub> derating |       | 0.028 | ns/pF |

Figure 276. JTAG Interface Signals



# ETM Timings

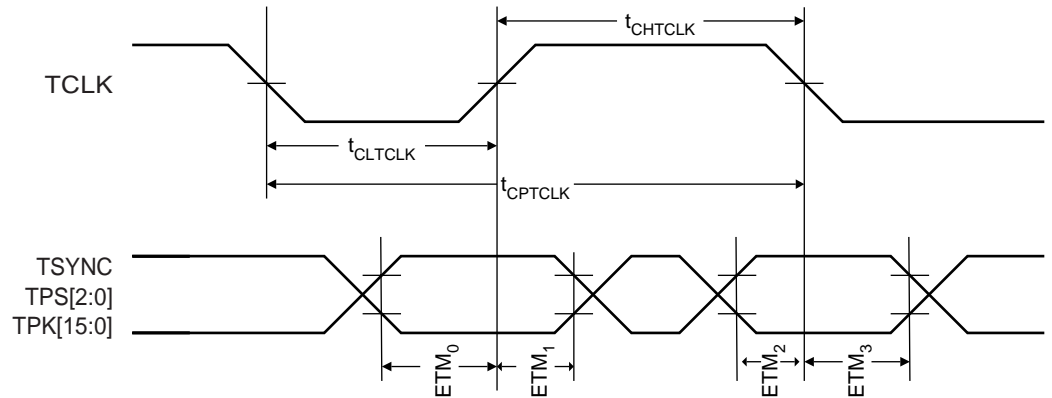
## Timings Data

Table 131 shows timings relative to operating condition limits defined in the section “Conditions and Timings Computation” on page 603.

**Table 131.** ETM Timing Characteristics

| Symbol           | Parameter                                       | Conditions                 | Min                   | Typ                      | Max   | Units |
|------------------|---|----------------------------|-----------------------|--------------------------|-------|-------|
| $1/(t_{CPTCLK})$ | Trace Clock Frequency                           |                            |                       | $1/(2 \times t_{CPPCK})$ | 86.54 | MHz   |
| $t_{CPTCLK}$     | Trace Clock Period                              |                            | 11.56                 | $2 \times t_{CPPCK}$     |       | ns    |
| $t_{CHTCLK}$     | TCLK High Half-period                           |                            | $t_{CPTCLK}/2 + 0.02$ |                          |       | ns    |
| $t_{CLTCLK}$     | TCLK Low Half-period                            |                            | $t_{CPTCLK}/2 - 0.02$ |                          |       | ns    |
| ETM <sub>0</sub> | Data Signals Out Valid before TCLK Rising Edge  | C = 0 pF                   | $t_{CLTCLK} - 1.06$   |                          |       | ns    |
|                  |   | C <sub>DATA</sub> derating | 0.044                 |                          |       | ns/pF |
| ETM <sub>1</sub> | Data Signals Out Valid after TCLK Rising Edge   | C = 0 pF                   | $t_{CHTCLK} - 0.49$   |                          |       | ns    |
|                  |   | C <sub>DATA</sub> derating | 0.044                 |                          |       | ns/pF |
| ETM <sub>2</sub> | Data Signals Out Valid before TCLK Falling Edge | C = 0 pF                   | $t_{CHTCLK} - 1.03$   |                          |       | ns    |
|                  |   | C <sub>DATA</sub> derating | 0.044                 |                          |       | ns/pF |
| ETM <sub>3</sub> | Data Signals Out Valid after TCLK Falling Edge  | C = 0 pF                   | $t_{CLTCLK} - 0.51$   |                          |       | ns    |
|                  |   | C <sub>DATA</sub> derating | 0.044                 |                          |       | ns/pF |

**Figure 277.** ETM Signals



## Design Considerations

When designing a PCB, it is important to keep the differences between trace length of ETM signals as small as possible to minimize skew between them. In addition, crosstalk on the trace port must be kept to a minimum as it can cause erroneous trace results. Stubs on these traces can cause unpredictable responses, thus it is recommended to avoid stubs on the trace lines.

The TCLK line should be series-terminated as close as possible to the microcontroller pins.

The maximum capacitance presented by the trace connector, cabling and interfacing logic must be less than 15 pF.

## AT91RM9200 Mechanical Characteristics

### Thermal and Reliability Considerations

#### Thermal Data

In Table 132, the device lifetime is estimated using the MIL-217 standard in the “moderately controlled” environmental model (this model is described as corresponding to an installation in a permanent rack with adequate cooling air), depending on the device Junction Temperature. (For details see the section “Junction Temperature” on page 626.)

Note that the user must be extremely cautious with this MTBF calculation. It should be noted that the MIL-217 model is pessimistic with respect to observed values due to the way the data/models are obtained (test under severe conditions). The life test results that have been measured are always better than the predicted ones.

**Table 132.** MTBF Versus Junction Temperature

| Junction Temperature (T <sub>J</sub> ) (°C) | Estimated Lifetime (MTBF) (Year) |
|---|----------------------------------|
| 100   | 6                                |
| 125   | 3                                |
| 150   | 2                                |
| 175   | 1                                |

Table 133 summarizes the thermal resistance data depending on the package.

**Table 133.** Thermal Resistance Data

| Symbol          | Parameter                              | Condition | Package  | Typ  | Unit |
|-----------------|--|-----------|----------|------|------|
| θ <sub>JA</sub> | Junction-to-ambient thermal resistance | Still Air | PQFP208  | 33.9 | °C/W |
|                 |  |           | LFBGA256 | 35.6 |      |
| θ <sub>JC</sub> | Junction-to-case thermal resistance    |           | PQFP208  | 15.7 |      |
|                 |  |           | LFBGA256 | 7.7  |      |

#### Reliability Data

The number of gates and the device die size are provided Table 134 so that the user can calculate reliability data for another standard and/or in another environmental model.

**Table 134.** Reliability Data

| Parameter              | Data | Unit            |
|------------------------|------|-----------------|
| Number of Logic Gates  | 4461 | K gates         |
| Number of Memory Gates | 2458 | K gates         |
| Device Die Size        | 33.9 | mm <sup>2</sup> |

## Junction Temperature

The average chip-junction temperature,  $T_J$ , in °C can be obtained from the following:

1.  $T_J = T_A + (P_D \times \theta_{JA})$
2.  $T_J = T_A + (P_D \times (\theta_{HEATSINK} + \theta_{JC}))$

where:

- $\theta_{JA}$  = package thermal resistance, Junction-to-ambient (°C/W), provided in Table 133 on page 625.
- $\theta_{JC}$  = package thermal resistance, Junction-to-case thermal resistance (°C/W), provided in Table 133 on page 625.
- $\theta_{HEAT\ SINK}$  = cooling device thermal resistance (°C/W), provided in the device datasheet.
- $P_D$  = device power consumption (W) estimated from data provided in the section “Power Consumption” on page 598.
- $T_A$  = ambient temperature (°C).

From the first equation, the user can derive the estimated lifetime of the chip and decide if a cooling device is necessary or not. If a cooling device is to be fitted on the chip, the second equation should be used to compute the resulting average chip-junction temperature  $T_J$  in °C.



### Package Drawings

Figure 278. 208-lead PQFP Package Drawing

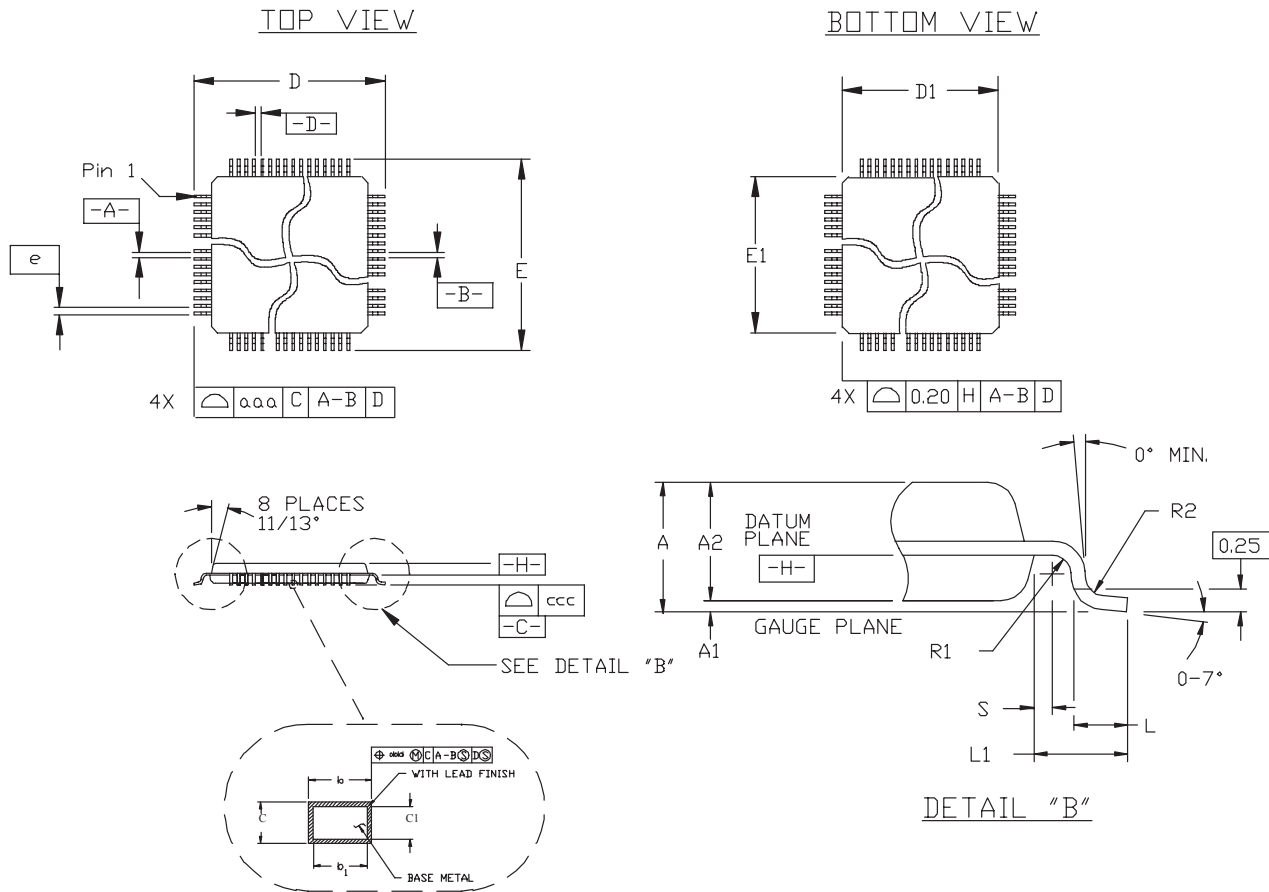
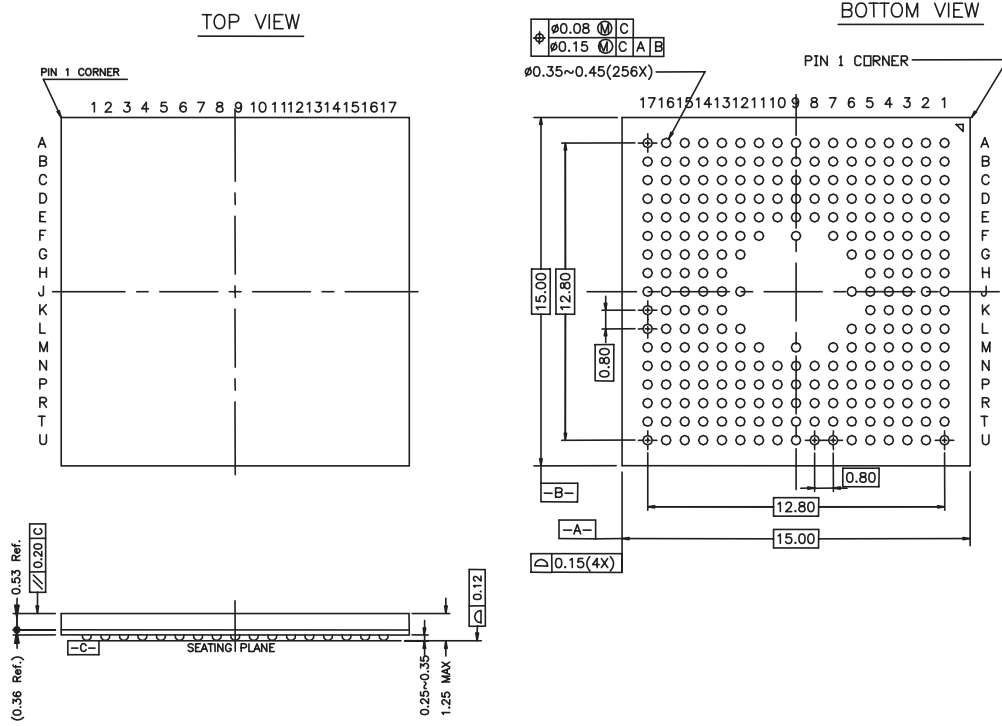


Table 135. 208-lead PQFP Package Dimensions (in mm)

| Symbol | Min      | Nom  | Max  | Symbol                                 | Min  | Nom   | Max  |
|--------|----------|------|------|--|------|-------|------|
| c      | 0.11     |      | 0.23 | b1                                     | 0.17 | 0.20  | 0.23 |
| c1     | 0.11     | 0.15 | 0.19 | ddd                                    | 0.10 |       |      |
| L      | 0.65     | 0.88 | 1.03 | <b>Tolerances of Form and Position</b> |      |       |      |
| L1     | 1.60 REF |      |      | aaa                                    |      | 0.25  |      |
| R2     | 0.13     |      | 0.3  | ccc                                    |      |       | 0.1  |
| R1     | 0.13     |      |      | <b>BSC</b>                             |      |       |      |
| S      | 0.4      |      |      | D                                      |      | 31.20 |      |
| A      | 4.10     |      |      | D1                                     |      | 28.00 |      |
| A1     | 0.25     |      | 0.50 | E                                      |      | 31.20 |      |
| A2     | 3.20     | 3.40 | 3.60 | E1                                     |      | 28.00 |      |
| b      | 0.17     |      | 0.27 | e                                      |      | 0.50  |      |

Figure 279. 256-ball BGA Package Drawing



## AT91RM9200 Ordering Information

**Table 136.** Ordering Information

| Ordering Code     | Package  | ROM Code Revision | Temperature Operating Range   |
|-------------------|----------|-------------------|-------------------------------|
| AT91RM9200-QI-002 | PQFP 208 | 002               | Industrial<br>(-40°C to 85°C) |
| AT91RM9200-CI-002 | BGA 256  |                   |                               |



## Document Details

**Title** AT91RM9200 Datasheet

**Literature Number** 1768

## Revision History

**Version A** **Publication Date:** 22-Apr-03

**Version B** **Publication Date:** 22-Aug-03

### *Changes since last issue*

|                 |   |
|-----------------|---|
| <i>Page 36</i>  | New Figure 8, ARM920T Internal Functional Block Diagram.  |
| <i>Page 56</i>  | Corrected fields in CP15 Register 7 register table.   |
| <i>Page 62</i>  | Updated Figure 9, AT91RM9200 Debug and Test Block Diagram with corrected DTXD and DRXD signal names and transfer direction of signals TST0 - TST1 and NRST. |
| <i>Page 85</i>  | Change signal name to NPCS0.  |
| <i>Page 86</i>  | Changes to Figure 15, Boot Program Algorithm Flow Diagram.  |
| <i>Page 87</i>  | Corrected BMS state to high during reset. Corrected address for internal ROM mapping.   |
| <i>Page 89</i>  | In Table 21 and text, corrected device names AT45DBxxx.   |
| <i>Page 91</i>  | Changes to Figure 20, Serial DataFlash Download.  |
| <i>Page 96</i>  | Updated Table 24 with new pins used and table note.   |
| <i>Page 108</i> | Code change in section Description of the SvcXmodem Structure.  |
| <i>Page 109</i> | Code change in Table 29: Xmodem Service, first table cell.  |
| <i>Page 110</i> | Code change in section Using the Service.   |
| <i>Page 111</i> | Code change in Table 30: DataFlash Service Methods, first table cell.   |
| <i>Page 116</i> | Code change in Steps 1 and 2 in section Using the Service.  |
| <i>Page 233</i> | Changed Table 58, I/O Line Description.   |
| <i>Page 245</i> | In AIC Source Mode Register, corrected descriptions of bits PRIOR and SRCTYPE.  |
| <i>Page 255</i> | Change number of programmable clocks to four. Correct oscillator speed to read 32.768 kHz.  |
| <i>Page 256</i> | Updated section I/O Lines with new information on clocks.   |
| <i>Page 257</i> | New PMC Block Diagram, Figure 117.  |

|                 |  |
|-----------------|--|
| <i>Page 258</i> | Updated Processor Clock and Programmable Clock Outputs descriptions. Updated Clock Generator description.  |
| <i>Page 259</i> | New Clock Generator Block Diagram, Figure 118. Section Slow Clock Oscillator Startup Time updated.   |
| <i>Page 261</i> | Added section Main Oscillator Bypass.  |
| <i>Page 263</i> | Updated section PLLB Divider by 2.   |
| <i>Page 264</i> | In section Master Clock Controller, changed references to PLLB Output to PLLB Clock. New Figure 124: Master Clock Controller. In section Processor Clock Source, specified differences between ARM7-based and ARM9-based systems.                                |
| <i>Page 265</i> | Section Programmable Clock Output Controller updated to show change in number of programmable clocks.  |
| <i>Page 267</i> | In Table 60: Clock Switching Timings (Worst Case), changed PLLA Output to PLLA Clock and PLLB Output to PLLB Clock.  |
| <i>Page 268</i> | In Figure 125: Switch Master Clock from Slow Clock to PLLA Clock and in Figure 126: Switch Master Clock from Main Clock to Slow Clock, changed signal names and waveform labels.   |
| <i>Page 269</i> | In Figure 127: Change PLLA Programming, changed signal names and labels. New Figure 128: Programmable Clock Output Programming.  |
| <i>Page 270</i> | Changed register names in Table 61: PMC Register Mapping: PMC_MOR to CKGR_MOR, PMC_MCFR to CKGR_MCFR, PMC_PLLAR to CKGR_PLLAR and PCM_PLLBR to CKGR_PLLBR. Remove registers PMC_PCK4, PMC_PCK5, PMC_PCK6 and PMC_PCK7 (addresses 0x0050 to 0x005C).              |
| <i>Page 271</i> | In register PMC_SCER, deleted bits PCK7 to PCK4, fields 15 to 12. All bit names updated to include "Enable". In UHP bit description, deleted reference to 12 MHz clock.  |
| <i>Page 272</i> | In register PMC_SCDR, deleted bits PCK7 to PCK4, fields 15 to 12. All bit names updated to include "Disable". In UHP bit description, deleted reference to 12 MHz clock.   |
| <i>Page 273</i> | In register PMC_SCSR, deleted bits PCK7 to PCK4, fields 15 to 12. All bit names updated to include "Status". In UHP bit description, corrected to read "USB Host Port".  |
| <i>Page 276</i> | Changed register name to PMC Clock Generator Main Oscillator Register. MOSCEN bit description changed to include information on Main Clock signal and crystal connection. OSCOUNT bit description changed to remove multiplication factor for Slow Clock cycles. |
| <i>Page 277</i> | Changed register name to PMC Clock Generator Main Clock Frequency Register. Corrected in MAINRDY field description reference to MAINF.   |
| <i>Page 278</i> | Changed register name to PMC Clock Generator PLL A Register. In OUTA and MULA bits, changed references to PLLA Output to PLL A Clock.  |

|                 |   |
|-----------------|---|
| <i>Page 279</i> | Changed register name to PMC Clock Generator PLL B Register. In OUTB and MULB bits, changed references to PLLB Output to PLL B Clock. Changed bit description for USB_96M.  |
| <i>Page 280</i> | In PMC_MCKR, new clock source selections specified for CSS. MDIV bit condition added.   |
| <i>Page 281</i> | In PMC_PCK0 to PMC_PCK3, new clock source selections specified for CSS.   |
| <i>Page 282</i> | In PMC_IER and PMC_IDR, bits PCK7RDY, PCK6RDY, PCK5RDY and PCK4RDY removed.   |
| <i>Page 283</i> | In PMC_SR, bits PCK7RDY, PCK6RDY, PCK5RDY and PCK4RDY removed.  |
| <i>Page 284</i> | In PMC_IMR, bits PCK7RDY, PCK6RDY, PCK5RDY and PCK4RDY removed.   |
| <i>Page 312</i> | Added Note to Figure 135.   |
| <i>Page 331</i> | In DBGU Chip ID Register, corrected NVPTYP field to 000 for ROM.  |
| <i>Page 343</i> | In Table 67: PIO Register Mapping, PIO_OWSR access changed to read-only.  |
| <i>Page 358</i> | In PIO_OWSR, access changed to read-only.   |
| <i>Page 368</i> | Changed all references from CPHA to NCPHA. Updated Figures 159 and 160 for clarity.   |
| <i>Page 391</i> | In CHDIV and CLDIV bit descriptions in register TWI_CWGR, corrected equations for calculation of SCL high and low periods. In CHDIV, CLDIV and CKDIV bit descriptions in register TWI_CWGR, SCL replaced by TWCK.   |
| <i>Page 452</i> | Updated Figure 214, Transmit Frame Format in Continuous Mode. Updated Figure 215, Receive Frame Format in Continuous Mode.  |
| <i>Page 460</i> | In register SSC_RFMR, new description of bit DATLEN.  |
| <i>Page 596</i> | In Table 109, DC Characteristics, changed conditions for Static Current.  |
| <i>Page 598</i> | New consumption figures in Table 113 and Table 114.   |
| <i>Page 599</i> | In Table 115: 32 kHz Oscillator Characteristics, $V_{DDOSC}$ defined in Startup Time conditions. In Table 116: Main Oscillator Characteristics, $V_{DDPLL}$ defined in Startup Time conditions. In Table 117: Phase Lock Loop Characteristics, corrected errors in Pump current max/min values. |
| <i>Page 601</i> | In Table 120: Switching Characteristics in Full Speed, min/max values for Rise/Fall Time Matching added.  |
| <i>Page 614</i> | In Table 125: SDRAMC Signals, changed min values for SDRAMC <sub>23</sub> to SDRAMC <sub>28</sub> .   |





## Table of Contents

|   |           |
|---|-----------|
| <b>AT91RM9200 Overview.....</b>                       | <b>1</b>  |
| <b>Features.....</b>                                  | <b>1</b>  |
| <b>Description.....</b>                               | <b>2</b>  |
| <b>Block Diagram.....</b>                             | <b>3</b>  |
| <b>Key Features.....</b>                              | <b>4</b>  |
| ARM920T Processor.....                                | 4         |
| Debug and Test.....                                   | 4         |
| Boot Program.....                                     | 5         |
| Embedded Software Services.....                       | 5         |
| Reset Controller.....                                 | 5         |
| Memory Controller.....                                | 5         |
| External Bus Interface.....                           | 6         |
| Static Memory Controller.....                         | 6         |
| SDRAM Controller.....                                 | 6         |
| Burst Flash Controller.....                           | 7         |
| Peripheral Data Controller.....                       | 7         |
| Advanced Interrupt Controller.....                    | 7         |
| Power Management Controller.....                      | 8         |
| System Timer.....                                     | 8         |
| Real Time Clock.....                                  | 8         |
| Debug Unit.....                                       | 8         |
| PIO Controller.....                                   | 9         |
| USB Host Port.....                                    | 9         |
| USB Device Port.....                                  | 9         |
| Ethernet MAC.....                                     | 10        |
| Serial Peripheral Interface.....                      | 10        |
| Two-wire Interface.....                               | 10        |
| USART.....  | 10        |
| Serial Synchronous Controller.....                    | 11        |
| Timer Counter.....                                    | 11        |
| MultiMedia Card Interface.....                        | 11        |
| <br>  |           |
| <b>AT91RM9200 Product Properties.....</b>             | <b>13</b> |
| <b>Power Supplies.....</b>                            | <b>13</b> |
| <b>Pinout.....</b>                                    | <b>13</b> |
| 208-lead PQFP Package Pinout.....                     | 14        |
| Mechanical Overview of the 208-lead PQFP Package..... | 15        |
| 256-ball BGA Package Pinout.....                      | 16        |
| Mechanical Overview of the 256-ball BGA Package.....  | 18        |
| <b>Peripheral Multiplexing on PIO Lines.....</b>      | <b>18</b> |
| PIO Controller A Multiplexing.....                    | 19        |
| PIO Controller B Multiplexing.....                    | 20        |
| PIO Controller C Multiplexing.....                    | 21        |
| PIO Controller D Multiplexing.....                    | 22        |
| <b>Pin Name Description.....</b>                      | <b>23</b> |





|                                       |           |
|---------------------------------------|-----------|
| <b>Peripheral Identifiers .....</b>   | <b>28</b> |
| System Interrupt.....                 | 29        |
| External Interrupts.....              | 29        |
| <b>Product Memory Mapping.....</b>    | <b>30</b> |
| External Memory Mapping .....         | 30        |
| Internal Memory Mapping .....         | 31        |
| Peripheral Mapping .....              | 32        |
| <b>Peripheral Implementation.....</b> | <b>34</b> |
| USART .....                           | 34        |
| Timer Counter .....                   | 34        |

---

|   |           |
|---|-----------|
| <b>ARM920T Processor Overview.....</b>                  | <b>35</b> |
| <b>Overview.....</b>                                    | <b>35</b> |
| <b>Block Diagram.....</b>                               | <b>36</b> |
| <b>ARM9TDMI Processor .....</b>                         | <b>37</b> |
| Instruction Type.....                                   | 37        |
| Data Types.....   | 37        |
| ARM9TDMI Operating Modes .....                          | 37        |
| ARM9TDMI Registers .....                                | 38        |
| ARM Instruction Set Overview .....                      | 40        |
| Thumb Instruction Set Overview .....                    | 41        |
| <b>CP15 Coprocessor.....</b>                            | <b>42</b> |
| CP15 Register Access .....                              | 43        |
| <b>Memory Management Unit (MMU).....</b>                | <b>44</b> |
| Domain.....   | 44        |
| MMU Faults.....   | 44        |
| <b>Caches, Write Buffers and Physical Address .....</b> | <b>45</b> |
| Instruction Cache (ICache) .....                        | 45        |
| Data Cache (DCache) and Write Buffer .....              | 45        |
| <b>ARM920T User Interface .....</b>                     | <b>47</b> |
| CP15 Register 0, ID Code and Cache Type .....           | 47        |
| CP15 Register 1, Control .....                          | 49        |
| CP15 Register 2, TTB .....                              | 51        |
| CP15 Register 3, Domain Access Control Register.....    | 52        |
| CP15 Register 4, Reserved .....                         | 53        |
| CP15 Register 5, Fault Status Register .....            | 53        |
| CP15 Register 6, Fault Address Register .....           | 54        |
| CP15 Register 7, Cache Operation Register .....         | 55        |
| CP15 Register 8, TLB Operations Register .....          | 57        |
| CP15 Register 9, Cache Lockdown Register.....           | 58        |
| CP15 Register 10, TLB Lockdown Register.....            | 59        |
| CP15 Registers 11, 12, Reserved.....                    | 59        |
| CP15 Register 13, FCSE PID Register .....               | 60        |
| CP15 Register 14, Reserved .....                        | 60        |
| CP15 Register 15, Test Configuration Register .....     | 60        |

---

|   |           |
|---|-----------|
| <b>Debug and Test Features (DBG Test)</b> ..... | <b>61</b> |
| <b>Overview</b> .....                           | <b>61</b> |
| <b>Block Diagram</b> .....                      | <b>62</b> |
| <b>Application Examples</b> .....               | <b>63</b> |
| Debug Environment .....                         | 63        |
| <b>Test Environment</b> .....                   | <b>63</b> |
| <b>Debug and Test Pin Description</b> .....     | <b>64</b> |
| <b>Functional Description</b> .....             | <b>65</b> |
| Test Mode Pins .....                            | 65        |
| Embedded In-Circuit Emulator .....              | 65        |
| Debug Unit .....                                | 65        |
| Embedded Trace Macrocell .....                  | 65        |
| IEEE 1149.1 JTAG Boundary Scan .....            | 69        |
| AT91RM9200 ID Code Register .....               | 83        |

---

|  |           |
|--|-----------|
| <b>Boot Program</b> .....                      | <b>85</b> |
| <b>Overview</b> .....                          | <b>85</b> |
| <b>Flow Diagram</b> .....                      | <b>86</b> |
| <b>Bootloader</b> .....                        | <b>87</b> |
| Valid Image Detection .....                    | 88        |
| Structure of ARM Vector 6 .....                | 89        |
| Bootloader Sequence.....                       | 90        |
| <b>Boot Uploader</b> .....                     | <b>94</b> |
| External Communication Channels.....           | 94        |
| <b>Hardware and Software Constraints</b> ..... | <b>96</b> |

---

|   |            |
|---|------------|
| <b>Embedded Software Services</b> ..... | <b>97</b>  |
| <b>Overview</b> .....                   | <b>97</b>  |
| <b>Service Definition</b> .....         | <b>97</b>  |
| Service Structure.....                  | 97         |
| Using a Service .....                   | 98         |
| <b>Embedded Software Services</b> ..... | <b>101</b> |
| Definition .....                        | 101        |
| ROM Entry Service .....                 | 101        |
| Tempo Service .....                     | 102        |
| Xmodem Service.....                     | 105        |
| DataFlash Service.....                  | 111        |
| CRC Service .....                       | 117        |
| Sine Service .....                      | 118        |

---

|  |            |
|--|------------|
| <b>AT91RM9200 Reset Controller</b> ..... | <b>119</b> |
| <b>Overview</b> .....                    | <b>119</b> |
| Reset Conditions.....                    | 119        |
| Reset Management.....                    | 120        |





|  |     |
|--|-----|
| Required Features for the Reset Controller ..... | 121 |
|--|-----|

---

|  |            |
|--|------------|
| <b>Memory Controller(MC).....</b>      | <b>123</b> |
| <b>Overview.....</b>                   | <b>123</b> |
| <b>Block Diagram.....</b>              | <b>124</b> |
| <b>Functional Description.....</b>     | <b>125</b> |
| Bus Arbiter .....                      | 125        |
| Address Decoder .....                  | 125        |
| Remap Command .....                    | 127        |
| Abort Status .....                     | 128        |
| Misalignment Detector .....            | 128        |
| Memory Controller Interrupt .....      | 129        |
| <b>User Interface.....</b>             | <b>129</b> |
| MC Remap Control Register .....        | 130        |
| MC Abort Status Register .....         | 131        |
| MC Abort Address Status Register ..... | 133        |
| MC Master Priority Register .....      | 134        |

---

|  |            |
|--|------------|
| <b>External Bus Interface (EBI) .....</b>                | <b>135</b> |
| <b>Overview.....</b>                                     | <b>135</b> |
| <b>Block Diagram.....</b>                                | <b>136</b> |
| <b>I/O Lines Description.....</b>                        | <b>137</b> |
| <b>Application Example .....</b>                         | <b>139</b> |
| Hardware Interface.....                                  | 139        |
| Connection Examples .....                                | 141        |
| <b>Product Dependencies.....</b>                         | <b>142</b> |
| I/O Lines.....   | 142        |
| <b>Functional Description.....</b>                       | <b>142</b> |
| Bus Multiplexing .....                                   | 142        |
| Pull-up Control .....                                    | 142        |
| Static Memory Controller.....                            | 142        |
| SDRAM Controller.....                                    | 142        |
| Burst Flash Controller .....                             | 142        |
| CompactFlash Support .....                               | 143        |
| SmartMedia and NAND Flash Support .....                  | 146        |
| <b>External Bus Interface (EBI) User Interface .....</b> | <b>148</b> |
| EBI Chip Select Assignment Register .....                | 149        |
| EBI Configuration Register.....                          | 150        |

---

|   |            |
|---|------------|
| <b>Static Memory Controller (SMC) .....</b> | <b>151</b> |
| <b>Overview.....</b>                        | <b>151</b> |
| <b>Block Diagram.....</b>                   | <b>152</b> |
| <b>Application Example .....</b>            | <b>153</b> |
| Hardware Interface.....                     | 153        |
| <b>Product Dependencies.....</b>            | <b>153</b> |

|  |            |
|--|------------|
| I/O Lines .....  | 153        |
| <b>Functional Description .....</b>                        | <b>154</b> |
| External Memory Interface .....                            | 154        |
| Write Access .....   | 156        |
| Read Access .....  | 159        |
| Wait State Management .....                                | 161        |
| Setup and Hold Cycles .....                                | 164        |
| LCD Interface Mode .....                                   | 168        |
| Memory Access Waveforms .....                              | 169        |
| <b>Static Memory Controller (SMC) User Interface .....</b> | <b>185</b> |
| SMC Chip Select Registers .....                            | 186        |

---

|   |            |
|---|------------|
| <b>SDRAM Controller (SDRAMC) .....</b>                | <b>189</b> |
| <b>Overview .....</b>                                 | <b>189</b> |
| <b>Block Diagram .....</b>                            | <b>190</b> |
| <b>I/O Lines Description .....</b>                    | <b>190</b> |
| <b>Application Example .....</b>                      | <b>191</b> |
| Hardware Interface .....                              | 191        |
| Software Interface .....                              | 192        |
| <b>Product Dependencies .....</b>                     | <b>194</b> |
| SDRAM Devices Initialization .....                    | 194        |
| I/O Lines .....                                       | 195        |
| Interrupt .....                                       | 195        |
| <b>Functional Description .....</b>                   | <b>195</b> |
| SDRAM Controller Write Cycle .....                    | 195        |
| SDRAM Controller Read Cycle .....                     | 196        |
| Border Management .....                               | 197        |
| SDRAM Controller Refresh Cycles .....                 | 198        |
| Power Management .....                                | 199        |
| <b>SDRAM Controller (SDRAMC) User Interface .....</b> | <b>201</b> |
| SDRAMC Mode Register .....                            | 202        |
| SDRAMC Refresh Timer Register .....                   | 203        |
| SDRAMC Configuration Register .....                   | 204        |
| SDRAMC Self-refresh Register .....                    | 206        |
| SDRAMC Low-power Register .....                       | 206        |
| SDRAMC Interrupt Enable Register .....                | 207        |
| SDRAMC Interrupt Disable Register .....               | 207        |
| SDRAMC Interrupt Mask Register .....                  | 208        |
| SDRAMC Interrupt Status Register .....                | 208        |

---

|   |            |
|---|------------|
| <b>Burst Flash Controller (BFC) .....</b> | <b>209</b> |
| <b>Overview .....</b>                     | <b>209</b> |
| <b>Block Diagram .....</b>                | <b>210</b> |
| <b>I/O Lines Description .....</b>        | <b>210</b> |
| <b>Application Example .....</b>          | <b>211</b> |



|  |            |
|--|------------|
| Burst Flash Interface .....                                  | 211        |
| <b>Product Dependencies.....</b>                             | <b>212</b> |
| Supported Burst Flash Devices.....                           | 212        |
| I/O Lines.....   | 212        |
| <b>Functional Description.....</b>                           | <b>212</b> |
| Burst Flash Controller Reset State.....                      | 212        |
| Burst Flash Controller Clock Selection.....                  | 212        |
| Burst Flash Controller Asynchronous Mode.....                | 213        |
| Burst Flash Controller Synchronous Mode .....                | 215        |
| <b>Burst Flash Controller (BFC) User Interface .....</b>     | <b>221</b> |
| Burst Flash Controller Mode Register .....                   | 221        |
| <hr/>  |            |
| <b>Peripheral Data Controller (PDC).....</b>                 | <b>223</b> |
| <b>Overview.....</b>   | <b>223</b> |
| <b>Block Diagram.....</b>                                    | <b>223</b> |
| <b>Functional Description.....</b>                           | <b>224</b> |
| Configuration.....   | 224        |
| Memory Pointers .....  | 224        |
| Transfer Counters .....                                      | 224        |
| Data Transfers .....   | 225        |
| Priority of PDC Transfer Requests.....                       | 225        |
| <b>Peripheral Data Controller (PDC) User Interface .....</b> | <b>226</b> |
| PDC Receive Pointer Register .....                           | 226        |
| PDC Receive Counter Register .....                           | 227        |
| PDC Transmit Pointer Register .....                          | 227        |
| PDC Transmit Counter Register .....                          | 227        |
| PDC Receive Next Pointer Register .....                      | 228        |
| PDC Receive Next Counter Register .....                      | 228        |
| PDC Transmit Next Pointer Register .....                     | 228        |
| PDC Transmit Next Counter Register .....                     | 229        |
| PDC Transfer Control Register .....                          | 229        |
| PDC Transfer Status Register.....                            | 230        |
| <hr/>  |            |
| <b>Advanced Interrupt Controller (AIC).....</b>              | <b>231</b> |
| <b>Overview.....</b>   | <b>231</b> |
| <b>Block Diagram.....</b>                                    | <b>232</b> |
| <b>Application Block Diagram .....</b>                       | <b>232</b> |
| <b>AIC Detailed Block Diagram .....</b>                      | <b>232</b> |
| <b>I/O Line Description.....</b>                             | <b>233</b> |
| <b>Product Dependencies.....</b>                             | <b>233</b> |
| I/O Lines.....   | 233        |
| Power Management .....                                       | 233        |
| Interrupt Sources.....                                       | 233        |
| <b>Functional Description.....</b>                           | <b>234</b> |
| Interrupt Source Control.....                                | 234        |

|   |            |
|---|------------|
| Interrupt Latencies .....                                       | 236        |
| Normal Interrupt .....  | 237        |
| Fast Interrupt.....   | 239        |
| Protect Mode.....   | 242        |
| Spurious Interrupt.....   | 243        |
| General Interrupt Mask .....                                    | 243        |
| <b>Advanced Interrupt Controller (AIC) User Interface .....</b> | <b>244</b> |
| AIC Source Mode Register .....                                  | 245        |
| AIC Source Vector Register .....                                | 245        |
| AIC Interrupt Vector Register .....                             | 246        |
| AIC FIQ Vector Register .....                                   | 246        |
| AIC Interrupt Status Register .....                             | 247        |
| AIC Interrupt Pending Register .....                            | 247        |
| AIC Interrupt Mask Register .....                               | 248        |
| AIC Core Interrupt Status Register .....                        | 248        |
| AIC Interrupt Enable Command Register.....                      | 249        |
| AIC Interrupt Disable Command Register.....                     | 249        |
| AIC Interrupt Clear Command Register .....                      | 250        |
| AIC Interrupt Set Command Register .....                        | 250        |
| AIC End of Interrupt Command Register .....                     | 251        |
| AIC Spurious Interrupt Vector Register.....                     | 251        |
| AIC Debug Control Register.....                                 | 252        |
| AIC Fast Forcing Enable Register.....                           | 253        |
| AIC Fast Forcing Disable Register.....                          | 253        |
| AIC Fast Forcing Status Register.....                           | 254        |

---

|  |            |
|--|------------|
| <b>Power Management Controller (PMC) .....</b> | <b>255</b> |
| <b>Overview.....</b>                           | <b>255</b> |
| <b>Product Dependencies.....</b>               | <b>256</b> |
| I/O Lines.....                                 | 256        |
| Interrupt.....                                 | 256        |
| Oscillator and PLL Characteristics .....       | 256        |
| Peripheral Clocks .....                        | 256        |
| USB Clocks .....                               | 256        |
| <b>Block Diagram.....</b>                      | <b>257</b> |
| <b>Functional Description.....</b>             | <b>258</b> |
| Operating Modes Definition.....                | 258        |
| Clock Definitions .....                        | 258        |
| Clock Generator .....                          | 258        |
| Slow Clock Oscillator .....                    | 259        |
| Main Oscillator .....                          | 260        |
| Divider and PLL Blocks .....                   | 262        |
| Clock Controllers.....                         | 263        |
| <b>Clock Switching Details .....</b>           | <b>267</b> |
| Master Clock Switching Timings .....           | 267        |
| Clock Switching Waveforms.....                 | 268        |





|   |            |
|---|------------|
| <b>Power Management Controller (PMC) User Interface .....</b> | <b>270</b> |
| PMC System Clock Enable Register .....                        | 271        |
| PMC System Clock Disable Register .....                       | 272        |
| PMC System Clock Status Register .....                        | 273        |
| PMC Peripheral Clock Enable Register .....                    | 274        |
| PMC Peripheral Clock Disable Register .....                   | 274        |
| PMC Peripheral Clock Status Register .....                    | 275        |
| PMC Clock Generator Main Oscillator Register .....            | 276        |
| PMC Clock Generator Main Clock Frequency Register .....       | 277        |
| PMC Clock Generator PLL A Register .....                      | 278        |
| PMC Clock Generator PLL B Register .....                      | 279        |
| PMC Master Clock Register .....                               | 280        |
| PMC Programmable Clock Register 0 to 3 .....                  | 281        |
| PMC Interrupt Enable Register .....                           | 282        |
| PMC Interrupt Disable Register .....                          | 282        |
| PMC Status Register .....                                     | 283        |
| PMC Interrupt Mask Register .....                             | 284        |

---

|   |            |
|---|------------|
| <b>System Timer (ST) .....</b>                | <b>285</b> |
| <b>Overview .....</b>                         | <b>285</b> |
| <b>Block Diagram .....</b>                    | <b>285</b> |
| <b>Application Block Diagram .....</b>        | <b>285</b> |
| <b>Product Dependencies .....</b>             | <b>286</b> |
| Power Management .....                        | 286        |
| Interrupt Sources .....                       | 286        |
| Watchdog Overflow .....                       | 286        |
| <b>Functional Description .....</b>           | <b>286</b> |
| <b>System Timer Clock .....</b>               | <b>286</b> |
| Period Interval Timer (PIT) .....             | 286        |
| Watchdog Timer (WDT) .....                    | 287        |
| Real-time Timer (RTT) .....                   | 287        |
| <b>System Timer (ST) User Interface .....</b> | <b>289</b> |
| ST Control Register .....                     | 289        |
| ST Period Interval Mode Register .....        | 290        |
| ST Watchdog Mode Register .....               | 290        |
| ST Real-Time Mode Register .....              | 291        |
| ST Status Register .....                      | 291        |
| ST Interrupt Enable Register .....            | 292        |
| ST Interrupt Disable Register .....           | 292        |
| ST Interrupt Mask Register .....              | 293        |
| ST Real-time Alarm Register .....             | 293        |
| ST Current Real-Time Register .....           | 294        |

---

|   |            |
|---|------------|
| <b>Real Time Controller (RTC) .....</b> | <b>295</b> |
| <b>Overview .....</b>                   | <b>295</b> |



|   |            |
|---|------------|
| <b>Block Diagram.....</b>                             | <b>295</b> |
| <b>Product Dependencies.....</b>                      | <b>295</b> |
| Power Management .....                                | 295        |
| Interrupt.....  | 295        |
| <b>Functional Description.....</b>                    | <b>296</b> |
| Reference Clock.....                                  | 296        |
| Timing .....  | 296        |
| Alarm.....  | 296        |
| Error Checking .....                                  | 296        |
| Updating Time/Calendar .....                          | 297        |
| <b>Real Time Controller (RTC) User Interface.....</b> | <b>298</b> |
| RTC Control Register .....                            | 299        |
| RTC Mode Register .....                               | 300        |
| RTC Time Register .....                               | 301        |
| RTC Calendar Register.....                            | 302        |
| RTC Time Alarm Register .....                         | 303        |
| RTC Calendar Alarm Register .....                     | 304        |
| RTC Status Register .....                             | 305        |
| RTC Status Clear Command Register .....               | 306        |
| RTC Interrupt Enable Register.....                    | 307        |
| RTC Interrupt Disable Register .....                  | 308        |
| RTC Interrupt Mask Register .....                     | 309        |
| RTC Valid Entry Register .....                        | 310        |
| <hr/>   |            |
| <b>Debug Unit (DBGU) .....</b>                        | <b>311</b> |
| <b>Overview.....</b>                                  | <b>311</b> |
| <b>Block Diagram.....</b>                             | <b>312</b> |
| <b>Product Dependencies.....</b>                      | <b>313</b> |
| I/O Lines.....  | 313        |
| Power Management .....                                | 313        |
| Interrupt Source .....                                | 313        |
| <b>UART Operations.....</b>                           | <b>313</b> |
| Baud Rate Generator .....                             | 313        |
| Receiver .....  | 314        |
| Transmitter .....                                     | 316        |
| Peripheral Data Controller.....                       | 317        |
| Test Modes .....                                      | 317        |
| Debug Communication Channel Support.....              | 319        |
| Chip Identifier .....                                 | 319        |
| ICE Access Prevention .....                           | 319        |
| <b>Debug Unit User Interface .....</b>                | <b>320</b> |
| Debug Unit Control Register .....                     | 321        |
| Debug Unit Mode Register .....                        | 322        |
| Debug Unit Interrupt Enable Register .....            | 323        |
| Debug Unit Interrupt Disable Register .....           | 324        |
| Debug Unit Interrupt Mask Register .....              | 325        |



|  |     |
|--|-----|
| Debug Unit Status Register.....              | 326 |
| Debug Unit Receiver Holding Register .....   | 328 |
| Debug Unit Baud Rate Generator Register..... | 329 |
| Debug Unit Chip ID Register.....             | 330 |
| Debug Unit Chip ID Extension Register .....  | 332 |
| Debug Unit Force NTRST Register.....         | 332 |

---

|   |            |
|---|------------|
| <b>Parallel Input/Output Controller (PIO) .....</b>               | <b>333</b> |
| <b>Overview.....</b>  | <b>333</b> |
| <b>Block Diagram.....</b>   | <b>334</b> |
| <b>Product Dependencies.....</b>                                  | <b>335</b> |
| Pin Multiplexing .....  | 335        |
| External Interrupt Lines .....                                    | 335        |
| Power Management .....  | 335        |
| Interrupt Generation .....  | 335        |
| <b>Functional Description.....</b>                                | <b>336</b> |
| Pull-up Resistor Control .....                                    | 337        |
| I/O Line or Peripheral Function Selection .....                   | 337        |
| Peripheral A or B Selection .....                                 | 337        |
| Output Control.....   | 337        |
| Synchronous Data Output.....                                      | 338        |
| Multi Drive Control (Open Drain).....                             | 338        |
| Output Line Timings .....   | 338        |
| Inputs .....  | 339        |
| Input Glitch Filtering .....                                      | 339        |
| Input Change Interrupt .....                                      | 340        |
| <b>I/O Lines Programming Example .....</b>                        | <b>341</b> |
| <b>Parallel Input/Output Controller (PIO) User Interface.....</b> | <b>342</b> |
| PIO Enable Register .....   | 344        |
| PIO Disable Register.....   | 344        |
| PIO Status Register .....   | 345        |
| PIO Output Enable Register.....                                   | 345        |
| PIO Output Disable Register .....                                 | 346        |
| PIO Output Status Register.....                                   | 346        |
| PIO Input Filter Enable Register .....                            | 347        |
| PIO Input Filter Disable Register.....                            | 347        |
| PIO Input Filter Status Register .....                            | 348        |
| PIO Set Output Data Register .....                                | 348        |
| PIO Clear Output Data Register.....                               | 349        |
| PIO Output Data Status Register .....                             | 349        |
| PIO Pin Data Status Register.....                                 | 350        |
| PIO Interrupt Enable Register .....                               | 350        |
| PIO Interrupt Disable Register .....                              | 351        |
| PIO Interrupt Mask Register .....                                 | 351        |
| PIO Interrupt Status Register .....                               | 352        |
| PIO Multi-driver Enable Register.....                             | 352        |

|   |     |
|---|-----|
| PIO Multi-driver Disable Register ..... | 353 |
| PIO Multi-driver Status Register.....   | 353 |
| PIO Pull Up Disable Register .....      | 354 |
| PIO Pull Up Enable Register .....       | 354 |
| PIO Pad Pull Up Status Register .....   | 355 |
| PIO Peripheral A Select Register.....   | 355 |
| PIO Peripheral B Select Register.....   | 356 |
| PIO Peripheral AB Status Register ..... | 356 |
| PIO Output Write Enable Register .....  | 357 |
| PIO Output Write Disable Register ..... | 357 |
| PIO Output Write Status Register .....  | 358 |

---

## **Serial Peripheral Interface (SPI)..... 359**

**Overview..... 359**

**Block Diagram..... 360**

**Application Block Diagram .....** 361

**Product Dependencies..... 362**

I/O Lines..... 362

Power Management..... 362

Interrupt..... 362

**Functional Description..... 362**

Master Mode Operations..... 362

SPI Slave Mode .....

Data Transfer .....

**Serial Peripheral Interface (SPI) User Interface..... 370**

SPI Control Register .....

SPI Mode Register .....

SPI Receive Data Register .....

SPI Transmit Data Register .....

SPI Status Register.....

SPI Interrupt Enable Register .....

SPI Interrupt Disable Register.....

SPI Interrupt Mask Register .....

SPI Chip Select Register.....

---

## **Two-wire Interface (TWI)..... 381**

**Overview..... 381**

**Block Diagram..... 381**

**Application Block Diagram .....** 381

**Product Dependencies..... 382**

I/O Lines..... 382

Power Management..... 382

Interrupt..... 382

**Functional Description..... 382**

Transfer Format .....



|  |            |
|--|------------|
| Modes of Operation.....                              | 383        |
| Transmitting Data.....                               | 383        |
| Read/Write Flowcharts.....                           | 385        |
| <b>Two-wire Interface (TWI) User Interface .....</b> | <b>388</b> |
| TWI Control Register.....                            | 389        |
| TWI Master Mode Register .....                       | 390        |
| TWI Internal Address Register .....                  | 391        |
| TWI Clock Waveform Generator Register.....           | 391        |
| TWI Status Register .....                            | 392        |
| TWI Interrupt Enable Register.....                   | 393        |
| TWI Interrupt Disable Register.....                  | 394        |
| TWI Interrupt Mask Register .....                    | 395        |
| TWI Receive Holding Register .....                   | 396        |
| TWI Transmit Holding Register .....                  | 396        |

---

**Universal Synchronous Asynchronous Receiver Transceiver (USART) 397**

|  |            |
|--|------------|
| <b>Overview.....</b>                       | <b>397</b> |
| <b>Block Diagram.....</b>                  | <b>398</b> |
| <b>Application Block Diagram .....</b>     | <b>399</b> |
| <b>I/O Lines Description .....</b>         | <b>399</b> |
| <b>Product Dependencies.....</b>           | <b>399</b> |
| I/O Lines.....                             | 399        |
| Power Management .....                     | 400        |
| Interrupt.....                             | 400        |
| <b>Functional Description.....</b>         | <b>400</b> |
| Baud Rate Generator .....                  | 400        |
| Receiver and Transmitter Control .....     | 404        |
| Synchronous and Asynchronous Modes.....    | 405        |
| ISO7816 Mode .....                         | 415        |
| IrDA Mode .....                            | 418        |
| RS485 Mode .....                           | 421        |
| Modem Mode .....                           | 422        |
| Test Modes .....                           | 422        |
| <b>USART User Interface .....</b>          | <b>424</b> |
| USART Control Register .....               | 425        |
| USART Mode Register.....                   | 427        |
| USART Interrupt Enable Register .....      | 430        |
| USART Interrupt Disable Register .....     | 431        |
| USART Interrupt Mask Register.....         | 432        |
| USART Channel Status Register .....        | 433        |
| USART Receive Holding Register .....       | 435        |
| USART Transmit Holding Register .....      | 435        |
| USART Baud Rate Generator Register .....   | 436        |
| USART Receiver Time-out Register .....     | 437        |
| USART Transmitter Timeguard Register ..... | 437        |
| USART FI DI RATIO Register.....            | 438        |

|                                       |     |
|---------------------------------------|-----|
| USART Number of Errors Register ..... | 439 |
| USART IrDA FILTER Register .....      | 440 |

---

|   |            |
|---|------------|
| <b>Serial Synchronous Controller (SSC).....</b>                 | <b>441</b> |
| <b>Overview.....</b>  | <b>441</b> |
| <b>Block Diagram.....</b>                                       | <b>442</b> |
| <b>Application Block Diagram .....</b>                          | <b>442</b> |
| <b>Pin Name List .....</b>                                      | <b>443</b> |
| <b>Product Dependencies.....</b>                                | <b>443</b> |
| I/O Lines.....  | 443        |
| Power Management .....  | 443        |
| Interrupt.....  | 443        |
| <b>Functional Description.....</b>                              | <b>444</b> |
| Clock Management .....  | 445        |
| Transmitter Operations .....                                    | 447        |
| Receiver Operations .....                                       | 448        |
| Start.....  | 448        |
| Frame Sync.....   | 450        |
| Data Format .....   | 450        |
| Loop Mode .....   | 452        |
| Interrupt.....  | 452        |
| <b>SSC Application Examples .....</b>                           | <b>453</b> |
| <b>Serial Synchronous Controller (SSC) User Interface .....</b> | <b>455</b> |
| SSC Control Register.....                                       | 456        |
| SSC Clock Mode Register .....                                   | 457        |
| SSC Receive Clock Mode Register .....                           | 458        |
| SSC Receive Frame Mode Register .....                           | 460        |
| SSC Transmit Clock Mode Register .....                          | 462        |
| SSC Transmit Frame Mode Register .....                          | 464        |
| SSC Receive Holding Register .....                              | 466        |
| SSC Transmit Holding Register .....                             | 466        |
| SSC Receive Synchronization Holding Register.....               | 467        |
| SSC Transmit Synchronization Holding Register.....              | 467        |
| SSC Status Register .....                                       | 468        |
| SSC Interrupt Enable Register.....                              | 470        |
| SSC Interrupt Disable Register .....                            | 471        |
| SSC Interrupt Mask Register .....                               | 472        |

---

|                                  |            |
|----------------------------------|------------|
| <b>Timer Counter (TC).....</b>   | <b>473</b> |
| <b>Overview.....</b>             | <b>473</b> |
| <b>Block Diagram.....</b>        | <b>474</b> |
| <b>Pin Name List .....</b>       | <b>475</b> |
| <b>Product Dependencies.....</b> | <b>475</b> |
| I/O Lines.....                   | 475        |
| Power Management .....           | 475        |



|   |            |
|---|------------|
| Interrupt.....                                | 475        |
| <b>Functional Description.....</b>            | <b>475</b> |
| TC Description .....                          | 475        |
| Capture Operating Mode.....                   | 478        |
| <b>Waveform Operating Mode.....</b>           | <b>480</b> |
| <b>Timer Counter (TC) User Interface.....</b> | <b>487</b> |
| TC Block Control Register.....                | 488        |
| TC Block Mode Register .....                  | 488        |
| TC Channel Control Register .....             | 489        |
| TC Channel Mode Register: Capture Mode.....   | 490        |
| TC Channel Mode Register: Waveform Mode ..... | 492        |
| TC Counter Value Register .....               | 495        |
| TC Register A.....                            | 495        |
| TC Register B.....                            | 495        |
| TC Register C .....                           | 496        |
| TC Status Register.....                       | 496        |
| TC Interrupt Enable Register .....            | 498        |
| TC Interrupt Disable Register.....            | 499        |
| TC Interrupt Mask Register .....              | 500        |

---

|   |            |
|---|------------|
| <b>MultiMedia Card Interface (MCI).....</b>       | <b>501</b> |
| <b>Overview.....</b>                              | <b>501</b> |
| <b>Block Diagram.....</b>                         | <b>502</b> |
| <b>Application Block Diagram .....</b>            | <b>503</b> |
| <b>Product Dependencies.....</b>                  | <b>504</b> |
| I/O Lines.....                                    | 504        |
| Power Management .....                            | 504        |
| Interrupt.....                                    | 504        |
| <b>Bus Topology.....</b>                          | <b>504</b> |
| <b>MultiMedia Card Operations .....</b>           | <b>506</b> |
| Command-response Operation.....                   | 507        |
| Data Transfer Operation .....                     | 508        |
| Read Operation.....                               | 509        |
| Write Operation .....                             | 510        |
| <b>SD Card Operations.....</b>                    | <b>511</b> |
| <b>MultiMedia Card (MCI) User Interface .....</b> | <b>512</b> |
| MCI Control Register.....                         | 513        |
| MCI Mode Register .....                           | 514        |
| MCI Data Timeout Register.....                    | 515        |
| MCI SD Card Register .....                        | 516        |
| MCI Argument Register.....                        | 516        |
| MCI Command Register.....                         | 517        |
| MCI SD Response Register .....                    | 518        |
| MCI SD Receive Data Register.....                 | 519        |
| MCI SD Transmit Data Register.....                | 519        |
| MCI Status Register .....                         | 520        |

|                                     |     |
|-------------------------------------|-----|
| MCI Interrupt Enable Register.....  | 522 |
| MCI Interrupt Disable Register..... | 523 |
| MCI Interrupt Mask Register .....   | 524 |

---

|  |            |
|--|------------|
| <b>USB Device Port (UDP) .....</b>                         | <b>525</b> |
| <b>Overview.....</b>                                       | <b>525</b> |
| <b>Block Diagram.....</b>                                  | <b>526</b> |
| <b>Product Dependencies.....</b>                           | <b>527</b> |
| I/O Lines.....   | 527        |
| Power Management .....                                     | 527        |
| Interrupt.....   | 527        |
| <b>Typical Connection.....</b>                             | <b>528</b> |
| <b>Functional Description.....</b>                         | <b>529</b> |
| USB V2.0 Full-speed Introduction.....                      | 529        |
| Handling Transactions with USB V2.0 Device Peripheral..... | 531        |
| Controlling Device States.....                             | 542        |
| <b>USB Device Port (UDP) User Interface .....</b>          | <b>544</b> |
| USB Frame Number Register .....                            | 545        |
| USB Global State Register.....                             | 546        |
| USB Function Address Register .....                        | 547        |
| USB Interrupt Enable Register.....                         | 548        |
| USB Interrupt Disable Register .....                       | 549        |
| USB Interrupt Mask Register .....                          | 550        |
| USB Interrupt Status Register.....                         | 551        |
| USB Interrupt Clear Register .....                         | 554        |
| USB Reset Endpoint Register.....                           | 555        |
| USB Endpoint Control and Status Register .....             | 556        |
| USB FIFO Data Register.....                                | 560        |

---

|   |            |
|---|------------|
| <b>USB Device Host Port (UHP) .....</b> | <b>561</b> |
| <b>Overview.....</b>                    | <b>561</b> |
| <b>Block Diagram.....</b>               | <b>561</b> |
| <b>Product Dependencies.....</b>        | <b>562</b> |
| I/O Lines.....                          | 562        |
| Power Management .....                  | 562        |
| Interrupt.....                          | 562        |
| <b>Functional Description.....</b>      | <b>562</b> |
| Host Controller Interface .....         | 562        |
| Host Controller Driver.....             | 563        |
| <b>Typical Connection.....</b>          | <b>564</b> |

---

|  |            |
|--|------------|
| <b>Ethernet MAC (EMAC).....</b>        | <b>565</b> |
| <b>Overview.....</b>                   | <b>565</b> |
| <b>Block Diagram.....</b>              | <b>566</b> |
| <b>Application Block Diagram .....</b> | <b>566</b> |





|  |            |
|--|------------|
| <b>Product Dependencies</b> .....                        | <b>567</b> |
| I/O Lines.....   | 567        |
| Power Management.....                                    | 567        |
| Interrupt.....   | 567        |
| <b>Functional Description</b> .....                      | <b>568</b> |
| Media Independent Interface.....                         | 569        |
| Transmit/Receive Operation.....                          | 570        |
| Frame Format Extensions.....                             | 571        |
| DMA Operations.....                                      | 572        |
| Address Checking.....                                    | 574        |
| <b>Ethernet MAC (EMAC) User Interface</b> .....          | <b>575</b> |
| EMAC Control Register.....                               | 577        |
| EMAC Configuration Register.....                         | 578        |
| EMAC Status Register.....                                | 580        |
| EMAC Transmit Address Register.....                      | 581        |
| EMAC Transmit Control Register.....                      | 582        |
| EMAC Transmit Status Register.....                       | 583        |
| EMAC Receive Buffer Queue Pointer Register.....          | 584        |
| EMAC Receive Status Register.....                        | 585        |
| EMAC Interrupt Status Register.....                      | 586        |
| EMAC Interrupt Enable Register.....                      | 587        |
| EMAC Interrupt Disable Register.....                     | 588        |
| EMAC Interrupt Mask Register.....                        | 589        |
| EMAC PHY Maintenance Register.....                       | 590        |
| EMAC Hash Address High Register.....                     | 591        |
| EMAC Hash Address Low Register.....                      | 591        |
| EMAC Specific Address (1, 2, 3 and 4) High Register..... | 592        |
| EMAC Specific Address (1, 2, 3 and 4) Low Register.....  | 592        |
| EMAC Statistics Register Block Registers.....            | 593        |
| <hr/>  |            |
| <b>AT91RM9200 Electrical Characteristics</b> .....       | <b>595</b> |
| <b>Absolute Maximum Ratings</b> .....                    | <b>595</b> |
| <b>DC Characteristics</b> .....                          | <b>596</b> |
| <b>Clocks Characteristics</b> .....                      | <b>597</b> |
| Processor Clock Characteristics.....                     | 597        |
| Master Clock Characteristics.....                        | 597        |
| XIN Clock Characteristics (1).....                       | 597        |
| <b>Power Consumption</b> .....                           | <b>598</b> |
| <b>Crystal Oscillators Characteristics</b> .....         | <b>599</b> |
| 32 kHz Oscillator Characteristics.....                   | 599        |
| Main Oscillator Characteristics.....                     | 599        |
| <b>PLL Characteristics</b> .....                         | <b>599</b> |
| <b>Transceiver Characteristics</b> .....                 | <b>600</b> |
| Electrical Characteristics.....                          | 600        |
| Switching Characteristics.....                           | 601        |



---

|  |            |
|--|------------|
| <b>AT91RM9200 AC Characteristics .....</b>           | <b>603</b> |
| <b>Applicable Conditions and Derating Data .....</b> | <b>603</b> |
| Conditions and Timings Computation .....             | 603        |
| Temperature Derating Factor .....                    | 604        |
| VDDCORE Voltage Derating Factor .....                | 604        |
| VDDIOM Voltage Derating Factor .....                 | 605        |
| <b>EBI Timings .....</b>                             | <b>606</b> |
| SMC Signals Relative to MCK .....                    | 606        |
| SDRAMC Signals Relative to SDCK .....                | 613        |
| BFC Signals Relative to BFCK .....                   | 616        |
| <b>JTAG/ICE Timings .....</b>                        | <b>621</b> |
| ICE Interface Signals .....                          | 621        |
| JTAG Interface Signals .....                         | 622        |
| <b>ETM Timings .....</b>                             | <b>624</b> |
| Timings Data .....                                   | 624        |
| Design Considerations .....                          | 624        |

---

|   |            |
|---|------------|
| <b>AT91RM9200 Mechanical Characteristics .....</b>  | <b>625</b> |
| <b>Thermal and Reliability Considerations .....</b> | <b>625</b> |
| Thermal Data .....                                  | 625        |
| Reliability Data .....                              | 625        |
| Junction Temperature .....                          | 626        |
| <b>Package Drawings .....</b>                       | <b>627</b> |

---

|  |            |
|--|------------|
| <b>AT91RM9200 Ordering Information .....</b> | <b>629</b> |
|--|------------|

---

|                               |            |
|-------------------------------|------------|
| <b>Document Details .....</b> | <b>631</b> |
| Revision History .....        | 631        |





## Atmel Corporation

2325 Orchard Parkway  
San Jose, CA 95131, USA  
Tel: 1(408) 441-0311  
Fax: 1(408) 487-2600

## Regional Headquarters

### Europe

Atmel Sarl  
Route des Arsenaux 41  
Case Postale 80  
CH-1705 Fribourg  
Switzerland  
Tel: (41) 26-426-5555  
Fax: (41) 26-426-5500

### Asia

Room 1219  
Chinachem Golden Plaza  
77 Mody Road Tsimshatsui  
East Kowloon  
Hong Kong  
Tel: (852) 2721-9778  
Fax: (852) 2722-1369

### Japan

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

2325 Orchard Parkway  
San Jose, CA 95131, USA  
Tel: 1(408) 441-0311  
Fax: 1(408) 436-4314

### Microcontrollers

2325 Orchard Parkway  
San Jose, CA 95131, USA  
Tel: 1(408) 441-0311  
Fax: 1(408) 436-4314

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

Zone Industrielle  
13106 Rousset Cedex, France  
Tel: (33) 4-42-53-60-00  
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.  
Colorado Springs, CO 80906, USA  
Tel: 1(719) 576-3300  
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park  
Maxwell Building  
East Kilbride G75 0QR, Scotland  
Tel: (44) 1355-803-000  
Fax: (44) 1355-242-743

### RF/Automotive

Theresienstrasse 2  
Postfach 3535  
74025 Heilbronn, Germany  
Tel: (49) 71-31-67-0  
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.  
Colorado Springs, CO 80906, USA  
Tel: 1(719) 576-3300  
Fax: 1(719) 540-1759

### Biometrics/Imaging/Hi-Rel MPU/ High Speed Converters/RF Datacom

Avenue de Rochepleine  
BP 123  
38521 Saint-Egreve Cedex, France  
Tel: (33) 4-76-58-30-00  
Fax: (33) 4-76-58-34-80



**Disclaimer:** 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 Corporation 2003. All rights reserved. ATMEL® and combinations thereof and DataFlash® are the registered trademarks of Atmel Corporation or its subsidiaries.

ARM®, ARM7TDMI® and Thumb® are the registered trademarks and ARM9TDMI™, ARM920T™ and AMBA™ are the trademarks of ARM Ltd.; CompactFlash® is a registered trademark of the CompactFlash Association; SmartMedia™ is a trademark of the Solid State Floppy Disk Card Forum.

Other terms and product names may be the trademarks of others.

## Literature Requests

[www.atmel.com/literature](http://www.atmel.com/literature)



Printed on recycled paper.

This datasheet has been download from:

[www.datasheetcatalog.com](http://www.datasheetcatalog.com)

Datasheets for electronics components.