

Features

- Incorporates the ARM7TDMI™ ARM® Thumb® Processor Core
 - High-performance 32-bit RISC Architecture
 - High-density 16-bit Instruction Set
 - Leader in MIPS/Watt
 - Embedded ICE (In-Circuit Emulation)
- On-chip SRAM and/or ROM
 - 32-bit Data Bus
 - Single-clock Cycle Access
- Fully Programmable External Bus Interface (EBI)
 - Maximum External Address Space of 64M Bytes
 - Up to 8 Chip Selects
 - Software Programmable 8/16-bit External Databus
- 8-level Priority, Individually Maskable, Vectored Interrupt Controller
 - 4 External Interrupts, Including a High-priority Low-latency Interrupt Request
- 32 Programmable I/O Lines
- 3-channel 16-bit Timer/Counter
 - 3 External Clock Inputs
 - 2 Multi-purpose I/O Pins per Channel
- 2 USARTs
 - 2 Dedicated Peripheral Data Controller (PDC) Channels per USART
- Programmable Watchdog Timer
- Advanced Power-saving Features
 - CPU and Peripheral Can be Deactivated Individually
- Available in a 100-lead TQFP Package

| Microcontroller | Primary SRAM Bank | Secondary SRAM Bank | ROM |
|-----------------|-------------------|---------------------|------------|
| AT91M40800 | 8K Bytes | – | – |
| AT91R40807 | 8K Bytes | 128K Bytes | – |
| AT91M40807 | 8K Bytes | – | 128K Bytes |
| AT91R40008 | 256K Bytes | – | – |

Description

The AT91X40 Series is a subset of the Atmel AT91 16/32-bit microcontroller family, which is based on the ARM7TDMI processor core. This processor has a high-performance 32-bit RISC architecture with a high-density 16-bit instruction set and very low power consumption. In addition, a large number of internally banked registers result in very fast exception handling, making the device ideal for real-time control applications.

The AT91X40 Series features a direct connection to off-chip memory, including Flash, through the fully programmable External Bus Interface (EBI). An eight-level priority vectored interrupt controller, in conjunction with the Peripheral Data Controller significantly improve the real-time performance of the device.

The devices are manufactured using Atmel's high-density CMOS technology. By combining the ARM7TDMI processor core with on-chip high-speed memory and a wide range of peripheral functions on a monolithic chip, the Atmel AT91X40 Series is a family of powerful microcontrollers that offer a flexible, cost-effective solution to many compute-intensive embedded control applications.



AT91 ARM® Thumb® Microcontrollers

AT91M40800
AT91R40807
AT91M40807
AT91R40008



Pin Configuration

Figure 1. AT91X40 Series Pinout (Top View)

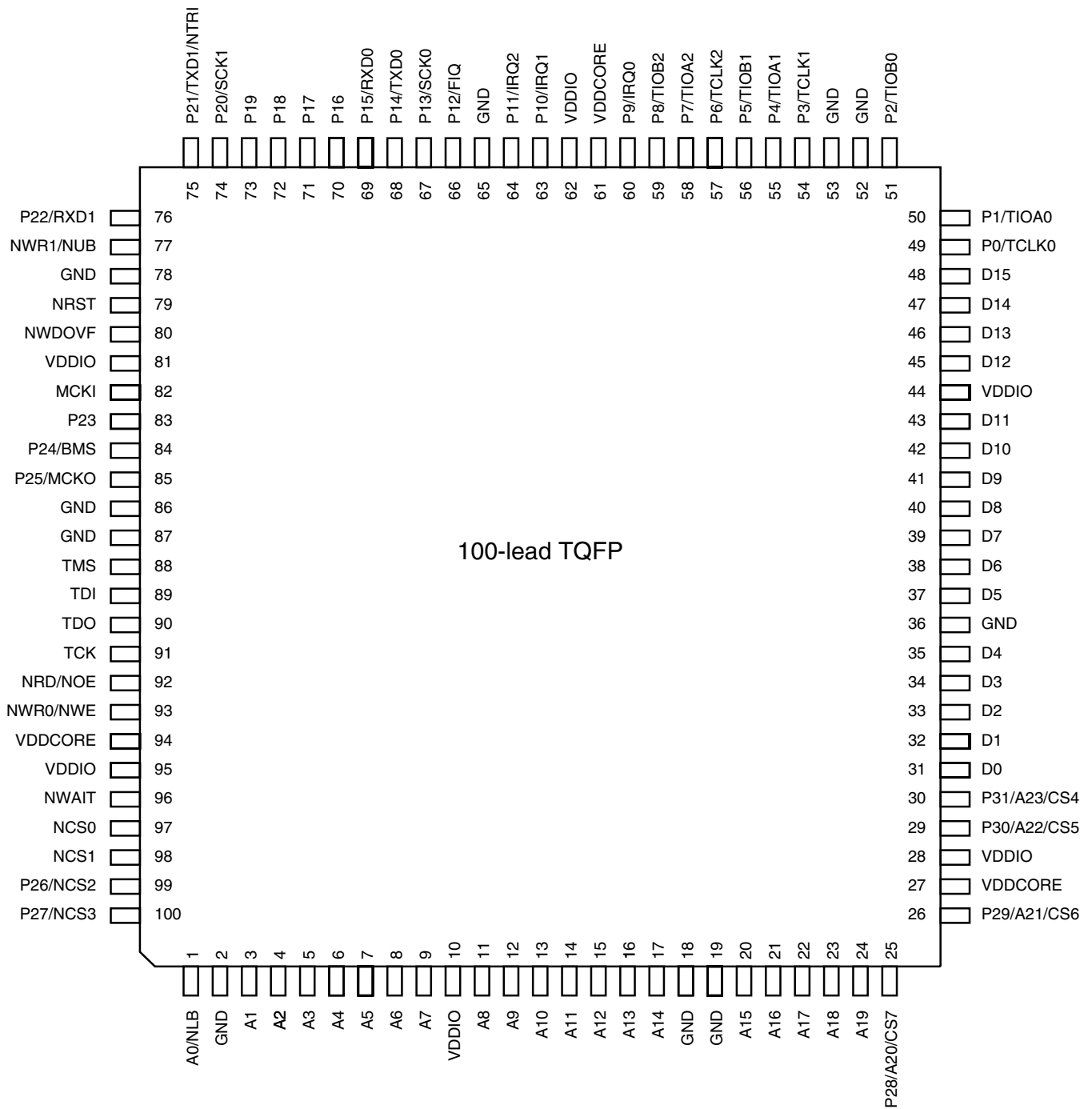
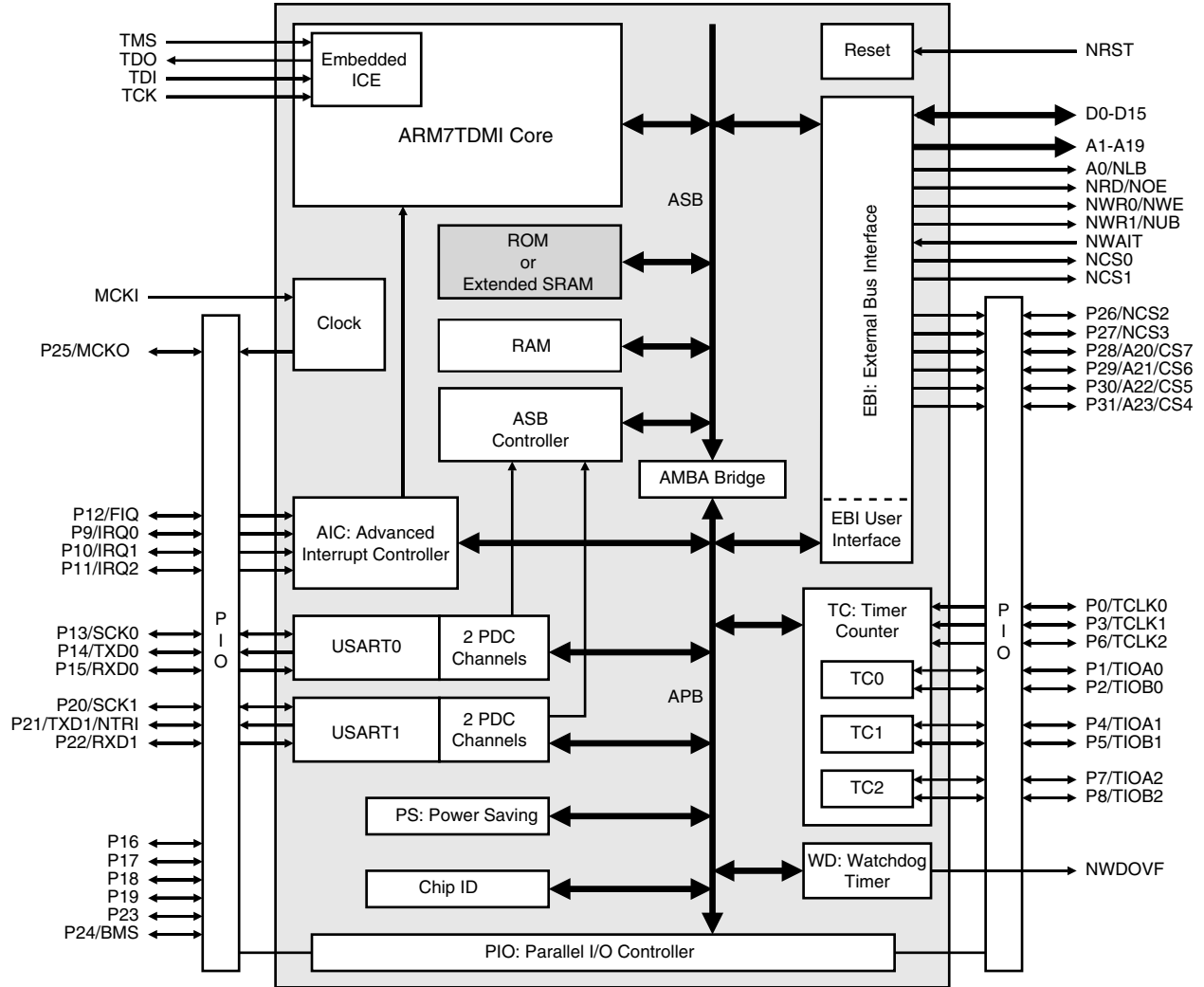


Table 1. AT91X40 Series Pin Description

| Module | Name | Function | Type | Active Level | Comments |
|--------|---------------|------------------------------|--------|--------------|-----------------------------------|
| EBI | A0 - A23 | Address Bus | Output | – | All valid after reset |
| | D0 - D15 | Data Bus | I/O | – | |
| | NCS0 - NCS3 | Chip Select | Output | Low | |
| | CS4 - CS7 | Chip Select | Output | High | A23 - A20 after reset |
| | NWR0 | Lower Byte 0 Write Signal | Output | Low | Used in Byte Write option |
| | NWR1 | Upper Byte 1 Write Signal | Output | Low | Used in Byte Write option |
| | NRD | Read Signal | Output | Low | Used in Byte Write option |
| | NWE | Write Enable | Output | Low | Used in Byte Select option |
| | NOE | Output Enable | Output | Low | Used in Byte Select option |
| | NUB | Upper Byte Select | Output | Low | Used in Byte Select option |
| | NLB | Lower Byte Select | Output | Low | Used in Byte Select option |
| | NWAIT | Wait Input | Input | Low | |
| | BMS | Boot Mode Select | Input | – | Sampled during reset |
| AIC | FIQ | Fast Interrupt Request | Input | – | PIO-controlled after reset |
| | IRQ0 - IRQ2 | External Interrupt Request | Input | – | PIO-controlled after reset |
| TC | TCLK0 - TCLK2 | Timer External Clock | Input | – | PIO-controlled after reset |
| | TIOA0 - TIOA2 | Multipurpose Timer I/O Pin A | I/O | – | PIO-controlled after reset |
| | TIOB0 - TIOB2 | Multipurpose Timer I/O Pin B | I/O | – | PIO-controlled after reset |
| USART | SCK0 - SCK1 | External Serial Clock | I/O | – | PIO-controlled after reset |
| | TXD0 - TXD1 | Transmit Data Output | Output | – | PIO-controlled after reset |
| | RXD0 - RXD1 | Receive Data Input | Input | – | PIO-controlled after reset |
| PIO | P0 - P31 | Parallel IO Line | I/O | – | |
| WD | NWDOVF | Watchdog Overflow | Output | Low | Open-drain |
| Clock | MCKI | Master Clock Input | Input | – | Schmidt trigger |
| | MCKO | Master Clock Output | Output | – | |
| Reset | NRST | Hardware Reset Input | Input | Low | Schmidt trigger |
| | NTRI | Tri-state Mode Select | Input | Low | Sampled during reset |
| ICE | TMS | Test Mode Select | Input | – | Schmidt trigger, internal pull-up |
| | TDI | Test Data Input | Input | – | Schmidt trigger, internal pull-up |
| | TDO | Test Data Output | Output | – | |
| | TCK | Test Clock | Input | – | Schmidt trigger, internal pull-up |
| Power | VDDIO | I/O Power | Power | – | |
| | VDDCORE | Core Power | Power | – | |
| | GND | Ground | Ground | – | |

Block Diagram

Figure 2. AT91X40 Series



Architectural Overview

The AT91X40 Series Microcontrollers integrate an ARM7TDMI with its embedded ICE interface, memories and peripherals. The series' architecture consists of two main buses, the Advanced System Bus (ASB) and the Advanced Peripheral Bus (APB). Designed for maximum performance and controlled by the memory controller, the ASB interfaces the ARM7TDMI processor with the on-chip 32-bit memories, the External Bus Interface (EBI) and the AMBA™ Bridge. The AMBA Bridge drives the APB, which is designed for accesses to on-chip peripherals and optimized for low-power consumption.

The AT91X40 Series Microcontrollers implement the ICE port of the ARM7TDMI processor on dedicated pins, offering a complete, low-cost and easy-to-use debug solution for target debugging.

Memories

The AT91X40 Series Microcontrollers embed up to 256K bytes of internal SRAM, and up to 128K bytes of ROM. The internal memories are directly connected to the 32-bit data bus and are single-cycle accessible. This provides maximum performance of 0.9 MIPS/MHz by using the ARM instruction set of the processor, minimizing system power consumption and improving the performance of separate memory solutions.

The AT91X40 Series Microcontrollers feature an External Bus Interface (EBI), which enables connection of external memories and application-specific peripherals. The EBI supports 8- or 16-bit devices and can use two 8-bit devices to emulate a single 16-bit device. The EBI implements the early read protocol, enabling faster memory accesses than standard memory interfaces.

Peripherals

The AT91X40 Series Microcontrollers integrate several peripherals, which are classified as system or user peripherals. All on-chip peripherals are 32-bit accessible by the AMBA Bridge, and can be programmed with a minimum number of instructions. The peripheral register set is composed of control, mode, data, status and enable/disable/status registers.

An on-chip Peripheral Data Controller (PDC) transfers data between the on-chip USARTs and on- and off-chip memories address space without processor intervention. Most importantly, the PDC removes the processor interrupt handling overhead, making it possible to transfer up to 64K continuous bytes without reprogramming the start address, thus increasing the performance of the microcontroller, and reducing the power consumption.

System Peripherals

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

The Power Saving (PS) module implements the Idle Mode (ARM7TDMI core clock stopped until the next interrupt) and enables the user to adapt the power consumption of the microcontroller to application requirements (independent peripheral clock control).

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

The Parallel Input/Output Controller (PIO) controls up to 32 I/O lines. It enables the user to select specific pins for on-chip peripheral input/output functions, and general-purpose input/output signal pins. The PIO controller can be programmed to detect an interrupt on a signal change from each line.



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

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

User Peripherals

Two USARTs, independently configurable, enable communication at a high baud rate in Synchronous or Asynchronous Mode. The format includes start, stop and parity bits and up to 8 data bits. Each USART also features a Timeout and a Time Guard register, facilitating the use of the two dedicated Peripheral Data Controller (PDC) channels.

The 3-channel, 16-bit Timer Counter (TC) is highly-programmable and supports Capture or Waveform Modes. Each TC channel can be programmed to measure or generate different kinds of waves, and can detect and control two input/output signals. The TC also has 3 external clock signals.

Associated Documentation

Table 2. Associated Documentation

| Product | Information | Document Title |
|------------|--|---------------------------------------|
| AT91M40800 | Internal architecture of processor ARM/Thumb instruction sets Embedded in-circuit-emulator | ARM7TDMI (Thumb) Datasheet |
| | Pinout Mechanical characteristics Ordering information | AT91M40800 Summary Datasheet |
| | Timings DC characteristics | AT91M40800 Electrical Characteristics |
| AT91R40807 | Internal architecture of processor ARM/Thumb instruction sets Embedded in-circuit-emulator | ARM7TDMI (Thumb) Datasheet |
| | Pinout Mechanical characteristics Ordering information | AT91R40807 Summary Datasheet |
| | Timings DC characteristics | AT91R40807 Electrical Characteristics |
| AT91M40807 | Internal architecture of processor ARM/Thumb instruction sets Embedded in-circuit-emulator | ARM7TDMI (Thumb) Datasheet |
| | Pinout Mechanical characteristics Ordering information | AT91M40807 Summary Datasheet |
| | Timings DC characteristics | AT91M40807 Electrical Characteristics |
| AT91R40008 | Internal architecture of processor ARM/Thumb instruction sets Embedded in-circuit-emulator | ARM7TDMI (Thumb) Datasheet |
| | Pinout Mechanical characteristics Ordering information | AT91R40008 Summary Datasheet |
| | Timings DC characteristics | AT91R40008 Electrical Characteristics |



Product Overview

Power Supply

The AT91x40 Series Microcontrollers have two types of power supply pins - VDDIO and VDDCORE. However, the AT91M40800, the AT91M40807 and the AT91R40807 have single-supply VDD, VDDIO and VDDCORE pins that have to be tied to the same voltage. For further details on power supplies and acceptable voltage range on VDD, VDDIO and VDDCORE, refer to the product Summary Datasheet or the product Electrical Characteristics datasheet.

Input/Output Considerations

The AT91M40807, the AT91R40807 and the AT91R40008 accept voltage levels up to their power supply limit on the pads.

The AT91M40800 Microcontroller I/O pads are 5V-tolerant, enabling it to interface with external 5V devices without any additional components. 5V-tolerant means that the AT91M40800 accepts 5V (3V) on the inputs even if it is powered at 3V (2V). Refer to the AT91M40800 Electrical Characteristics datasheet for further details.

After the reset, the peripheral I/Os are initialized as inputs to provide the user with maximum flexibility. It is recommended that in any application phase, the inputs to the AT91X40 Series Microcontroller be held at valid logic levels to minimize the power consumption.

Master Clock

The AT91X40 Series Microcontrollers have a fully static design and work on the Master Clock (MCK), provided on the MCKI pin from an external source.

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

Reset

Reset restores the default states of the user interface registers (defined in the user interface of each peripheral), and forces the ARM7TDMI to perform the next instruction fetch from address zero. Except for the program counter the ARM7TDMI registers do not have defined reset states.

NRST Pin

NRST is active low-level input. It is asserted asynchronously, but exit from reset is synchronized internally to the MCK. The signal presented on MCK must be active within the specification for a minimum of 10 clock cycles up to the rising edge of NRST, to ensure correct operation. The first processor fetch occurs 80 clock cycles after the rising edge of NRST.

Watchdog Reset

The watchdog can be programmed to generate an internal reset. In this case, the reset has the same effect as the NRST pin assertion, but the pins BMS and NTRI are not sampled. Boot Mode and Tri-state Mode are not updated. If the NRST pin is asserted and the watchdog triggers the internal reset, the NRST pin has priority.

Emulation Function

Tri-state Mode

The AT91X40 Series provides a tri-state mode, which is used for debug purposes. This enables the connection of an emulator probe to an application board without having to desolder the device from the target board. In tri-state mode, all the output pin drivers of the AT91X40 Series Microcontroller are disabled.

To enter tri-state mode, the pin NTRI must be held low during the last 10 clock cycles before the rising edge of NRST. For normal operation, the pin NTRI must be held high during reset, by a resistor of up to 400K Ohm.

NTRI is multiplexed with I/O line P21 and USART 1 serial data transmit line TXD1.

Standard RS-232 drivers generally contain internal 400K Ohm pull-up resistors. If TXD1 is connected to a device not including this pull-up, the user must make sure that a high level is tied on NTRI while NRST is asserted.

JTAG/ICE Debug

ARM standard embedded In-circuit Emulation is supported via the JTAG/ICE port. The pins TDI, TDO, TCK and TMS are dedicated to this debug function and can be connected to a host computer via the external ICE interface.

In ICE Debug Mode, the ARM7TDMI core responds with a non-JTAG chip ID that identifies the microcontroller. This is not fully IEEE1149.1 compliant.

Memory Controller

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

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

In any of these address spaces, the ARM7TDMI operates in Little-Endian Mode only.

Internal Memories

The AT91X40 Series Microcontrollers integrate one or two banks of internal static SRAM and/or one bank of ROM. All internal memories are 32 bits wide and single-clock cycle accessible. Byte (8-bit), halfword (16-bit) or word (32-bit) accesses are supported and are executed within one cycle. Fetching Thumb or ARM instructions is supported and internal memory can store twice as many Thumb instructions as ARM ones.

All the AT91X40 Series Microcontrollers integrate a primary 8-Kbyte or 256-Kbyte SRAM bank, accessible at address 0x0 (after the remap).

The AT91R40807 integrates a secondary SRAM memory bank of 128K bytes at address 0x10 0000. This secondary bank can be used to emulate the ROM of the AT91M40807.

The AT91M40807 Microcontroller integrates 128K bytes of internal ROM at address 0x10 0000. It offers a reduced-cost option of the AT91R40807 for high-volume applications in which the software is stable.

Using Internal Memories

The primary RAM bank is always mapped at address 0x30 0000 before remap and at address 0x0 after the remap, allowing ARM7TDMI exception vectors to be modified by the software. Making the RAM bank accessible before remap allows the user to copy ARM exception vectors and boot code into the bank prior to remap.

The rest of the bank can be used for stack allocation to speed up context saving and restoration, or as data and program storage for critical algorithms.

Placing the SRAM on-chip and using a 32-bit data bus bandwidth maximizes microcontroller performance while minimizing system power consumption. The 32-bit bus optimizes use of the ARM instruction set and offers the ability to process data wider than 16 bits, thus making optimal use of the ARM7TDMI advanced performance.

The capability to update application software dynamically in an internal SRAM bank adds an extra dimension to the AT91X40 Series Microcontrollers.

ROM Emulation

The AT91R40807 provides an ideal means of emulating the ROM version AT91M40807. The secondary SRAM bank of the AT91R40807 is mapped to the same address as the ROM of the AT91M40807. It is write-protected after a reset; writing 0x1 in the Memory Mode Register of the Special Function Module can disable this protection.

At system power-up, the code is downloaded from an external non-volatile memory or through a debugger to the on-chip secondary SRAM bank of the AT91R40807. After the secondary SRAM bank write-protection is enabled, the application is in the same environment as though it were running on an AT91M40807.

Boot Mode Select

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

The input level on the BMS pin during the last 10 clock cycles before the rising edge of the NRST selects the type of boot memory. The Boot Mode depends on BMS and whether or not the AT91X40 Series Microcontroller has on-chip ROM or extended SRAM (see Table 3).

The AT91R40807 supports boot in on-chip extended SRAM, for the purpose of emulating ROM versions. In this case, the microcontroller must first boot from external non-volatile memory, and ensure that a valid program is downloaded in the on-chip extended SRAM. Then, the NRST must be reasserted by external circuitry after the level on the pin BMS is changed.

The pin BMS is multiplexed with the I/O line P24 that can be programmed after reset like any standard PIO line.

Table 3. Boot Mode Select

| BMS | Product | Boot Memory |
|-----|------------|--------------------------------|
| 1 | AT91M40800 | External 8-bit memory on NCS0 |
| | AT91R40807 | Internal 32-bit extended SRAM |
| | AT91M40807 | Internal 32-bit ROM |
| | AT91R40008 | External 8-bit memory on NCS0 |
| 0 | All | External 16-bit memory on NCS0 |

Remap Command

The ARM vectors (Reset, Abort, Data Abort, Prefetch Abort, Undefined Instruction, Interrupt, Fast Interrupt) are mapped from address 0x0 to address 0x20. In order to allow these vectors to be redefined dynamically by the software, the AT91X40 Series Microcontrollers use a remap command that enables switching between the boot memory and the internal primary SRAM bank addresses. The remap command is accessible through the EBI User Interface, by writing one in RCB of EBI_RCR (Remap Control Register). Performing a remap command is mandatory if access to the other external devices (connected to chip selects 1 to 7) is required. The remap operation can only be changed back by an internal reset or an NRST assertion.

Abort Control

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

- When accessing an undefined address in the EBI address space
- When writing to a write-protected internal memory area on the AT91R40807

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

When a write-protected area is accessed, the memory controller detects it and generates an abort but does not cancel the access.

External Bus Interface

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

For each of these banks, the user can program:

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

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

Peripherals

The AT91X40 Series' peripherals are connected to the 32-bit wide Advanced Peripheral Bus. Peripheral registers are only word accessible – byte and half-word accesses are not supported. If a byte or a half-word access is attempted, the memory controller automatically masks the lowest address bits and generates a word access.

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

Peripheral Registers

The following registers are common to all peripherals:

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

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

Peripheral Interrupt Control

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

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

Peripheral Data Controller

The AT91X40 Series Microcontroller has a 4-channel PDC dedicated to the two on-chip USARTs. One PDC channel is dedicated to the receiver and one to the transmitter of each USART.

The user interface of a PDC channel is integrated in the memory space of each USART. It contains a 32-bit Address Pointer Register (RPR or TPR) and a 16-bit Transfer Counter Register (RCR or TCR). When the programmed number of transfers are performed, a status bit indicating the end of transfer is set in the USART Status Register and an interrupt can be generated.

System Peripherals

PS: Power-saving

The Power-saving feature optimizes power consumption, enabling the software to stop the ARM7TDMI clock (Idle Mode) and restarting it when the module receives an interrupt (or reset). It also enables on-chip peripheral clocks to be enabled and disabled individually, matching power consumption and application needs.

AIC: Advanced Interrupt Controller

The AIC has an 8-level priority, individually maskable, vectored interrupt controller, and drives the NIRQ and NFIQ pins of the ARM7TDMI from:

- The external fast interrupt line (FIQ)
- The three external interrupt request lines (IRQ0 - IRQ2)
- The interrupt signals from the on-chip peripherals

The AIC is extensively programmable, offering maximum flexibility, and its vectoring features reduce the real-time overhead in handling interrupts.

The AIC also features a spurious vector, which reduces spurious interrupt handling to a minimum, and a protect mode that facilitates the debug capabilities.

PIO: Parallel IO Controller

The AT91X40 Series has 32 programmable I/O lines. Six pins are dedicated as general-purpose I/O pins. Other I/O lines are multiplexed with an external signal of a peripheral to optimize the use of available package pins. The PIO controller enables generation of an interrupt on input change and insertion of a simple input glitch filter on any of the PIO pins.

WD: Watchdog

The Watchdog is built around a 16-bit counter, and is used to prevent system lock-up if the software becomes trapped in a deadlock. It can generate an internal reset or interrupt, or assert an active level on the dedicated pin NWD0VF. All programming registers are password-protected to prevent unintentional programming.

SF: Special Function

The AT91X40 Series provides registers that implement the following special functions.

- Chip identification
- RESET status
- Protect Mode
- Write protection for the AT91R40807 internal 128-Kbyte memory

User Peripherals

USART: Universal Synchronous/Asynchronous Receiver Transmitter

The AT91X40 Series provides two identical, full-duplex, universal synchronous/asynchronous receiver/transmitters.

Each USART has its own baud rate generator, and two dedicated Peripheral Data Controller channels. The data format includes a start bit, up to 8 data bits, an optional programmable parity bit and up to 2 stop bits.

The USART also features a Receiver Timeout register, facilitating variable length Frame support when it is working with the PDC, and a Time Guard register, used when interfacing with slow remote equipment.

TC: Timer Counter

The AT91X40 Series features a Timer Counter block that 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.

The Timer Counter can be used in Capture or Waveform Mode, and all three counter channels can be started simultaneously and chained together.

Memory Map

Figure 3. AT91M40800/R40008 Memory Map Before and After the Remap Command

| Before | | | | After | | | |
|------------|---|----------|---------------|------------|----------------------------------|---|---------------|
| Address | Function | Size | Abort Control | Address | Function | Size | Abort Control |
| 0xFFFFFFF | On-chip Peripherals | 4M Bytes | No | 0xFFFFFFF | On-chip Peripherals | 4M Bytes | No |
| 0xFFC00000 | Reserved | | Yes | 0xFFC00000 | External Devices (Up to 8) | Up to 8 Devices Programmable Page Size 1, 4, 16, 64M Bytes | Yes |
| 0xFFBFFFF | | | | | | | |
| 0x00400000 | On-chip Primary RAM Bank | 1M Byte | No | 0x00400000 | Reserved | 1M Byte | No |
| 0x003FFFF | | | | | | | |
| 0x00300000 | Reserved On-chip Device | 1M Byte | No | 0x00300000 | Reserved On-chip Device | 1M Byte | No |
| 0x002FFFF | | | | | | | |
| 0x00200000 | Reserved On-chip Device | 1M Byte | No | 0x00200000 | Reserved On-chip Device | 1M Byte | No |
| 0x001FFFF | | | | | | | |
| 0x00100000 | External Devices Selected by NCS0 | 1M Byte | No | 0x00100000 | On-chip Primary RAM Bank | 1M Byte | No |
| 0x000FFFF | | | | | | | |
| 0x00000000 | | | | 0x00000000 | | | |

Figure 4. AT91R40807/M40807 Before and After the Remap Command

| Before | | | | After | | | |
|-------------|---|----------|--|------------|---|---|---|
| Address | Function | Size | Abort Control | Address | Function | Size | Abort Control |
| 0xFFFFFFF | On-chip Peripherals | 4M Bytes | No | 0xFFFFFFF | On-chip Peripherals | 4M Bytes | No |
| 0xFFC00000 | Reserved | | Yes | 0xFFC00000 | External Devices (Up to 8) | Up to 8 Devices Programmable Page Size 1, 4, 16, 64M Bytes | Yes |
| 0xFFBFFFFFF | | | | | | | |
| 0x00400000 | On-chip Primary RAM Bank | 1M Byte | No | 0x00400000 | Reserved | 1M Byte | No |
| 0x003FFFFFF | | | | | | | |
| 0x00300000 | Reserved On-chip Device | 1M Byte | No | 0x00300000 | Reserved On-chip Device | 1M Byte | No |
| 0x002FFFFFF | | | | | | | |
| 0x00200000 | On-chip ROM or Secondary RAM Bank | 1M Byte | Yes (AT91R40807, If Write-protect Feature is Enabled) | 0x00200000 | On-chip ROM or Secondary RAM Bank | 1M Byte | Yes (AT91R40807, If Write-protect Feature is Enabled) |
| 0x001FFFFFF | | | | | | | |
| 0x00100000 | External Device Selected by NCS0 or On-chip ROM or Secondary RAM Bank | 1M Byte | No | 0x00100000 | On-chip Primary RAM Bank | 1M Byte | No |
| 0x000FFFFFF | | | | | | | |
| 0x00000000 | | | | 0x00000000 | | | |

Peripheral Memory Map

Figure 5. Peripheral Memory Map

| Address | Peripheral | Peripheral Name | Size |
|------------|------------|--|-----------|
| 0xFFFFFFFF | AIC | Advanced Interrupt Controller | 4K Bytes |
| 0xFFFFF000 | | Reserved | |
| 0xFFFFBFFF | WD | WatchdogTimer | 16K Bytes |
| 0xFFFF8000 | | | |
| 0xFFFF7FFF | PS | Power Saving | 16K Bytes |
| 0xFFFF4000 | | | |
| 0xFFFF3FFF | PIO | Parallel I/O Controller | 16K Bytes |
| 0xFFFF0000 | | Reserved | |
| 0xFFFE3FFF | TC | Timer Counter | 16K Bytes |
| 0xFFFE0000 | | Reserved | |
| 0xFFFD3FFF | USART0 | Universal Synchronous/ Asynchronous Receiver/Transmitter 0 | 16K Bytes |
| 0xFFFD0000 | | | |
| 0xFFFCFFFF | USART1 | Universal Synchronous/ Asynchronous Receiver/Transmitter 1 | 16K Bytes |
| 0xFFFC0000 | | Reserved | |
| 0xFFF03FFF | SF | Special Function | 16K Bytes |
| 0xFFF00000 | | Reserved | |
| 0xFFE03FFF | EBI | External Bus Interface | 16K Bytes |
| 0xFFE00000 | | | |
| 0xFFC00000 | | Reserved | |

EBI: External Bus Interface

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

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

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

The main features are:

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

The “EBI User Interface” is described on page 45.

External Memory Mapping

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

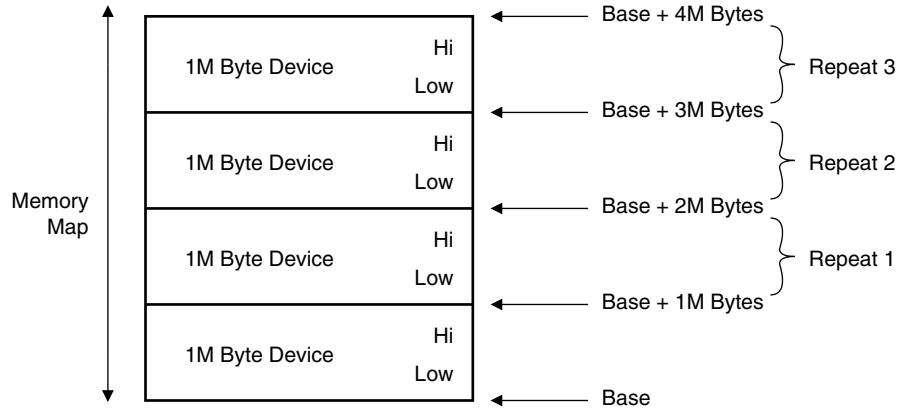
The memory map is defined by programming the base address and page size of the external memories (see “EBI User Interface” registers EBI_CSR0 to EBI_CSR7). Note that A0 - A23 is only significant for 8-bit memory; A1 - A23 is used for 16-bit memory.

If the physical memory device is smaller than the programmed page size, it wraps around and appears to be repeated within the page. The EBI correctly handles any valid access to the memory device within the page (see Figure 6).

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

If two chip selects are defined as having the same base address, an access to the overlapping address space asserts both NCS lines. The Chip Select Register with the smaller number defines the characteristics of the external access and the behavior of the control signals.

Figure 6. External Memory Smaller than Page Size



External Bus Interface Pin Description

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

The following table shows how certain EBI signals are multiplexed:

Table 4. EBI Signals

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

Chip Select Lines

The EBI provides up to eight chip select lines:

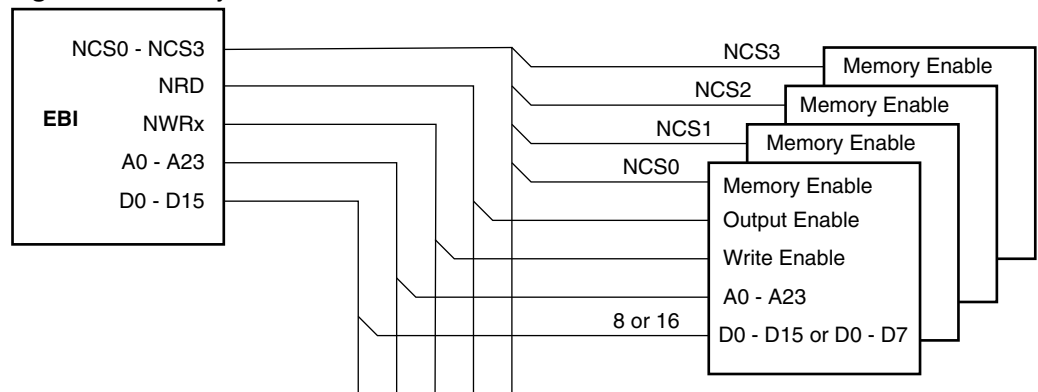
- Chip select lines NCS0 - NCS3 are dedicated to the EBI (not multiplexed).
- Chip select lines CS4 - CS7 are multiplexed with the top four address lines A23 - A20.

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

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

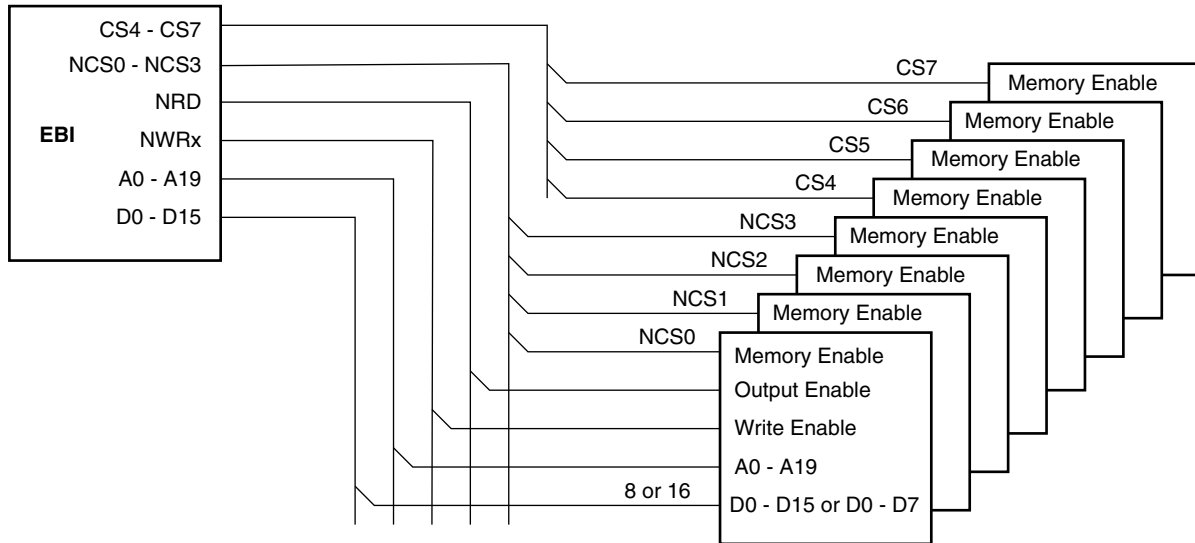
- A20, A21, A22, A23 (configuration by default)
- A20, A21, A22, CS4
- A20, A21, CS5, CS4
- A20, CS6, CS5, CS4
- CS7, CS6, CS5, CS4

Figure 7. Memory Connections for Four External Devices



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

Figure 8. Memory Connections for Eight External Devices



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

Data Bus Width

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

Figure 9 shows how to connect a 512K x 8-bit memory on NCS2.

Figure 9. Memory Connection for an 8-bit Data Bus

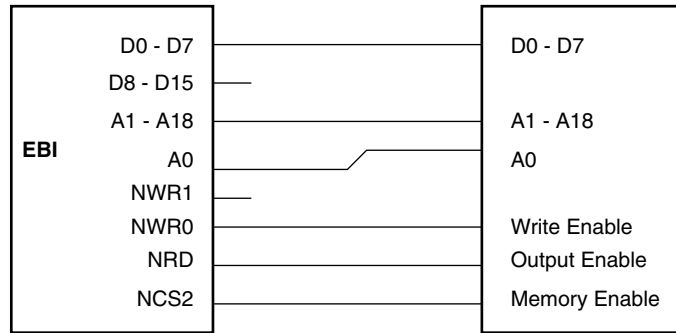
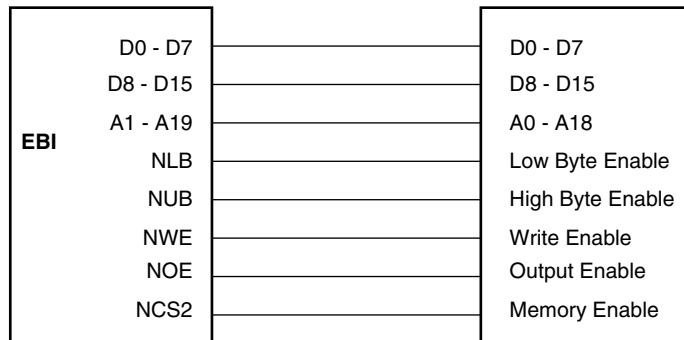


Figure 10 shows how to connect a 512K x 16-bit memory on NCS2.

Figure 10. Memory Connection for a 16-bit Data Bus



Byte Write or Byte Select Access

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

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

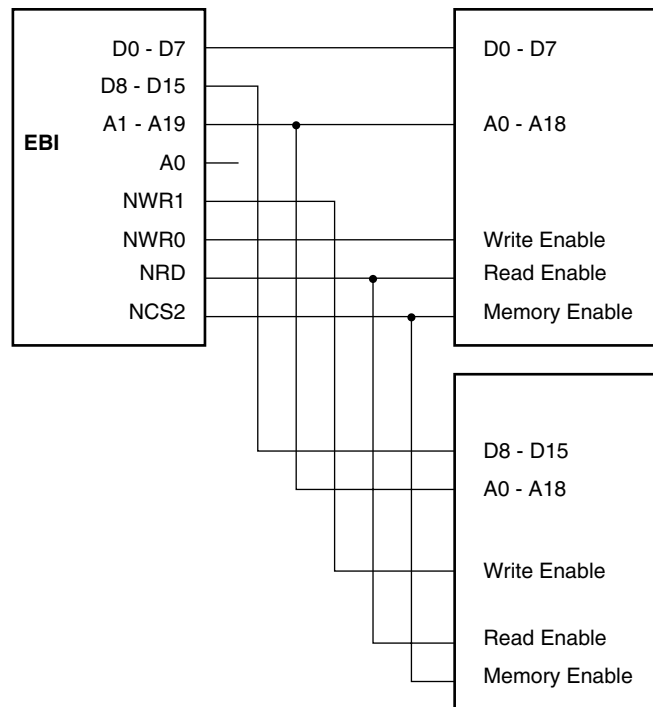
This option is controlled by the BAT field in the EBI_CSR (Chip Select Register) for the corresponding chip select.

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

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

Figure 11 shows how to connect two 512K x 8-bit devices in parallel on NCS2.

Figure 11. Memory Connection for 2 x 8-bit Data Buses



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

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

Figure 12 shows how to connect a 16-bit device with byte and half-word access (e.g. 16-bit SRAM) on NCS2.

Figure 12. Connection for a 16-bit Data Bus with Byte and Half-word Access

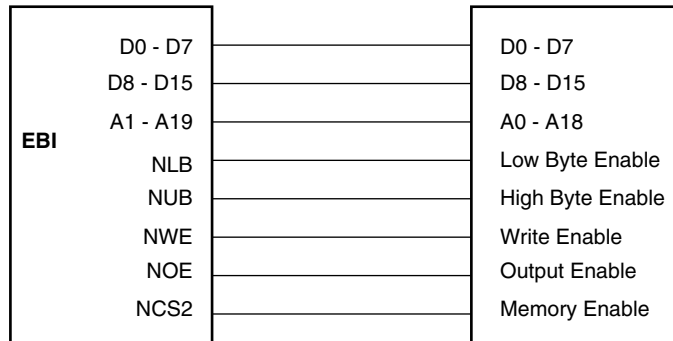
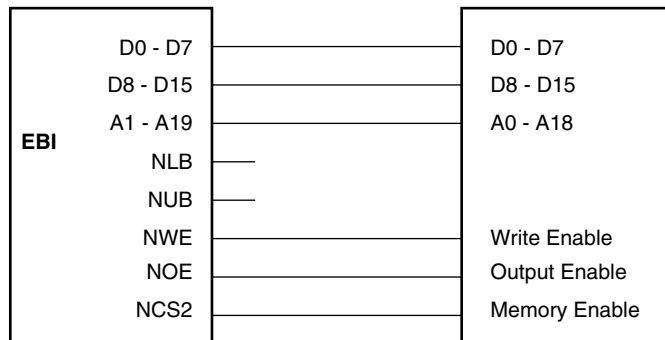


Figure 13 shows how to connect a 16-bit device without byte access (e.g. Flash) on NCS2.

Figure 13. Connection for a 16-bit Data Bus without Byte Write Capability.



Boot on NCS0

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

- 8 wait states (WSE = 1, NWS = 7)
- 8-bit or 16-bit data bus width, depending on BMS

Byte access type and number of data float time are respectively set to Byte Write Access and 0. With a non-volatile memory interface, any values can be programmed for these parameters.

Before the remap command, the user can modify the chip select 0 configuration, programming the EBI_CSR0 with exact boot memory characteristics. the base address becomes effective after the remap command, but the new number of wait states can be changed immediately. This is useful if a boot sequence needs to be faster.

Read Protocols

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

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

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

Standard Read Protocol

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

During a standard read protocol, external memory access, NCS is set low and ADDR is valid at the beginning of the access while NRD goes low only in the second half of the master clock cycle to avoid bus conflict (see Figure 14). NWE is the same in both protocols. NWE always goes low in the second half of the master clock cycle (see Figure 15).

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.

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 16). This wait state is generated in addition to any other programmed wait states (i.e. data float wait).

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

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

Figure 14. Standard Read Protocol

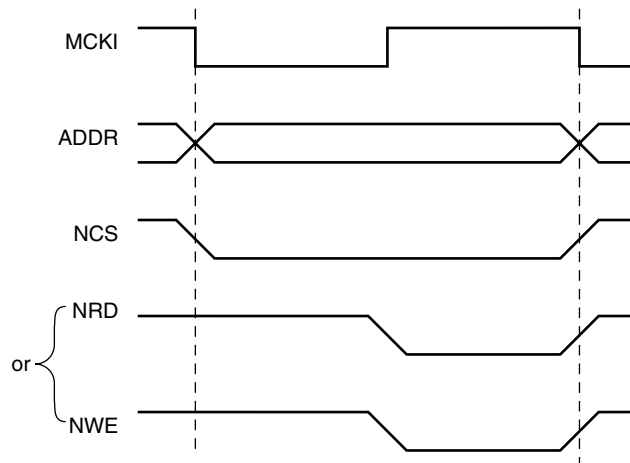


Figure 15. Early Read Protocol

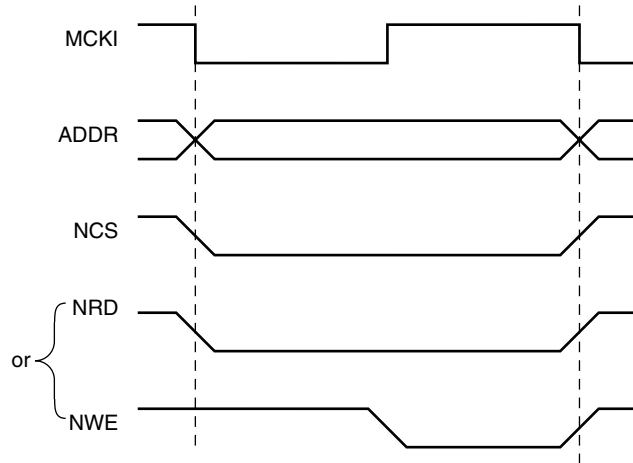
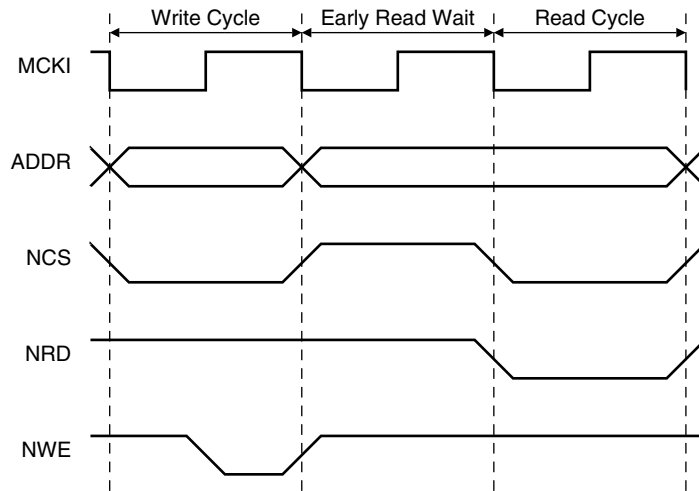


Figure 16. Early Read Wait State

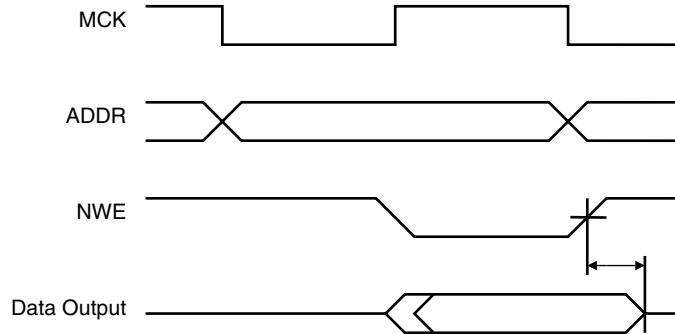


Write Data Hold Time

During write cycles in both protocols, output data becomes valid after the falling edge of the NWE signal and remains valid after the rising edge of NWE, as illustrated in Figure 17. The external NWE waveform (on the NWE pin) is used to control the output data timing to guarantee this operation.

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

Figure 17. Data Hold Time



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

Wait States

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

- Standard wait states
- Data float wait states
- External wait states
- Chip select change wait states
- Early read wait states (see “Read Protocols”)

Standard Wait States

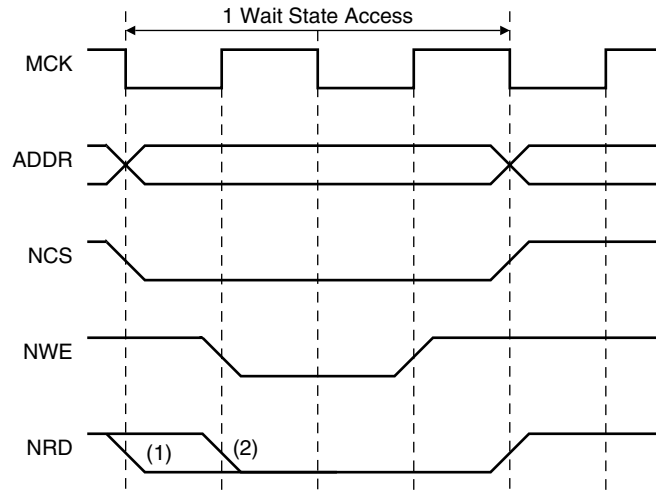
Each chip select can be programmed to insert one or more wait states during an access on the corresponding device. This is done by setting the WSE field in the corresponding EBI_CSR. The number of cycles to insert is programmed in the NWS field in the same register.

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

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

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

Figure 18. One Wait State Access



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

Data Float Wait State

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

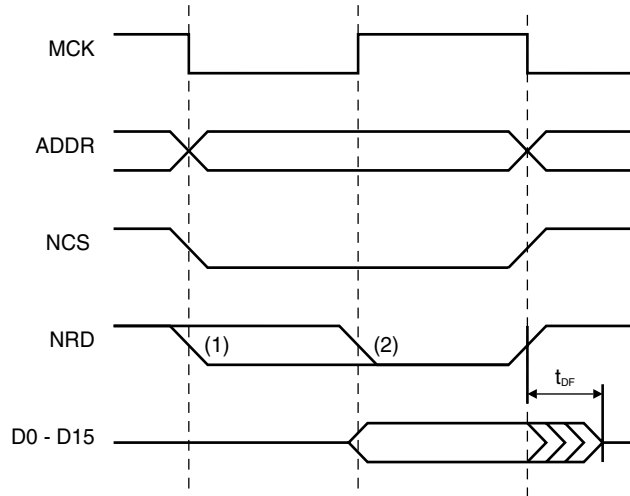
The Data Float Output Time (t_{DF}) for each external memory device is programmed in the TDF field of the EBI_CSR register for the corresponding chip select. The value (0 - 7 clock cycles) indicates the number of data float waits to be inserted and represents the time allowed for the data output to go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

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

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

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

Figure 19. Data Float Output Time



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

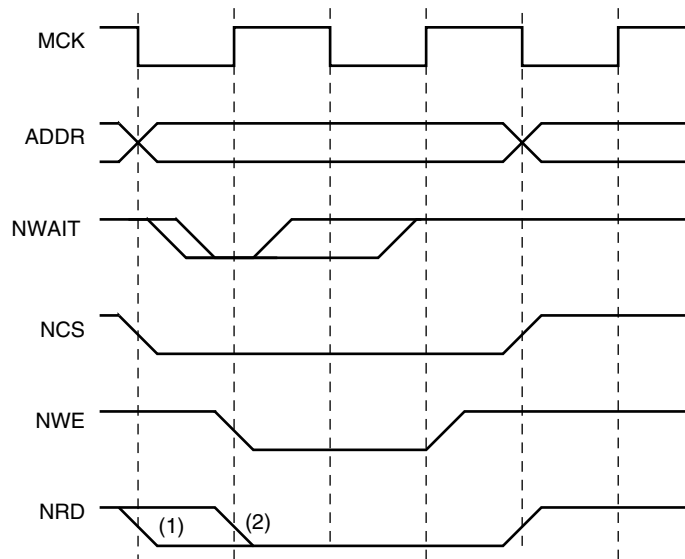
External Wait

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

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

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

Figure 20. External Wait



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

Additional constraints are applicable to the AT91R40807, the AT91M40807 and the AT91 40800. The behavior of the EBI is correct when NWAIT is asserted during an external memory access:

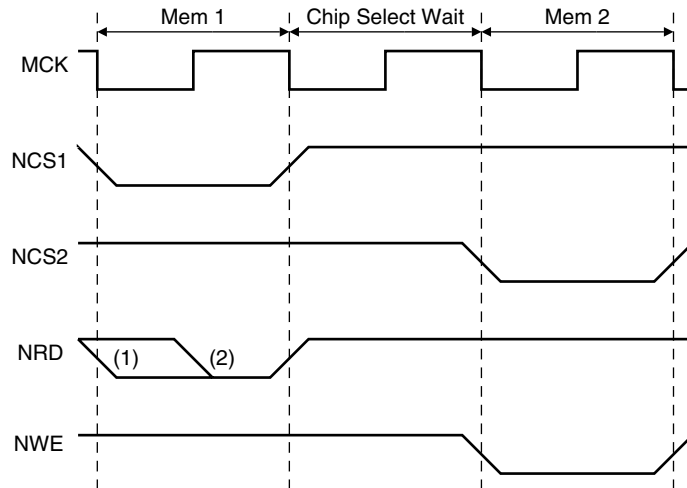
- When NWAIT is asserted before the first rising edge of MCKI
- When NWAIT is de-asserted and at least one standard wait state remains to be executed

These constraints are not applicable to the AT91R40008.

Chip Select Change Wait States

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

Figure 21. Chip Select Wait



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

Memory Access Waveforms

Figures 22 through 25 show examples of the two alternative protocols for external memory read access.

Figure 22. Standard Read Protocol without t_{DF}

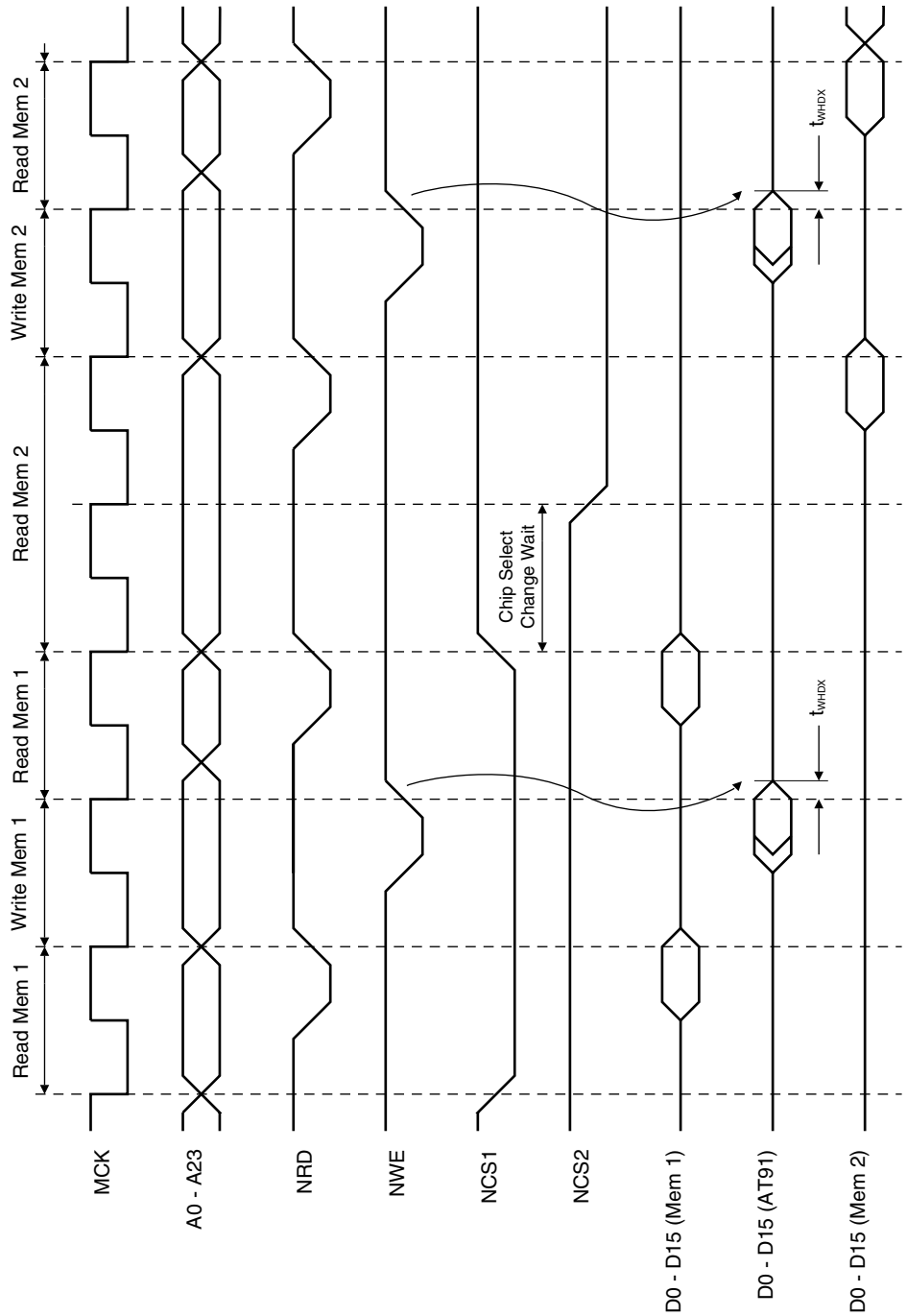


Figure 23. Early Read Protocol Without t_{DF}

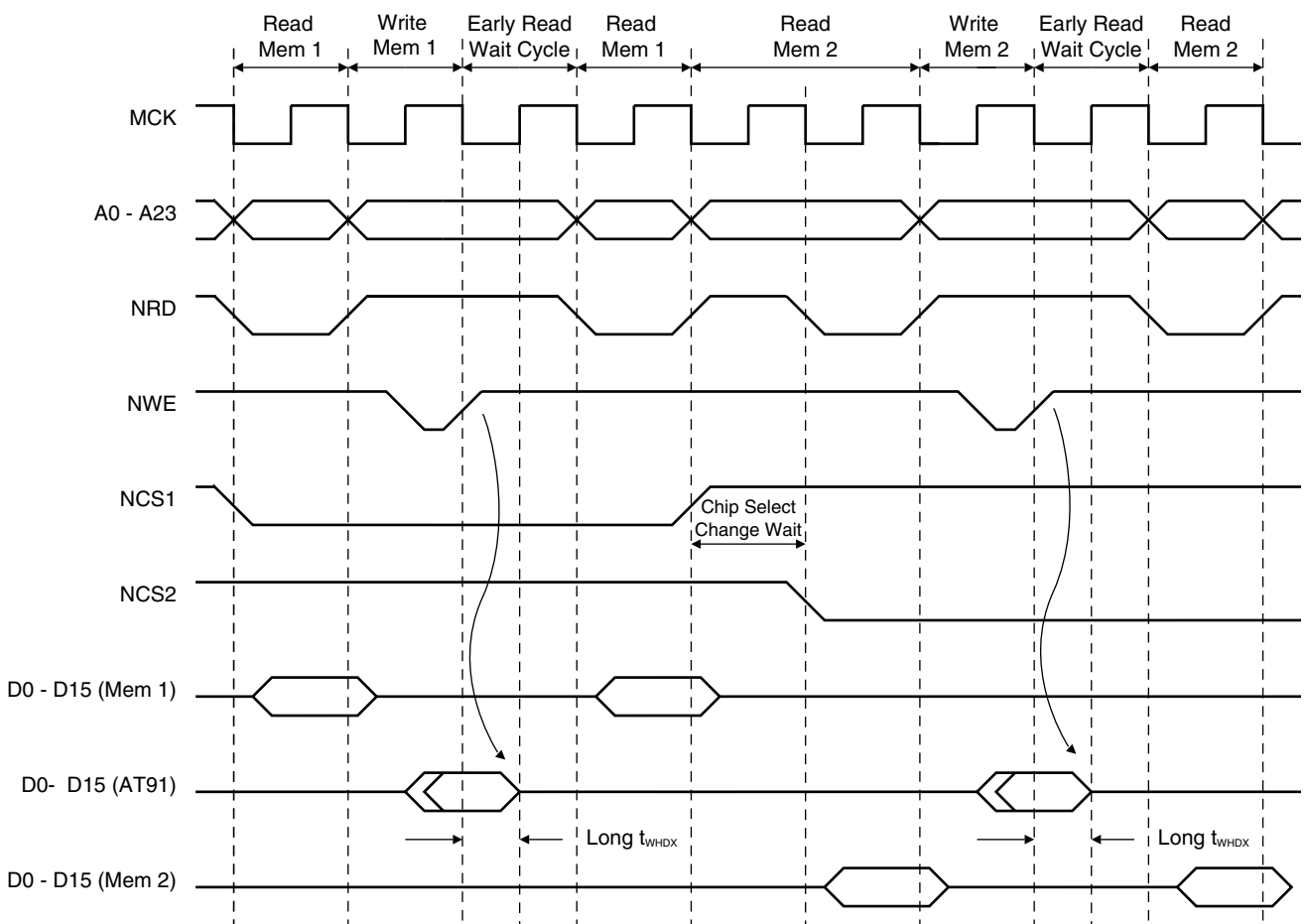


Figure 24. Standard Read Protocol with t_{DF}

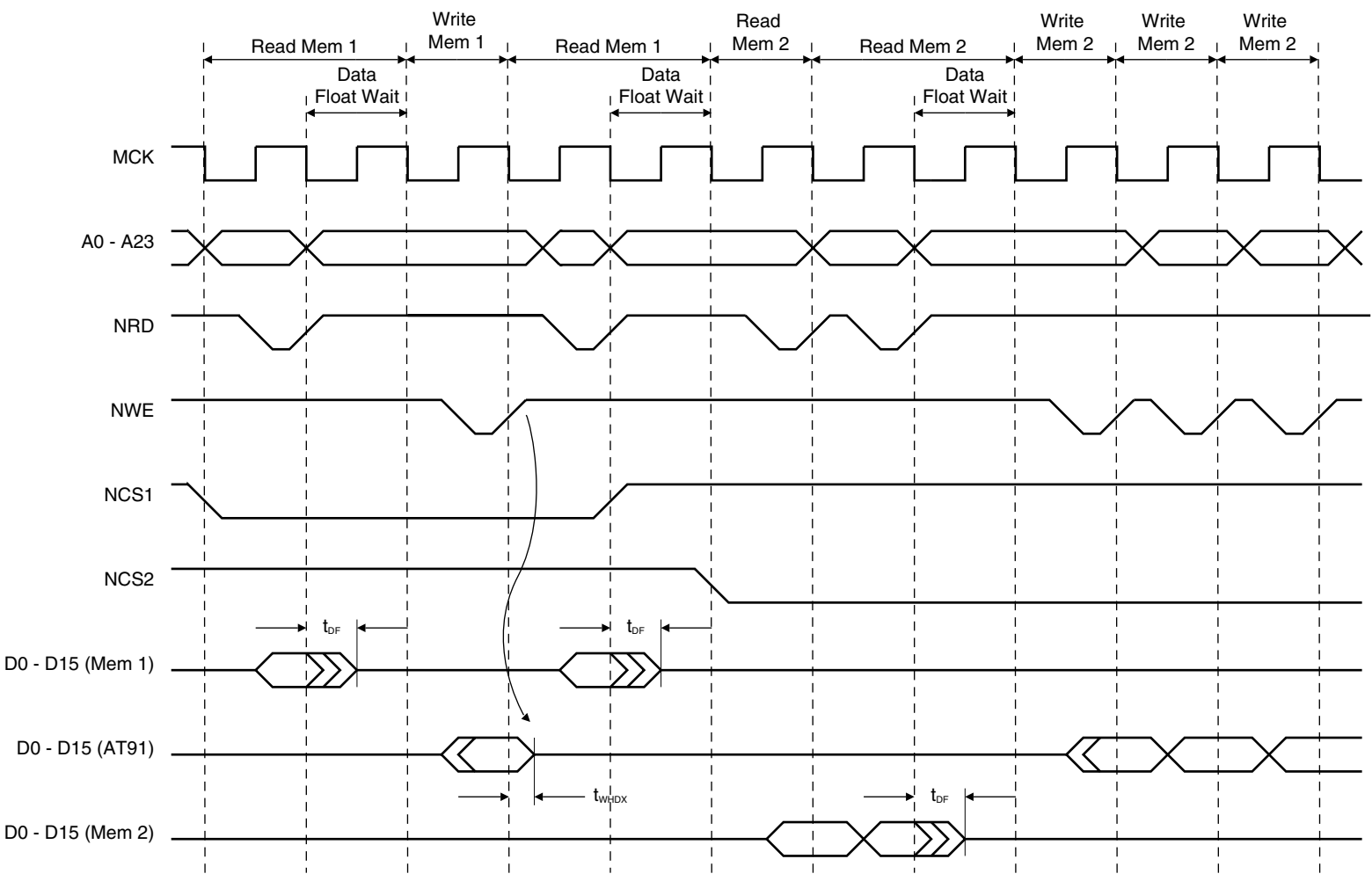
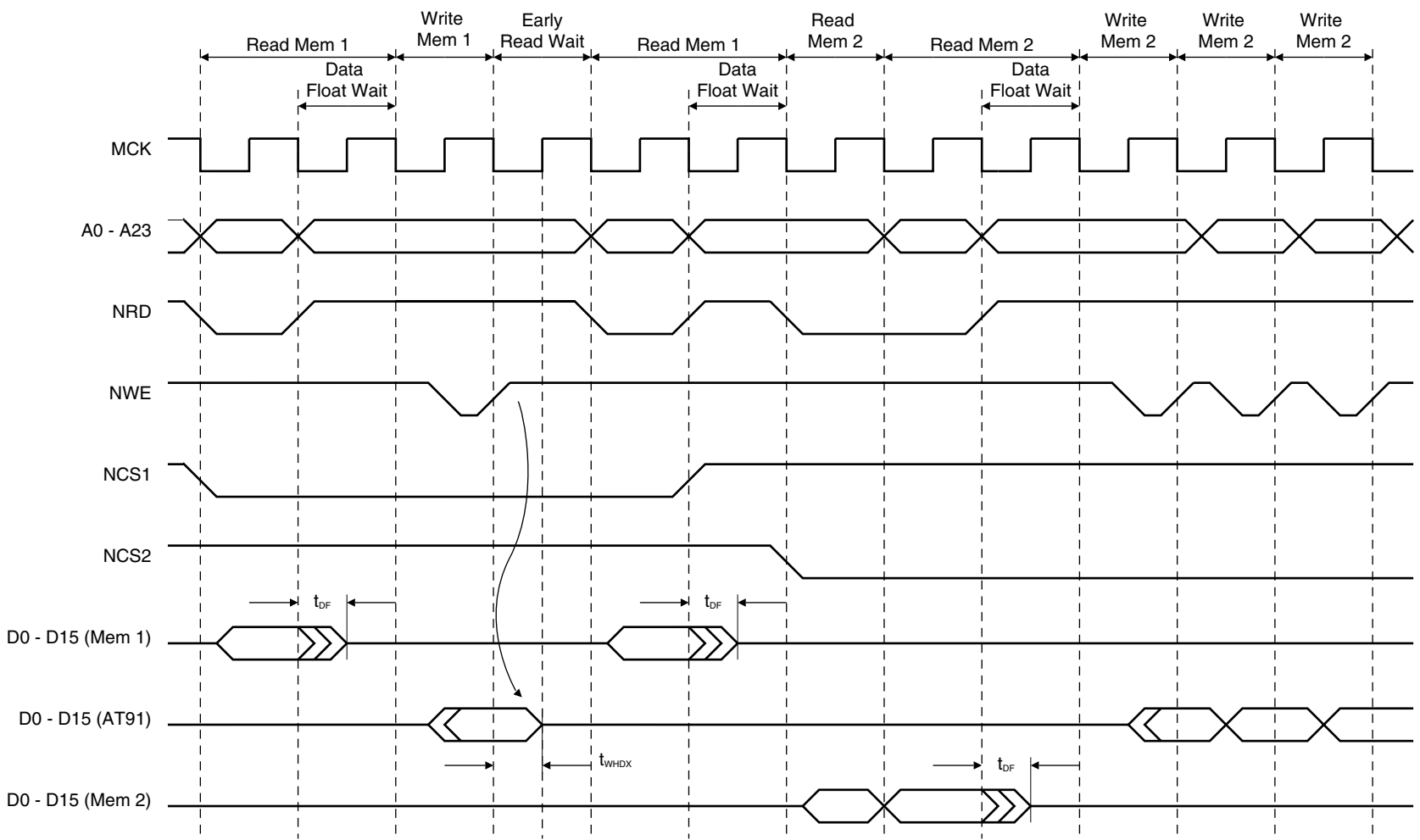


Figure 25. Early Read Protocol With t_{DF}



Figures 26 through 32 show the timing cycles and wait states for read and write access to the various AT91X40 Series external memory devices. The configurations described are shown in the following table:

Table 5. Memory Access Waveforms

| Figure Number | Number of Wait States | Bus Width | Size of Data Transfer |
|---------------|-----------------------|-----------|-----------------------|
| 26 | 0 | 16 | Word |
| 27 | 1 | 16 | Word |
| 28 | 1 | 16 | Half-word |
| 29 | 0 | 8 | Word |
| 30 | 1 | 8 | Half-word |
| 31 | 1 | 8 | Byte |
| 32 | 0 | 16 | Byte |

Figure 26. 0 Wait States, 16-bit Bus Width, Word Transfer

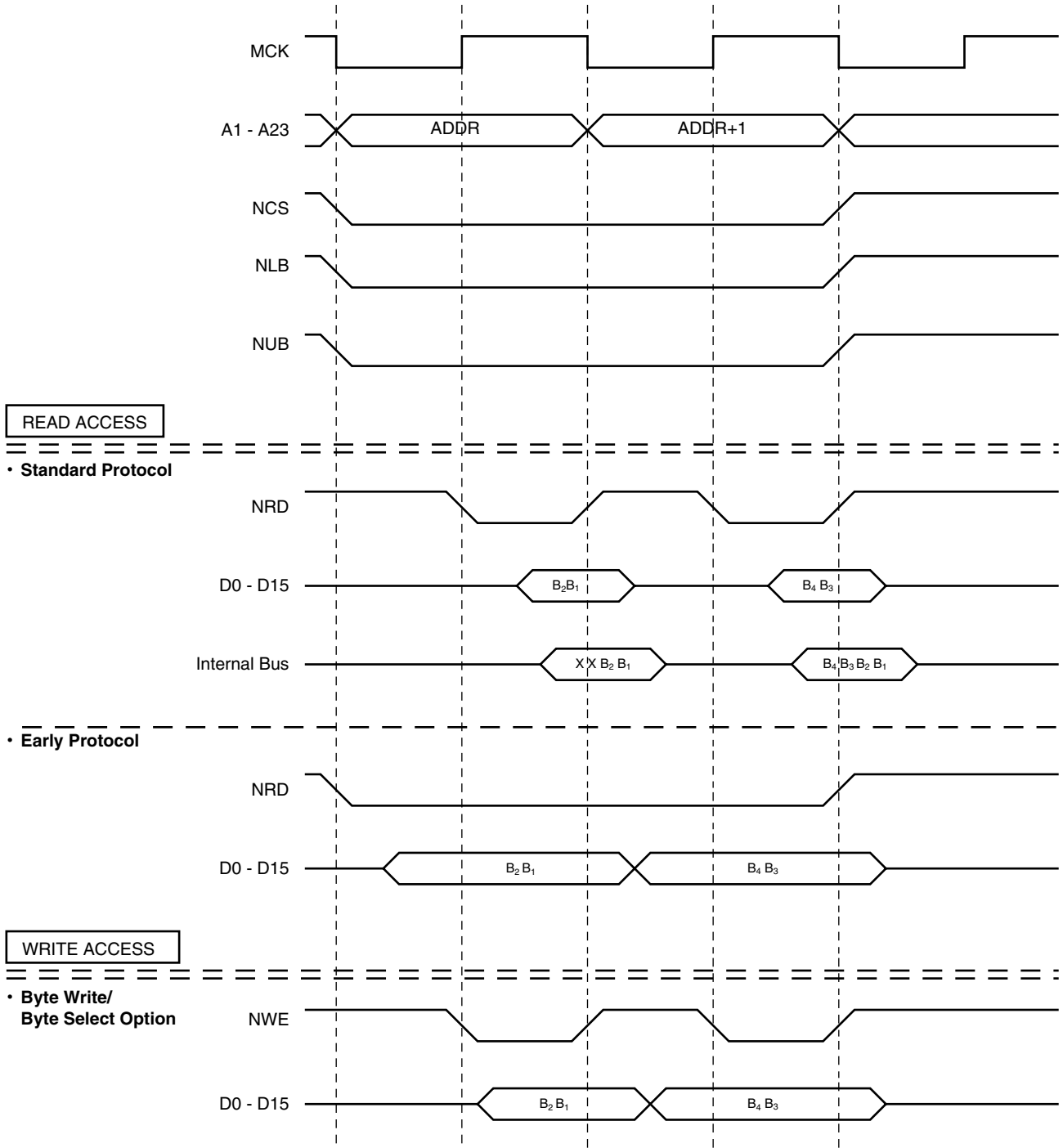


Figure 27. 1 Wait, 16-bit Bus Width, Word Transfer

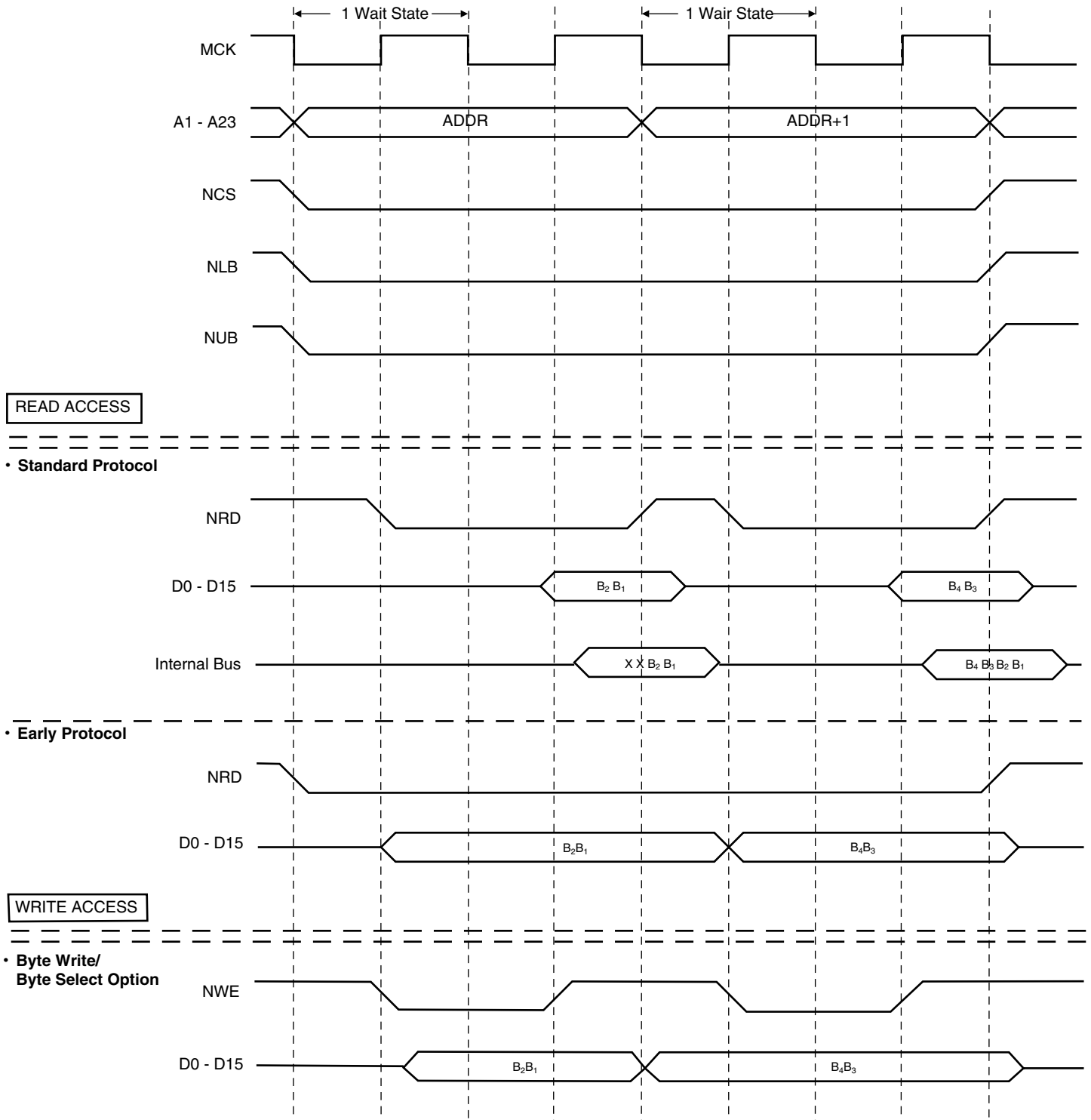


Figure 28. 1 Wait State, 16-bit Bus Width, Half-word Transfer

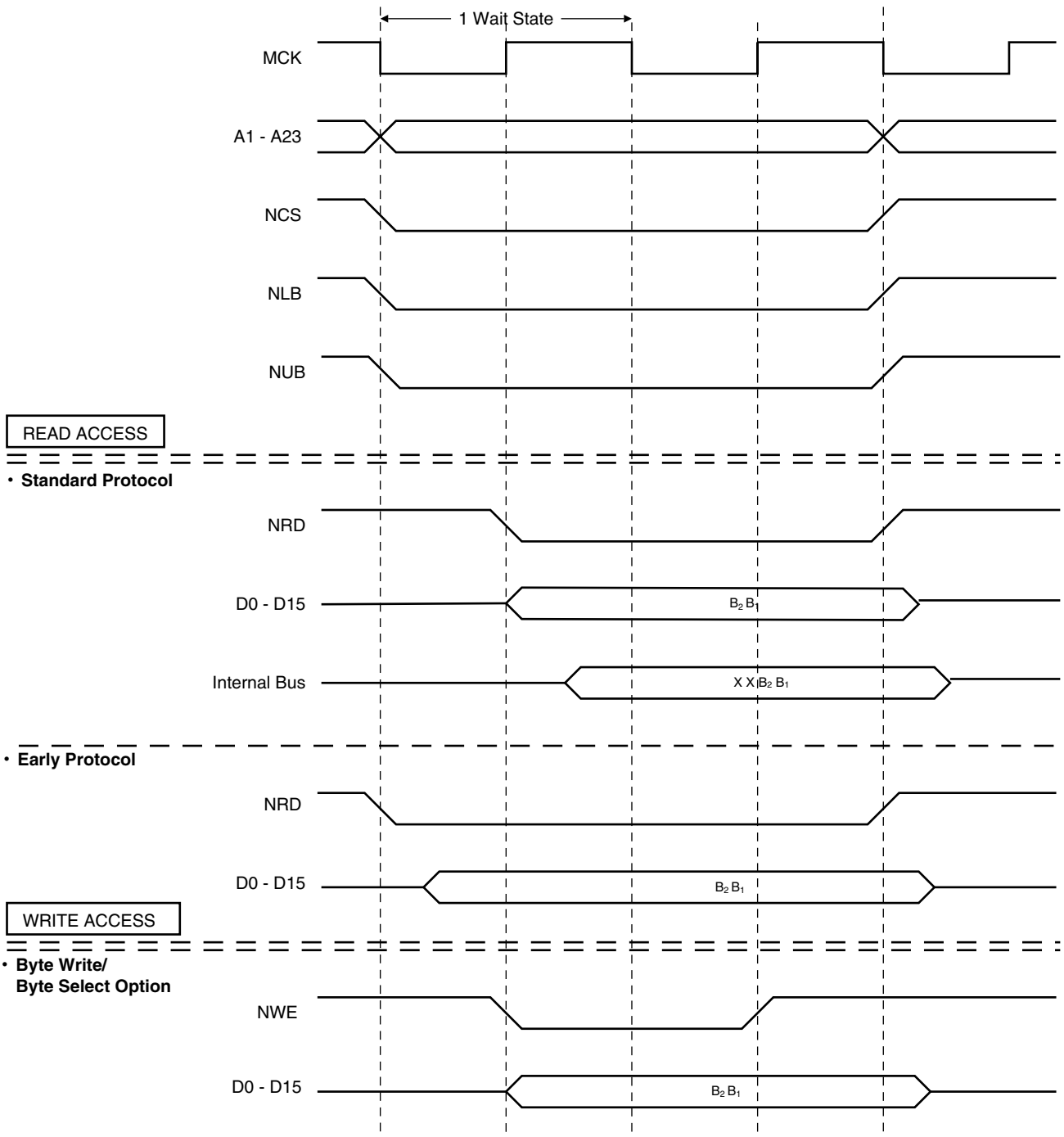


Figure 29. 0 Wait States, 8-bit Bus Width, Word Transfer

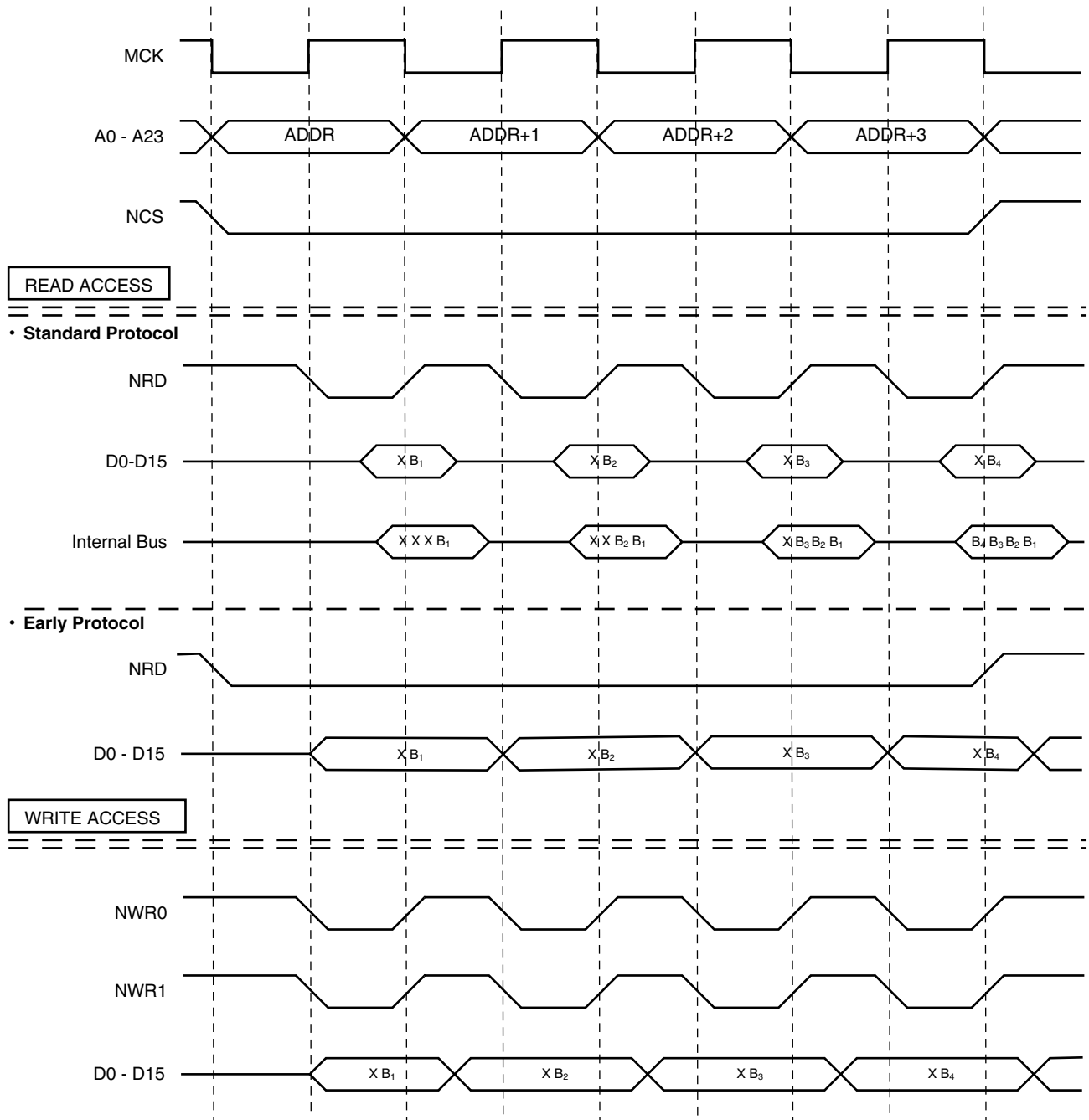


Figure 30. 1 Wait State, 8-bit Bus Width, Half-word Transfer

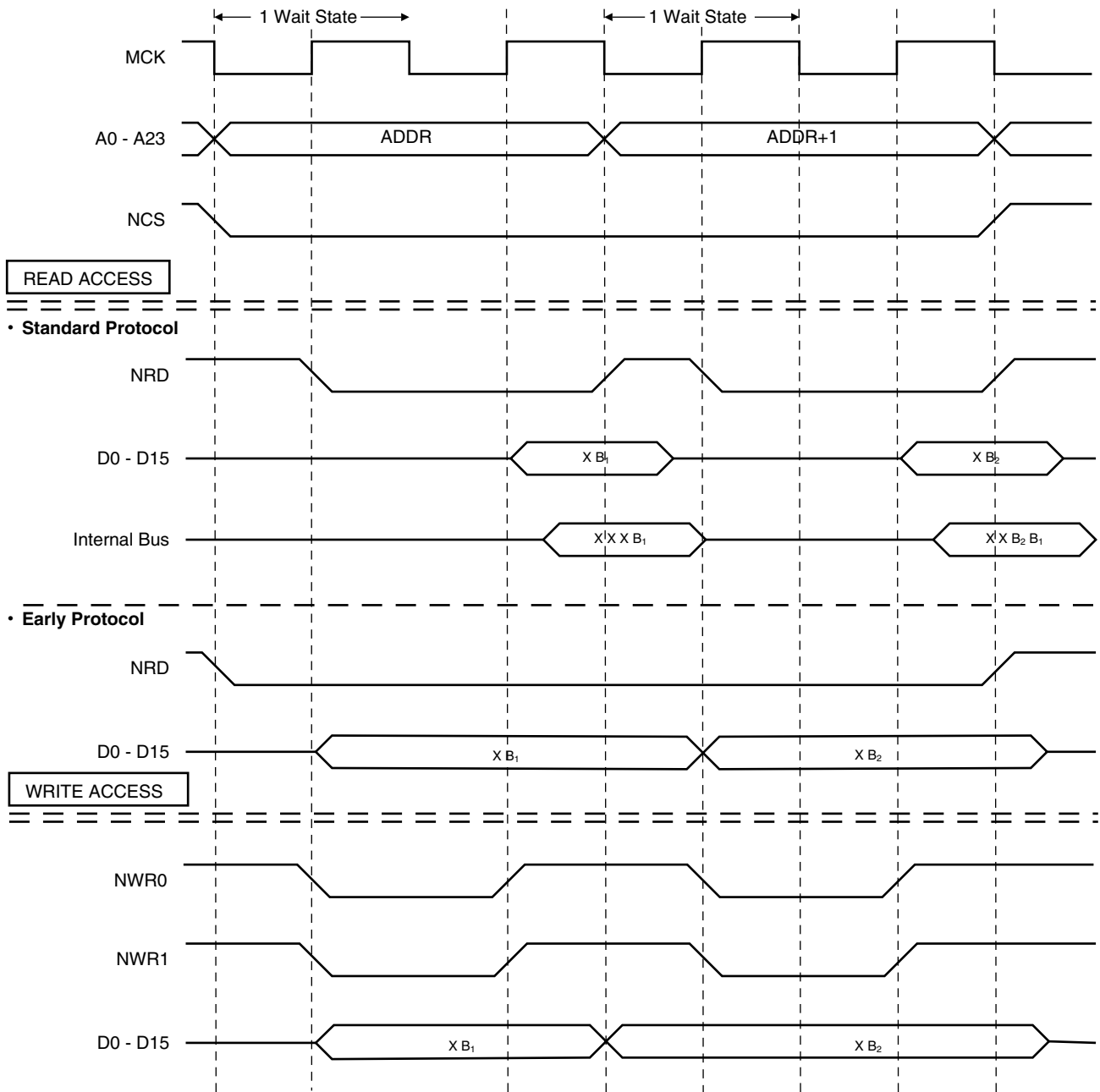


Figure 31. 1 Wait State, 8-bit Bus Width, Byte Transfer

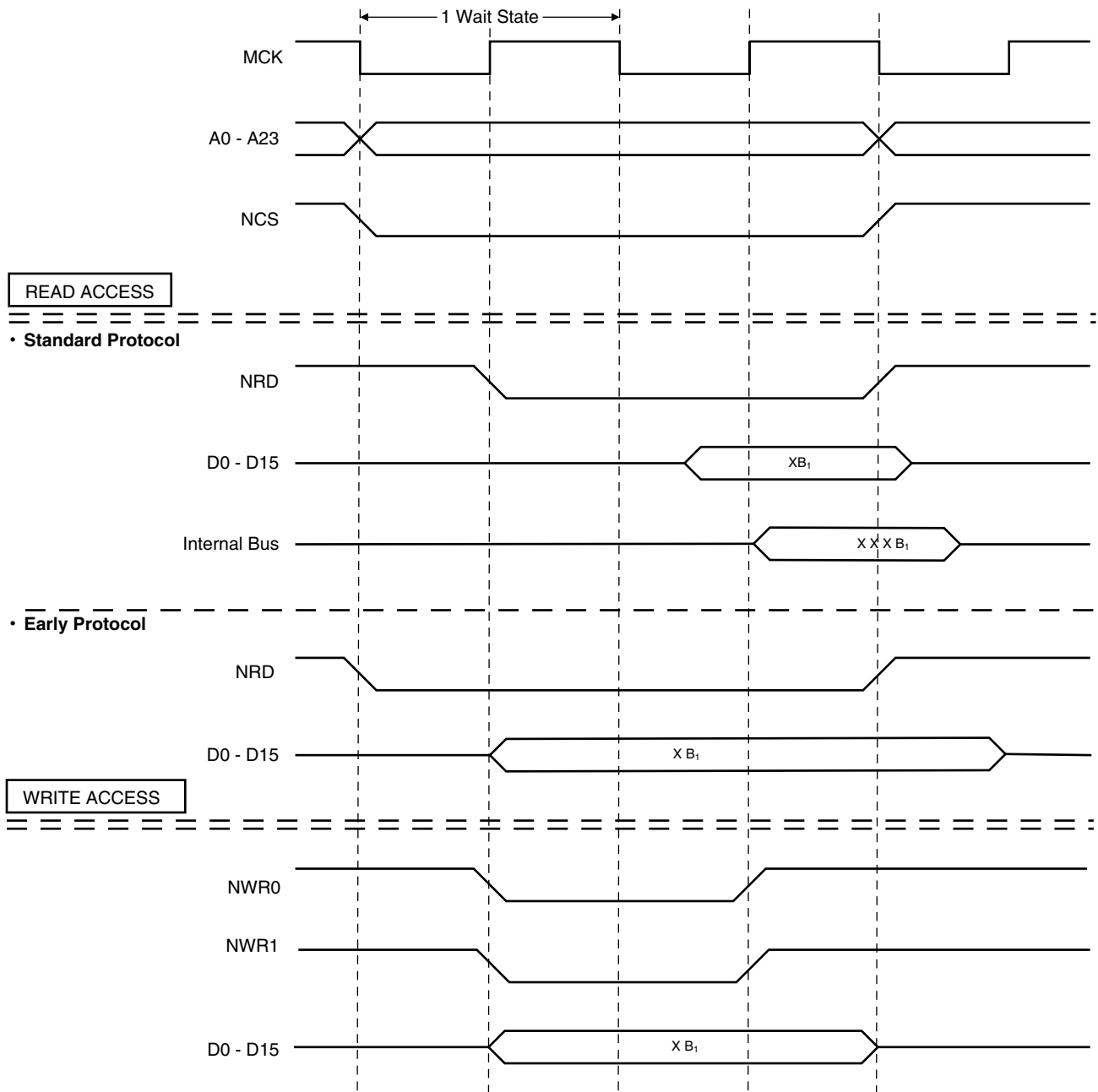
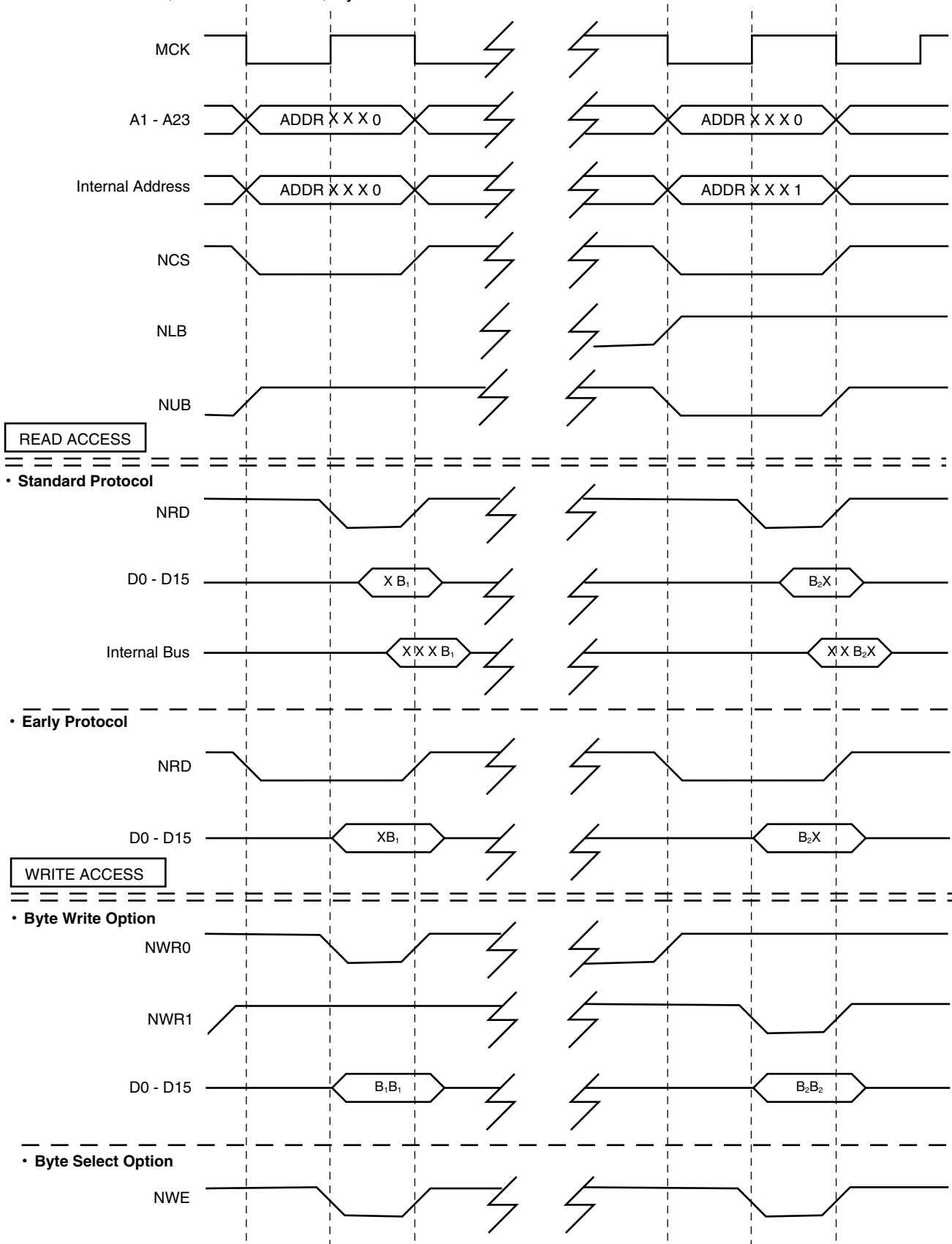


Figure 32. 0 Wait States, 16-bit Bus Width, Byte Transfer



EBI User Interface

The EBI is programmed using the registers listed in the table below. The Remap Control Register (EBI_RCR) controls exit from Boot Mode (See “Boot on NCS0” on page 26.) The Memory Control Register (EBI_MCR) is used to program the number of active chip selects and data read protocol. Eight Chip Select Registers (EBI_CSR0 to EBI_CSR7) are used to program the parameters for the individual external memories. Each EBI_CSR must be programmed with a different base address, even for unused chip selects.

Base Address: 0xFFE00000 (Code Label EBI_BASE)

Table 6. EBI Memory Map

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

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



EBI Chip Select Register

Register Name: EBI_CSR0 - EBI_CSR7

Access Type: Read/Write

Reset Value: See Table 6

Absolute Address: 0xFFE00000 - 0xFFE0001C

Offset: 0x00 - 0x1C

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

- DBW: Data Bus Width**

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

- NWS: Number of Wait States**

This field is valid only if WSE is set.

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

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

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

1 = Wait state generation is enabled.

- **PAGES: Page Size**

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

- **TDF: Data Float Output Time**

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

- **BAT: Byte Access Type**

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

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

0 = Chip select is disabled.

1 = Chip select is enabled.

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

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

EBI Remap Control Register

Register Name: EBI_RCR

Access Type: Write Only

Absolute Address: 0xFFE00020

Offset: 0x20

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

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

0 = No effect.

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

EBI Memory Control Register

Register Name: EBI_MCR

Access Type: Read/Write

Reset Value: 0

Absolute Address: 0xFFE00024

Offset: 0x24

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

- ALE: Address Line Enable**

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

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

- DRP: Data Read Protocol**

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

PS: Power-saving

The AT91X40 Series' Power-saving feature enables optimization of power consumption. The PS controls the CPU and Peripheral Clocks. One control register (PS_CR) enables the user to stop the ARM7TDMI Clock and enter Idle Mode. One set of registers with a set/clear mechanism enables and disables the peripheral clocks individually.

The ARM7TDMI clock is enabled after a reset and is automatically re-enabled by any enabled interrupt in the Idle Mode.

Peripheral Clocks

The clock of each peripheral integrated in the AT91X40 Series can be individually enabled and disabled by writing to the Peripheral Clock Enable (PS_PCER) and Peripheral Clock Disable Registers (PS_PCDR). The status of the peripheral clocks can be read in the Peripheral Clock Status Register (PS_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.

To avoid data corruption or erroneous behavior of the system, the system software only disables the clock after all programmed peripheral operations have finished.

The peripheral clocks are automatically enabled after a reset.

The bits that control the peripheral clocks are the same as those that control the Interrupt Sources in the AIC.

PS User Interface

Base Address: 0xFFFF4000 (Code Label PS_BASE)

Table 7. PS Memory Map

| Offset | Register | Name | Access | Reset State |
|--------|-----------------------------------|---------|------------|-------------|
| 0x00 | Control Register | PS_CR | Write Only | – |
| 0x04 | Peripheral Clock Enable Register | PS_PCER | Write Only | – |
| 0x08 | Peripheral Clock Disable Register | PS_PCDR | Write Only | – |
| 0x0C | Peripheral Clock Status Register | PS_PCSR | Read Only | 0x17C |

PS Control Register

Name: PS_CR
Access: Write Only
Offset: 0x00

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

- **CPU: CPU Clock Disable**

0 = No effect.

1 = Disables the CPU clock.

The CPU clock is re-enabled by any enabled interrupt or by hardware reset.

PS Peripheral Clock Enable Register

Name: PS_PCER

Access: Write Only

Offset: 0x04

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

- **US0: USART 0 Clock Enable**

0 = No effect.

1 = Enables the USART 0 clock.

- **US1: USART 1 Clock Enable**

0 = No effect.

1 = Enables the USART 1 clock.

- **TC0: Timer Counter 0 Clock Enable**

0 = No effect.

1 = Enables the Timer Counter 0 clock.

- **TC1: Timer Counter 1 Clock Enable**

0 = No effect.

1 = Enables the Timer Counter 1 clock.

- **TC2: Timer Counter 2 Clock Enable**

0 = No effect.

1 = Enables the Timer Counter 2 clock.

- **PIO: Parallel IO Clock Enable**

0 = No effect.

1 = Enables the Parallel IO clock.

PS Peripheral Clock Disable Register

Name: PS_PCDR

Access: Write Only

Offset: 0x08

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

- **US0: USART 0 Clock Disable**

0 = No effect.

1 = Disables the USART 0 clock.

- **US1: USART 1 Clock Disable**

0 = No effect.

1 = Disables the USART 1 clock.

- **TC0: Timer Counter 0 Clock Disable**

0 = No effect.

1 = Disables the Timer Counter 0 clock.

- **TC1: Timer Counter 1 Clock Disable**

0 = No effect.

1 = Disables the Timer Counter 1 clock.

- **TC2: Timer Counter 2 Clock Disable**

0 = No effect.

1 = Disables the Timer Counter 2 clock.

- **PIO: Parallel IO Clock Disable**

0 = No effect.

1 = Disables the Parallel IO clock.

PS Peripheral Clock Status Register

Name: PS_PCSR

Access: Read Only

Reset Value: 0x17C

Offset: 0x0C

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

- **US0: USART 0 Clock Status**

0 = USART 0 clock is disabled.

1 = USART 0 clock is enabled.

- **US1: USART 1 Clock Status**

0 = USART 1 clock is disabled.

1 = USART 1 clock is enabled.

- **TC0: Timer Counter 0 Clock Status**

0 = Timer Counter 0 clock is disabled.

1 = Timer Counter 0 clock is enabled.

- **TC1: Timer Counter 1 Clock Status**

0 = Timer Counter 1 clock is disabled.

1 = Timer Counter 1 clock is enabled.

- **TC2: Timer Counter 2 Clock Status**

0 = Timer Counter 2 clock is disabled.

1 = Timer Counter 2 clock is enabled.

- **PIO: Parallel IO Clock Status**

0 = Parallel IO clock is disabled.

1 = Parallel IO clock is enabled.

AIC: Advanced Interrupt Controller

The AT91X40 Series has an 8-level priority, individually maskable, vectored interrupt controller. This feature substantially reduces the software and real-time overhead in handling internal and external interrupts.

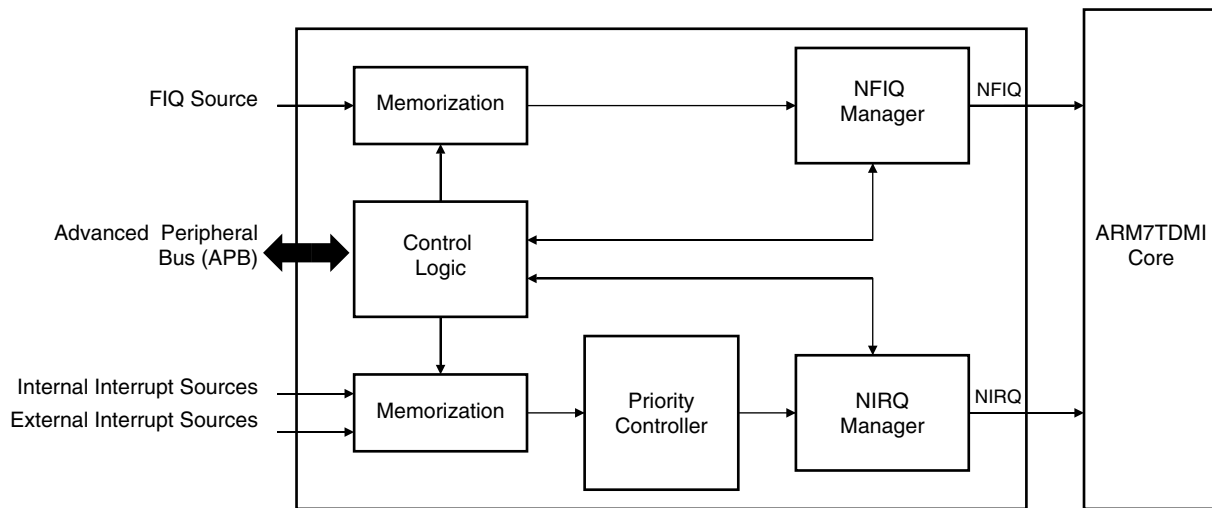
The interrupt controller is connected to the NFIQ (fast interrupt request) and the NIRQ (standard interrupt request) inputs of the ARM7TDMI processor. The processor's NFIQ line can only be asserted by the external fast interrupt request input: FIQ. The NIRQ line can be asserted by the interrupts generated by the on-chip peripherals and the external interrupt request lines: IRQ0 to IRQ2.

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

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

The interrupt sources are listed in Table 8 and the AIC programmable registers in Table 9.

Figure 33. Interrupt Controller Block Diagram



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

Table 8. AIC Interrupt Sources

| Interrupt Source ⁽¹⁾ | Interrupt Name | Interrupt Description |
|---------------------------------|----------------|-----------------------------------|
| 0 | FIQ | Fast Interrupt |
| 1 | SWIRQ | Software Interrupt |
| 2 | US0IRQ | USART Channel 0 interrupt |
| 3 | US1IRQ | USART Channel 1 interrupt |
| 4 | TC0IRQ | Timer Channel 0 interrupt |
| 5 | TC1IRQ | Timer Channel 1 interrupt |
| 6 | TC2IRQ | Timer Channel 2 interrupt |
| 7 | WDIRQ | Watchdog interrupt |
| 8 | PIOIRQ | Parallel I/O Controller interrupt |
| 9 | – | Reserved |
| 10 | – | Reserved |
| 11 | – | Reserved |
| 12 | – | Reserved |
| 13 | – | Reserved |
| 14 | – | Reserved |
| 15 | – | Reserved |
| 16 | IRQ0 | External interrupt 0 |
| 17 | IRQ1 | External interrupt 1 |
| 18 | IRQ2 | External interrupt 2 |
| 19 | – | Reserved |
| 20 | – | Reserved |
| 21 | – | Reserved |
| 22 | – | Reserved |
| 23 | – | Reserved |
| 24 | – | Reserved |
| 25 | – | Reserved |
| 26 | – | Reserved |
| 27 | – | Reserved |
| 28 | – | Reserved |
| 29 | – | Reserved |
| 30 | – | Reserved |
| 31 | – | Reserved |

Note: 1. Reserved interrupt sources are not available. Corresponding registers must not be used and read 0.

Hardware Interrupt Vectoring

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

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

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

Priority Controller

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

When the AIC receives more than one unmasked interrupt at a time, the interrupt with the highest priority is serviced first. If both interrupts have equal priority, the interrupt with the lowest interrupt source number (see table 8) is serviced first.

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

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

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

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

Interrupt Handling

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

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

Interrupt Masking

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

Interrupt Clearing and Setting

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

Fast Interrupt Request

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

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

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

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

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

Software Interrupt

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

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

Spurious Interrupt

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

This behavior is called a Spurious Interrupt.

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

A spurious interrupt may occur in the following cases:

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

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

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

Protect Mode

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

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

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

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

This is avoided by using Protect Mode.

The Protect Mode is enabled by setting the AIC bit in the SF Protect Mode Register (see “SF: Special Function Registers” on page 94).

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

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

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

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

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

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

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

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

AIC User Interface

- **Base Address:** 0xFFFFF000 (Code Label AIC_BASE)

Table 9. AIC Memory Map

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

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

AIC Source Mode Register

Register Name: AIC_SMR0 - AIC_SMR31

Access Type: Read/Write

Reset Value: 0

Offset: 0x000 - 0x07C

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

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

Program the priority level for all sources except source 0 (FIQ).

The priority level can be between 0 (lowest) and 7 (highest).

The priority level is not used for the FIQ, in the SMR0.

- **SRCTYPE: Interrupt Source Type**

Program the input to be positive or negative level sensitive or positive or negative edge triggered.

The active level or edge is not programmable for the internal sources.

| SRCTYPE | | External Sources | Code Label |
|---------|---|-------------------------|-------------------------------|
| | | | AIC_SRCTYPE |
| 0 | 0 | Low Level Sensitive | AIC_SRCTYPE_EXT_LOW_LEVEL |
| 0 | 1 | Negative Edge Triggered | AIC_SRCTYPE_EXT_NEGATIVE_EDGE |
| 1 | 0 | High Level Sensitive | AIC_SRCTYPE_EXT_HIGH_LEVEL |
| 1 | 1 | Positive Edge Triggered | AIC_SRCTYPE_EXT_POSITIVE_EDGE |

| SRCTYPE | | Internal Sources | Code Label |
|---------|---|------------------|-----------------------|
| | | | AIC_SRCTYPE |
| x | 0 | Level Sensitive | AIC_SRCTYPE_INT_LEVEL |
| x | 1 | Edge Triggered | AIC_SRCTYPE_INT_EDGE |

AIC Source Vector Register

Register Name: AIC_SVR0 - AIC_SVR31

Access Type: Read/Write

Reset Value: 0

Offset: 0x080 - 0x0FC

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

- **VECTOR: Interrupt Handler Address**

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

AIC Interrupt Vector Register

Register Name: AIC_IVR

Access Type: Read Only

Reset Value: 0

Offset: 0x100

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

- **IRQV: Interrupt Vector Register**

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

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

When there is no current interrupt, the IRQ Vector Register reads 0.

AIC FIQ Vector Register

Register Name: AIC_FVR

Access Type: Read Only

Reset Value: 0

Offset: 0x104

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

- **FIQV: FIQ Vector Register**

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

AIC Interrupt Status Register

Register Name: AIC_ISR

Access Type: Read Only

Reset Value: 0

Offset: 0x108

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

- **IRQID: Current IRQ Identifier (Code Label AIC_IRQID)**

The Interrupt Status Register returns the current interrupt source number.

AIC Interrupt Pending Register

Register Name: AIC_IPR

Access Type: Read Only

Reset Value: 0

Offset: 0x10C

| | | | | | | | |
|-------|--------|--------|--------|--------|--------|-------|--------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | IRQ2 | IRQ1 | IRQ0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | PIOIRQ |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WDIRQ | TC2IRQ | TC1IRQ | TC0IRQ | US1IRQ | US0IRQ | SWIRQ | FIQ |

- **Interrupt Pending**

0 = Corresponding interrupt is inactive.

1 = Corresponding interrupt is pending.

AIC Interrupt Mask Register

Register Name: AIC_IMR

Access Type: Read Only

Reset Value: 0

Offset: 0x110

| | | | | | | | |
|-------|--------|--------|--------|--------|--------|-------|--------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | IRQ2 | IRQ1 | IRQ0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | PIOIRQ |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WDIRQ | TC2IRQ | TC1IRQ | TC0IRQ | US1IRQ | US0IRQ | SWIRQ | FIQ |

- Interrupt Mask**

0 = Corresponding interrupt is disabled.

1 = Corresponding interrupt is enabled.

AIC Core Interrupt Status Register

Register Name: AIC_CISR

Access Type: Read Only

Reset Value: 0

Offset: 0x114

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

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

0 = NFIQ line inactive.

1 = NFIQ line active.

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

0 = NIRQ line inactive.

1 = NIRQ line active.

AIC Interrupt Enable Command Register

Register Name: AIC_IECR

Access Type: Write Only

Offset: 0x120

| | | | | | | | |
|-------|--------|--------|--------|--------|--------|-------|--------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | IRQ2 | IRQ1 | IRQ0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | PIOIRQ |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WDIRQ | TC2IRQ | TC1IRQ | TC0IRQ | US1IRQ | US0IRQ | SWIRQ | FIQ |

- Interrupt Enable**

0 = No effect.

1 = Enables corresponding interrupt.

AIC Interrupt Disable Command Register

Register Name: AIC_IDCR

Access Type: Write Only

Offset: 0x124

| | | | | | | | |
|-------|--------|--------|--------|--------|--------|-------|--------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | IRQ2 | IRQ1 | IRQ0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | PIOIRQ |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WDIRQ | TC2IRQ | TC1IRQ | TC0IRQ | US1IRQ | US0IRQ | SWIRQ | FIQ |

- Interrupt Disable**

0 = No effect.

1 = Disables corresponding interrupt.

AIC Interrupt Clear Command Register

Register Name: AIC_ICCR

Access Type: Write Only

Offset: 0x128

| | | | | | | | |
|-------|--------|--------|--------|--------|--------|-------|--------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | IRQ2 | IRQ1 | IRQ0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | PIOIRQ |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WDIRQ | TC2IRQ | TC1IRQ | TC0IRQ | US1IRQ | US0IRQ | SWIRQ | FIQ |

- Interrupt Clear**

0 = No effect.

1 = Clears corresponding interrupt.

AIC Interrupt Set Command Register

Register Name: AIC_ISCR

Access Type: Write Only

Offset: 0x12C

| | | | | | | | |
|-------|--------|--------|--------|--------|--------|-------|--------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | IRQ2 | IRQ1 | IRQ0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | PIOIRQ |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WDIRQ | TC2IRQ | TC1IRQ | TC0IRQ | US1IRQ | US0IRQ | SWIRQ | FIQ |

- Interrupt Set**

0 = No effect.

1 = Sets corresponding interrupt.

AIC End of Interrupt Command Register

Register Name: AIC_EOICR

Access Type: Write Only

Offset: 0x130

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

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

AIC Spurious Vector Register

Register Name: AIC_SPU

Access Type: Read/Write

Reset Value: 0

Offset: 0x134

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

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

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

Standard Interrupt Sequence

It is assumed that:

- The Advanced Interrupt Controller has been programmed, AIC_SVR are loaded with corresponding interrupt service routine addresses and interrupts are enabled.
- The Instruction at address 0x18 (IRQ exception vector address) is `ldr pc, [pc, # - &F20]`

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

1. The CPSR is stored in SPSR_irq, the current value of the Program Counter is loaded in the IRQ link register (r14_irq) and the Program Counter (r15) is loaded with 0x18. In the following cycle during fetch at address 0x1C, the ARM core adjusts r14_irq, decrementing it by 4.
2. The ARM core enters IRQ Mode, if it is not already.
3. When the instruction loaded at address 0x18 is executed, the Program Counter is loaded with the value read in AIC_IVR. Reading the AIC_IVR has the following effects:
 - Set the current interrupt to be the pending one with the highest priority. The current level is the priority level of the current interrupt.
 - De-assert the NIRQ line on the processor. (Even if vectoring is not used, AIC_IVR must be read in order to de-assert NIRQ)
 - Automatically clear the interrupt, if it has been programmed to be edge triggered
 - Push the current level on to the stack
 - Return the value written in the AIC_SVR corresponding to the current interrupt
4. The previous step has effect to branch to the corresponding interrupt service routine. This should start by saving the Link Register (r14_irq) and the SPSR (SPSR_irq). Note that the Link Register must be decremented by 4 when it is saved, if it is to be restored directly into the Program Counter at the end of the interrupt.
5. Further interrupts can then be unmasked by clearing the I bit in the CPSR, allowing re-assertion of the NIRQ to be taken into account by the core. This can occur if an interrupt with a higher priority than the current one occurs.
6. The Interrupt Handler can then proceed as required, saving the registers which will be used and restoring them at the end. During this phase, an interrupt of priority higher than the current level will restart the sequence from step 1. Note that if the interrupt is programmed to be level sensitive, the source of the interrupt must be cleared during this phase.
7. The I bit in the CPSR must be set in order to mask interrupts before exiting, to ensure that the interrupt is completed in an orderly manner.
8. The End Of Interrupt Command Register (AIC_EOICR) must be written in order to indicate to the AIC that the current interrupt is finished. This causes the current level to be popped from the stack, restoring the previous current level if one exists on the stack. If another interrupt is pending, with lower or equal priority than old current level but with higher priority than the new current level, the NIRQ line is re-asserted, but the interrupt sequence does not immediately start because the I bit is set in the core.
9. The SPSR (SPSR_irq) is restored. Finally, the saved value of the Link Register is restored directly into the PC. This has effect of returning from the interrupt to whatever was being executed before, and of loading the CPSR with the stored



SPSR, masking or unmasking the interrupts depending on the state saved in the SPSR (the previous state of the ARM core).

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

Fast Interrupt Sequence

It is assumed that:

- The Advanced Interrupt Controller has been programmed, AIC_SVR[0] is loaded with fast interrupt service routine address and the fast interrupt is enabled.
- The Instruction at address 0x1C(FIQ exception vector address) is:
- `ldr pc, [pc, # - &F20]`.
- Nested Fast Interrupts are not needed by the user.

When NFIQ is asserted, if the 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 4.
2. The ARM core enters FIQ Mode.
3. When the instruction loaded at address 0x1C is executed, the Program Counter is loaded with the value read in AIC_FVR. Reading the AIC_FVR has effect of automatically clearing the fast interrupt (source 0 connected to the FIQ line), if it has been programmed to be edge triggered. In this case only, it de-asserts the NFIQ line on the processor.
4. The previous step has effect to branch to the corresponding interrupt service routine. It is not necessary to save the Link Register(r14_fiq) and the SPSR(SPSR_fiq) if nested fast interrupts are not needed.
5. The Interrupt Handler can then proceed as required. It is not necessary to save registers r8 to r13 because FIQ Mode has its own dedicated registers and the user r8 to r13 are banked. The other registers, r0 to r7, must be saved before being used, and restored at the end (before the next step). Note that if the fast interrupt is programmed to be level sensitive, the source of the interrupt must be cleared during this phase in order to de-assert the NFIQ line.
6. Finally, the Link Register (r14_fiq) is restored into the PC after decrementing it by 4 (with instruction `sub pc, lr, #4` for example). This has effect of returning from the interrupt to whatever was being executed before, and of loading the CPSR with the SPSR, masking or unmasking the fast interrupt depending on the state saved in the SPSR.

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

PIO: Parallel I/O Controller

The AT91X40 Series has 32 programmable I/O lines. Six pins are dedicated as general purpose I/O pins (P16, P17, P18, P19, P23 and P24). Other I/O lines are multiplexed with an external signal of a peripheral to optimize the use of available package pins (see Table 10). The PIO controller also provides an internal interrupt signal to the Advanced Interrupt Controller.

Multiplexed I/O Lines

Some I/O lines are multiplexed with an I/O signal of a peripheral. After reset, the pin is generally controlled by the PIO Controller and is in Input Mode. Table 10 indicates which of these pins are not controlled by the PIO Controller after reset.

When a peripheral signal is not used in an application, the corresponding pin can be used as a parallel I/O. Each parallel I/O line is bi-directional, whether the peripheral defines the signal as input or output. Figure 34 shows the multiplexing of the peripheral signals with Parallel I/O signals.

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

If a pin is a general-purpose parallel I/O pin (not multiplexed with a peripheral), PIO_PER and PIO_PDR have no effect and PIO_PSR returns 1 for the bits corresponding to these pins.

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

Output Selection

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

I/O Levels

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

If a pin is controlled by the PIO Controller and is defined as an output (see “Output Selection” above), the level is programmed using the registers PIO_SODR (Set Output Data) and PIO_CODR (Clear Output Data). In this case, the programmed value can be read in PIO_ODSR (Output Data Status).

If a pin is controlled by the PIO Controller and is not defined as an output, the level is determined by the external circuit.

If a pin is not controlled by the PIO Controller, the state of the pin is defined by the peripheral (see peripheral datasheets).

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

Filters

Optional input glitch filtering is available on each pin of the AT91M40800, the AT91M40807 and the AT91R40807. Filtering is controlled by the registers PIO_IFER (Input Filter Enable) and PIO_IFDR (Input Filter Disable). The input glitch filtering can be selected whether the pin is used for its peripheral function or as a parallel I/O line. The register PIO_IFSR (Input Filter Status) indicates whether or not the filter is activated for each pin.

Interrupts

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

When PIO_ISR is read, the register is automatically cleared.

User Interface

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

Table 10. Multiplexed Parallel I/Os

| PIO Controller | | Peripheral | | | Reset State | Pin Number |
|---------------------------|-----------|------------|------------------------------|------------------|-------------|------------|
| Bit Number ⁽¹⁾ | Port Name | Port Name | Signal Description | Signal Direction | | |
| 0 | P0 | TCLK0 | Timer 0 Clock signal | Input | PIO Input | 49 |
| 1 | P1 | TIOA0 | Timer 0 Signal A | Bi-directional | PIO Input | 50 |
| 2 | P2 | TIOB0 | Timer 0 Signal B | Bi-directional | PIO Input | 51 |
| 3 | P3 | TCLK1 | Timer 1 Clock signal | Input | PIO Input | 54 |
| 4 | P4 | TIOA1 | Timer 1 Signal A | Bi-directional | PIO Input | 55 |
| 5 | P5 | TIOB1 | Timer 1 Signal B | Bi-directional | PIO Input | 56 |
| 6 | P6 | TCLK2 | Timer 2 Clock signal | Input | PIO Input | 57 |
| 7 | P7 | TIOA2 | Timer 2 Signal A | Bi-directional | PIO Input | 58 |
| 8 | P8 | TIOB2 | Timer 2 Signal B | Bi-directional | PIO Input | 59 |
| 9 | P9 | IRQ0 | External Interrupt 0 | Input | PIO Input | 60 |
| 10 | P10 | IRQ1 | External Interrupt 1 | Input | PIO Input | 63 |
| 11 | P11 | IRQ2 | External Interrupt 2 | Input | PIO Input | 64 |
| 12 | P12 | FIQ | Fast Interrupt | Input | PIO Input | 66 |
| 13 | P13 | SCK0 | USART 0 clock signal | Bi-directional | PIO Input | 67 |
| 14 | P14 | TXD0 | USART 0 transmit data signal | Output | PIO Input | 68 |
| 15 | P15 | RXD0 | USART 0 receive data signal | Input | PIO Input | 69 |
| 16 | P16 | – | – | – | PIO Input | 70 |
| 17 | P17 | – | – | – | PIO Input | 71 |
| 18 | P18 | – | – | – | PIO Input | 72 |
| 19 | P19 | – | – | – | PIO Input | 73 |
| 20 | P20 | SCK1 | USART 1 clock signal | Bi-directional | PIO Input | 74 |
| 21 | P21 | TXD1 | USART 1 transmit data signal | Output | PIO Input | 75 |
| 22 | P22 | RXD1 | USART 1 receive data signal | Input | PIO Input | 76 |
| 23 | P23 | – | – | – | PIO Input | 83 |
| 24 | P24 | – | – | – | PIO Input | 84 |
| 25 | P25 | MCKO | Master Clock Output | Output | MCKO | 85 |
| 26 | P26 | NCS2 | Chip Select 2 | Output | NCS2 | 99 |
| 27 | P27 | NCS3 | Chip Select 3 | Output | NCS3 | 100 |
| 28 | P28 | A20/CS7 | Address 20/Chip Select 7 | Output | A20 | 25 |
| 29 | P29 | A21/CS6 | Address 21/Chip Select 6 | Output | A21 | 26 |
| 30 | P30 | A22/CS5 | Address 22/Chip Select 5 | Output | A22 | 29 |
| 31 | P31 | A23/CS4 | Address 23/Chip Select 4 | Output | A23 | 30 |

Note: Bit Number refers to the data bit that corresponds to this signal in each of the User Interface registers.

PIO User Interface

PIO Base Address: 0xFFFF0000 (Code Label PIO_BASE)

Table 11. PIO Controller Memory Map

| Offset | Register | Name | Access | Reset State |
|--------|---|----------|------------|-----------------------------------|
| 0x00 | PIO Enable Register | PIO_PER | Write Only | – |
| 0x04 | PIO Disable Register | PIO_PDR | Write Only | – |
| 0x08 | PIO Status Register | PIO_PSR | Read Only | 0x01FFFFFF (see also Table 10) |
| 0x0C | Reserved | – | – | – |
| 0x10 | Output Enable Register | PIO_OER | Write Only | – |
| 0x14 | Output Disable Register | PIO_ODR | Write Only | – |
| 0x18 | Output Status Register | PIO_OSR | Read Only | 0 |
| 0x1C | Reserved | – | – | – |
| 0x20 | Input Filter Enable Register | PIO_IFER | Write Only | – |
| 0x24 | Input Filter Disable Register | PIO_IFDR | Write Only | – |
| 0x28 | Input Filter Status Register ⁽³⁾ | PIO_IFSR | Read Only | 0 |
| 0x2C | Reserved | – | – | – |
| 0x30 | Set Output Data Register | PIO_SODR | Write Only | – |
| 0x34 | Clear Output Data Register | PIO_CODR | Write Only | – |
| 0x38 | Output Data Status Register | PIO_ODSR | Read Only | 0 |
| 0x3C | Pin Data Status Register | PIO_PDSR | Read Only | (see Note 1) |
| 0x40 | Interrupt Enable Register | PIO_IER | Write Only | – |
| 0x44 | Interrupt Disable Register | PIO_IDR | Write Only | – |
| 0x48 | Interrupt Mask Register | PIO_IMR | Read Only | 0 |
| 0x4C | Interrupt Status Register | PIO_ISR | Read Only | (see Note 2) |

- Notes:
1. The reset value of this register depends on the level of the external pins at reset.
 2. This register is cleared at reset. However, the first read of the register can give a value not equal to zero if any changes have occurred on any pins between the reset and the read.
 3. This register exists in the AT91R40008 but its value has no meaning, since the filters are not implemented.

PIO Enable Register

Register Name: PIO_PER

Access Type: Write Only

Offset: 0x00

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

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

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

0 = No effect.

PIO Disable Register

Register Name: PIO_PDR

Access Type: Write Only

Offset: 0x04

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

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

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

0 = No effect.



PIO Status Register

Register Name: PIO_PSR

Access Type: Read Only

Reset Value: 0x01FFFFFF

Offset: 0x08

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

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

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

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

PIO Output Enable Register

Register Name: PIO_OER

Access Type: Write Only

Offset: 0x10

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

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

1 = Enables the PIO output on the corresponding pin.

0 = No effect.

PIO Output Disable Register

Register Name: PIO_ODR

Access Type: Write Only

Offset: 0x14

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

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

1 = Disables the PIO output on the corresponding pin.

0 = No effect.



PIO Output Status Register

Register Name: PIO_OSR

Access Type: Read Only

Reset Value: 0

Offset: 0x18

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

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

1 = The corresponding PIO is output on this line.

0 = The corresponding PIO is input on this line.

PIO Input Filter Enable Register

Register Name: PIO_IFER

Access Type: Write Only

Offset: 0x20

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

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

1 = Enables the glitch filter on the corresponding pin.

0 = No effect.

PIO Input Filter Disable Register

Register Name: PIO_IFDR

Access Type: Write Only

Offset: 0x24

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

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

1 = Disables the glitch filter on the corresponding pin.

0 = No effect.

PIO Input Filter Status Register

Register Name: PIO_IFSR

Access Type: Read Only

Reset Value: 0

Offset: 0x28

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

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

1 = Filter is selected on the corresponding input (peripheral and PIO).

0 = Filter is not selected on the corresponding input.

PIO Set Output Data Register

Register Name: PIO_SODR

Access Type: Write Only

Offset: 0x30

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

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

1 = PIO output data on the corresponding pin is set.

0 = No effect.

PIO Clear Output Data Register

Register Name: PIO_CODR

Access Type: Write Only

Offset: 0x34

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

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

1 = PIO output data on the corresponding pin is cleared.

0 = No effect.

PIO Output Data Status Register

Register Name: PIO_ODSR

Access Type: Read Only

Reset Value: 0

Offset: 0x38

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

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

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

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

PIO Pin Data Status Register

Register Name: PIO_PDSR

Access Type: Read Only

Reset Value: see Table 11

Offset: 0x3C

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

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

1 = The corresponding pin is at logic 1.

0 = The corresponding pin is at logic 0.

PIO Interrupt Enable Register

Register Name: PIO_IER

Access Type: Write Only

Offset: 0x40

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

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

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

0 = No effect.

PIO Interrupt Disable Register

Register Name: PIO_IDR

Access Type: Write Only

Offset: 0x44

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

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

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

0 = No effect.

PIO Interrupt Mask Register

Register Name: PIO_IMR

Access Type: Read Only

Reset Value: 0

Offset: 0x48

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

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

1 = Interrupt is enabled on the corresponding input pin.

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

PIO Interrupt Status Register

Register Name: PIO_ISR

Access Type: Read Only

Reset Value: 0

Offset: 0x4C

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

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

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

1 = At least one change has been detected on the corresponding pin since the register was last read.

0 = No change has been detected on the corresponding pin since the register was last read.

WD: Watchdog Timer

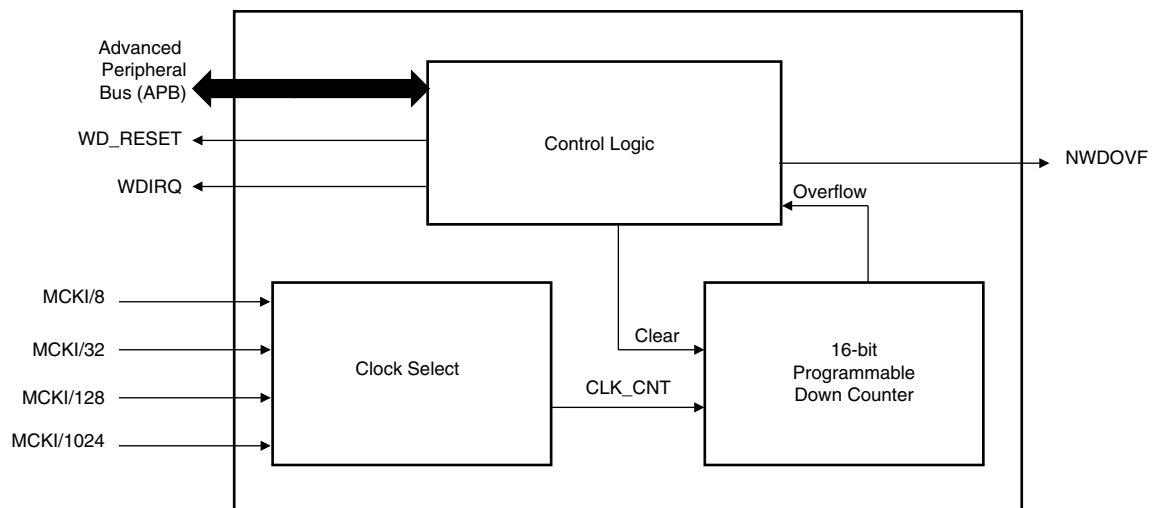
The AT91X40 Series has an internal watchdog timer which can be used to prevent system lock-up if the software becomes trapped in a deadlock. In normal operation the user reloads the watchdog at regular intervals before the timer overflow occurs. If an overflow does occur, the watchdog timer generates one or a combination of the following signals, depending on the parameters in WD_OMR (Overflow Mode Register):

- If RSTEN is set, an internal reset is generated (WD_RESET as shown in Figure 35).
- If IRQEN is set, a pulse is generated on the signal WDIRQ which is connected to the Advanced Interrupt Controller
- If EXTEN is set, a low level is driven on the NWDOVF signal for a duration of 8 MCK cycles.

The watchdog timer has a 16-bit down counter. Bits 12-15 of the value loaded when the watchdog is restarted are programmable using the HPVC parameter in WD_CMR (Clock Mode). Four clock sources are available to the watchdog counter: MCK/8, MCK/32, MCK/128 or MCK/1024. The selection is made using the WDCLKS parameter in WD_CMR. This provides a programmable time-out period of 1 ms to 2 sec. with a 33 MHz system clock.

All write accesses are protected by control access keys to help prevent corruption of the watchdog should an error condition occur. To update the contents of the mode and control registers it is necessary to write the correct bit pattern to the control access key bits at the same time as the control bits are written (the same write access).

Figure 35. Watchdog Timer Block Diagram



WD User Interface

WD Base Address: 0xFFFF8000 (Code Label WD_BASE)

Table 12. WD Memory Map

| Offset | Register | Name | Access | Reset State |
|--------|------------------------|--------|------------|-------------|
| 0x00 | Overflow Mode Register | WD_OMR | Read/Write | 0 |
| 0x04 | Clock Mode Register | WD_CMR | Read/Write | 0 |
| 0x08 | Control Register | WD_CR | Write Only | – |
| 0x0C | Status Register | WD_SR | Read Only | 0 |

WD Overflow Mode Register

Name: WD_OMR

Access: Read/Write

Reset Value: 0

Offset: 0x00

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

- **WDEN: Watch Dog Enable (Code Label WD_WDEN)**

0 = Watch Dog is disabled and does not generate any signals.

1 = Watch Dog is enabled and generates enabled signals.

- **RSTEN: Reset Enable (Code Label WD_RSTEN)**

0 = Generation of an internal reset by the Watch Dog is disabled.

1 = When overflow occurs, the Watch Dog generates an internal reset.

- **IRQEN: Interrupt Enable (Code Label WD_IRQEN)**

0 = Generation of an interrupt by the Watch Dog is disabled.

1 = When overflow occurs, the Watch Dog generates an interrupt.

- **EXTEN: External Signal Enable (Code Label WD_EXTEN)**

0 = Generation of a pulse on the pin NWDOVF by the Watch Dog is disabled.

1 = When an overflow occurs, a pulse on the pin NWDOVF is generated.

- **OKEY: Overflow Access Key (Code Label WD_OKEY)**

Used only when writing WD_OMR. OKEY is read as 0.

0x234 = Write access in WD_OMR is allowed.

Other value = Write access in WD_OMR is prohibited.

WD Clock Mode Register

Name: WD_CMR

Access: Read/Write

Reset Value: 0

Offset: 0x04

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

- WDCLKS: Clock Selection**

| WDCLKS | | Clock Selected | Code Label |
|--------|---|----------------|-------------------|
| | | | WD_WDCLKS |
| 0 | 0 | MCK/8 | WD_WDCLKS_MCK8 |
| 0 | 1 | MCK/32 | WD_WDCLKS_MCK32 |
| 1 | 0 | MCK/128 | WD_WDCLKS_MCK128 |
| 1 | 1 | MCK/1024 | WD_WDCLKS_MCK1024 |

- HPCV: High Preload Counter Value (Code Label WD_HPCV)**

Counter is preloaded when watchdog counter is restarted with bits 0 to 11 set (FFF) and bits 12 to 15 equaling HPCV.

- CKEY: Clock Access Key (Code Label WD_CKEY)**

Used only when writing WD_CMR. CKEY is read as 0.

0x06E: Write access in WD_CMR is allowed.

Other value: Write access in WD_CMR is prohibited.

WD Control Register

Name: WD_CR
Access: Write Only
Offset: 0x08

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

- **RSTKEY: Restart Key (Code Label WD_RSTKEY)**

0xC071 = Watch Dog counter is restarted.

Other value = No effect.

WD Status Register

Name: WD_SR
Access: Read Only
Reset Value: 0
Offset: 0x0C

| | | | | | | | |
|----|----|----|----|----|----|----|-------|
| 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 |
| – | – | – | – | – | – | – | WDOVF |

- **WDOVF: Watchdog Overflow (Code Label WD_WDOVF)**

0 = No watchdog overflow.

1 = A watchdog overflow has occurred since the last restart of the watchdog counter or since internal or external reset.

WD Enabling Sequence

To enable the Watchdog Timer the sequence is as follows:

1. Disable the Watchdog by clearing the bit WDEN:
Write 0x2340 to WD_OMR
This step is unnecessary if the WD is already disabled (reset state).
2. Initialize the WD Clock Mode Register:
Write 0x373C to WD_CMCR
(HPCV = 15 and WDCLKS = MCK/8)
3. Restart the timer:
Write 0xC071 to WD_CR
4. Enable the watchdog:
Write 0x2345 to WD_OMR (interrupt enabled)

SF: Special Function Registers

The AT91X40 Series provides registers which implement the following special functions.

- Chip identification
- RESET status
- Protect Mode (see “Protect Mode” on page 60)

Chip Identification

Table 13 provides the Chip ID values for the products described in this datasheet.

Table 13. Chip ID Values

| Product | Chip |
|------------|------------|
| AT91M40800 | 0x14080044 |
| AT91R40807 | 0x44080746 |
| AT91M40807 | 0x14080745 |
| AT91R40008 | 0x44000840 |

SF User Interface

Chip ID Base Address = 0xFFFF0000 (Code Label SF_BASE)

Table 14. SF Memory Map

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

Chip ID Register

Register Name: SF_CIDR

Access Type: Read Only

Reset Value: Hardwired

Offset: 0x00

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

- **VERSION: Version of the chip (Code Label SF_VERSION)**

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

- **NVPSIZ: Non Volatile Program Memory Size**

| NVPSIZ | | | | Size | Code Label |
|--------|---|---|---|------------|----------------|
| | | | | | SF_NVPSIZ |
| 0 | 0 | 0 | 0 | None | SF_NVPSIZ_NONE |
| 0 | 0 | 1 | 1 | 32K bytes | SF_NVPSIZ_32K |
| 0 | 1 | 0 | 1 | 64K bytes | SF_NVPSIZ_64K |
| 0 | 1 | 1 | 1 | 128K bytes | SF_NVPSIZ_128K |
| 1 | 0 | 0 | 1 | 256K bytes | SF_NVPSIZ_256K |
| Others | | | | Reserved | — |

- **NVDSIZ: Non Volatile Data Memory Size**

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

- **VDSIZ: Volatile Data Memory Size**

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

- **ARCH: Chip Architecture (Code Label SF_ARCH)**

Code of Architecture: Two BCD digits.

| 0100 0000 | | AT91x40yyy | | Code Label |
|-----------|--|------------|--|-----------------|
| | | | | SF_ARCH_AT91x40 |
| | | | | |

- **NVPTYP: Non Volatile Program Memory Type**

| NVPTYP | | | Type | Code Label |
|--------|---|---|------------|-------------|
| | | | | SF_NVPTYP |
| 0 | 0 | 0 | Reserved | – |
| 0 | 0 | 1 | “F” Series | SF_NVPTYP_M |
| 1 | x | x | Reserved | – |
| 1 | 0 | 0 | “R” Series | SF_NVPTYP_R |

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

0 = Chip ID has a single register definition without extensions

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

Chip ID Extension Register

Register Name: SF_EXID

Access Type: Read Only

Reset Value: Hardwired

Offset: 0x04

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

Reset Status Register

Register Name: SF_RSR

Access Type: Read Only

Reset Value: See Below

Offset: 0x08

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

- RESET: Reset Status Information**

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

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

SF Memory Mode Register

This register only applies to the AT91R40807.

Register Name: SF_MMR

Access Type: Read/Write

Reset Value: 0

Offset: 0x0C

| | | | | | | | |
|----|----|----|----|----|----|----|-------|
| 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 |
| – | – | – | – | – | – | – | RAMWU |

- **RAMWU: Internal Extended RAM Write Detection (Code Label SF_RAMWU)**

0 = Writing in RAM generates an Abort.

1 = Writing in RAM is allowed.

SF Protect Mode Register

Register Name: SF_PMR

Access Type: Read/Write

Reset Value: 0

Offset: 0x18

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

- **PMRKEY: Protect Mode Register Key (Code Label SF_PMRKEY)**

Used only when writing SF_PMR. PMRKEY is reads 0.

0x27A8: Write access in SF_PMR is allowed.

Other value: Write access in SF_PMR is prohibited.

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

0 = The Advanced Interrupt Controller runs in Normal Mode.

1 = The Advanced Interrupt Controller runs in Protect Mode.

See “Protect Mode” on page 60.

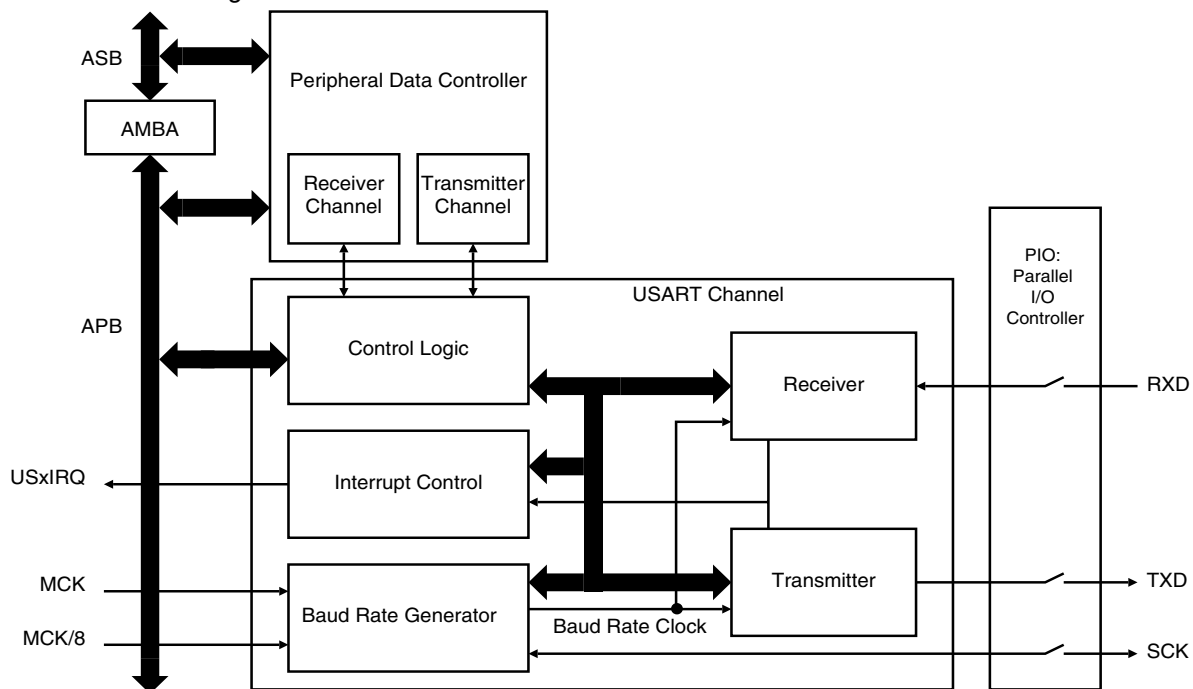
USART: Universal Synchronous/Asynchronous Receiver/Transmitter

The AT91X40 Series provides two identical, full-duplex, universal synchronous/asynchronous receiver/transmitters that interface to the APB and are connected to the Peripheral Data Controller.

The main features are:

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

Figure 36. USART Block Diagram



Pin Description

Each USART channel has the following external signals:

| Name | Description |
|------|---|
| SCK | USART Serial clock can be configured as input or output: SCK is configured as input if an External clock is selected (USCLKS[1] = 1) SCK is driven as output if the External Clock is disabled (USCLKS[1] = 0) and Clock output is enabled (CLKO = 1) |
| TXD | Transmit Serial Data is an output |
| RXD | Receive Serial Data is an input |

- Notes:
1. After a hardware reset, the USART pins are not enabled by default (see "PIO: Parallel I/O Controller" on page 74). The user must configure the PIO Controller before enabling the transmitter or receiver.
 2. If the user selects one of the internal clocks, SCK can be configured as a PIO.

Baud Rate Generator

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

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

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

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

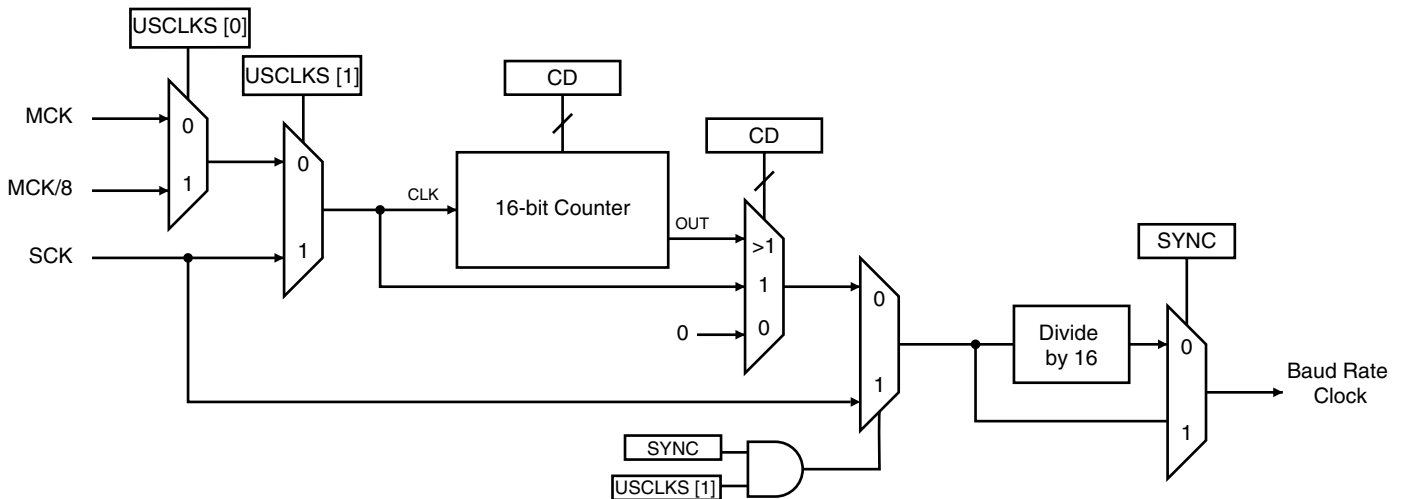
$$\text{Baud Rate} = \frac{\text{Selected Clock}}{16 \times \text{CD}}$$

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

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

In Synchronous Mode with external clock selected (USCLKS[1] = 1), the clock is provided directly by the signal on the SCK pin. No division is active. The value written in US_BRGR has no effect.

Figure 37. Baud Rate Generator



Receiver

Asynchronous Receiver

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

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

Figure 38. Asynchronous Mode: Start Bit Detection

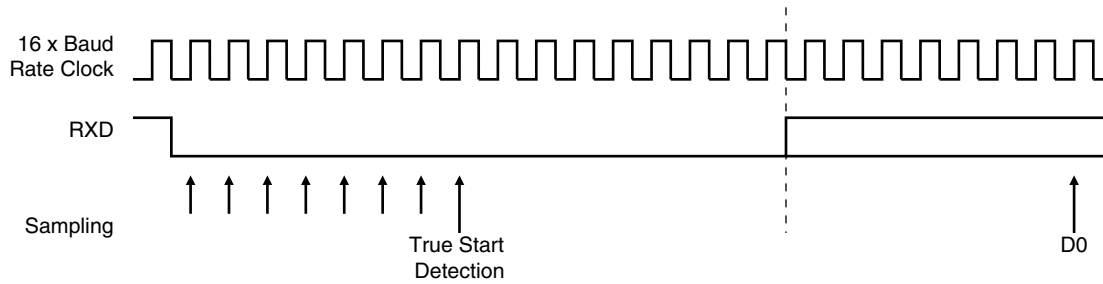
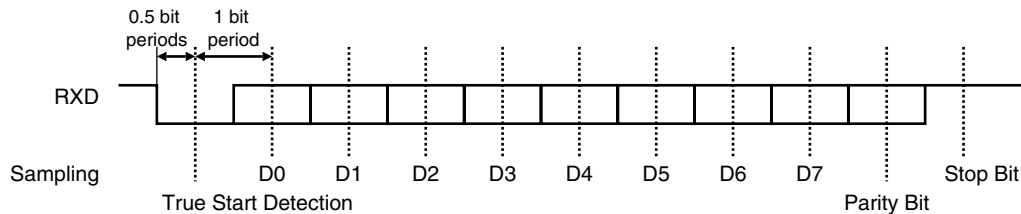


Figure 39. Asynchronous Mode: Character Reception

Example: 8-bit, parity enabled 1 stop

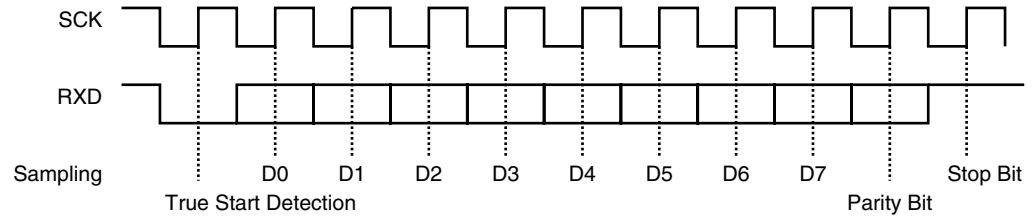


Synchronous Receiver

When configured for synchronous operation (SYNC = 1), the receiver samples the RXD signal on each rising edge of the Baud Rate clock. If a low level is detected, it is considered as a start. Data bits, parity bit and stop bit are sampled and the receiver waits for the next start bit. See example in Figure 40.

Figure 40. Synchronous Mode: Character Reception

Example: 8-bit, parity enabled 1 stop



Receiver Ready

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

Parity Error

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

Framing Error

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

Time-out

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

Calculation of time-out duration:

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

Transmitter

The transmitter has the same behavior in both synchronous and asynchronous operating modes. Start bit, data bits, parity bit and stop bits are serially shifted, lowest significant bit first, on the falling edge of the serial clock. See example in Figure 41.

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

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

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

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

Time-guard

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

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

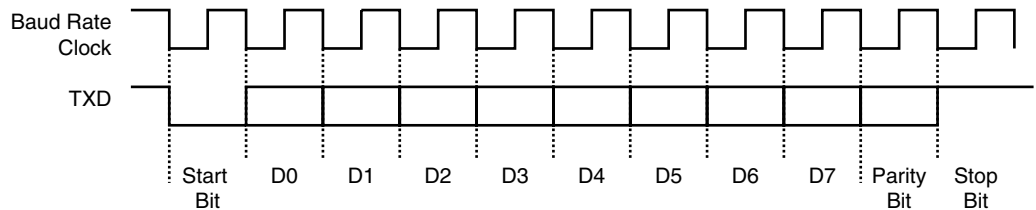
Multi-drop Mode

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

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

Figure 41. Synchronous and Asynchronous Modes: Character Transmission

Example: 8-bit, parity enabled 1 stop



Break

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

Transmit Break

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

To cancel a break condition on the TXD line, the STPBRK command in US_CR must be set. The USART completes a minimum break duration of one character length. The TXD line then returns to high level (idle state) for at least 12 bit periods to ensure that the end of break is correctly detected. Then the transmitter resumes normal operation.

The BREAK is managed like a character:

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

In order to avoid unpredictable states:

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

The standard break transmission sequence is:

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

The next byte can then be sent:

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

Each of these steps can be scheduled by using the interrupt if the bit TXRDY in US_IMR is set. For character transmission, the USART channel must be enabled before sending a break.

Receive Break

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

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

Peripheral Data Controller

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

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

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

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

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

Interrupt Generation

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

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

Channel Modes

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

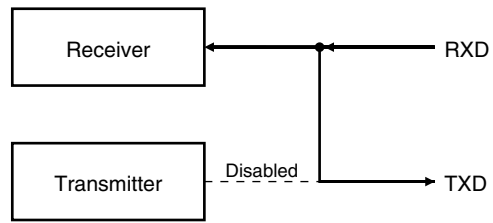
Automatic Echo Mode allows bit by bit re-transmission. When a bit is received on the RXD line, it is sent to the TXD line. Programming the transmitter has no effect.

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

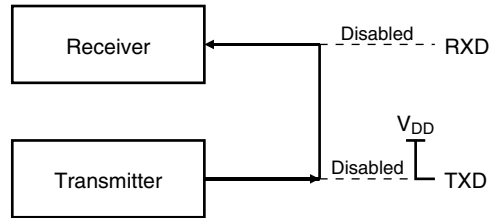
Remote Loopback Mode directly connects the RXD pin to the TXD pin. The Transmitter and the Receiver are disabled and have no effect. This mode allows bit by bit re-transmission.

Figure 42. Channel Modes

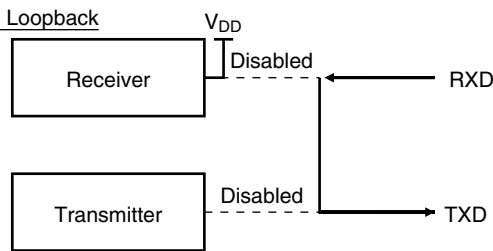
Automatic Echo



Local Loopback



Remote Loopback



USART User Interface

Base Address USART0: 0xFFFFD0000 (Code Label USART0_BASE)

Base Address USART1: 0xFFFFCC000 (Code Label USART1_BASE)

Table 15. USART Memory Map

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

USART Control Register

Name: US_CR

Access Type: Write Only

Offset: 0x00

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

- **RSTRX: Reset Receiver (Code Label US_RSTRX)**

0 = No effect.

1 = The receiver logic is reset.

- **RSTTX: Reset Transmitter (Code Label US_RSTTX)**

0 = No effect.

1 = The transmitter logic is reset.

- **RXEN: Receiver Enable (Code Label US_RXEN)**

0 = No effect.

1 = The receiver is enabled if RXDIS is 0.

- **RXDIS: Receiver Disable (Code Label US_RXDIS)**

0 = No effect.

1 = The receiver is disabled.

- **TXEN: Transmitter Enable (Code Label US_TXEN)**

0 = No effect.

1 = The transmitter is enabled if TXDIS is 0.

- **TXDIS: Transmitter Disable (Code Label US_TXDIS)**

0 = No effect.

1 = The transmitter is disabled.

- **RSTSTA: Reset Status Bits (Code Label US_RSTSTA)**

0 = No effect.

1 = Resets the status bits PARE, FRAME, OVRE and RXBRK in the US_CSR.

- **STTBRK: Start Break (Code Label US_STTBRK)**

0 = No effect.

1 = If break is not being transmitted, start transmission of a break after the characters present in US_THR and the Transmit Shift Register have been transmitted.

- **STPBRK: Stop Break (Code Label US_STPBRK)**

0 = No effect.

1 = If a break is being transmitted, stop transmission of the break after a minimum of one character length and transmit a high level during 12 bit periods.

- **STTTO: Start Time-out (Code Label `US_STTTO`)**

0 = No effect.

1 = Start waiting for a character before clocking the time-out counter.

- **SEND_A: Send Address (Code Label `US_SEND_A`)**

0 = No effect.

1 = In Multi-drop Mode only, the next character written to the `US_THR` is sent with the address bit set.

USART Mode Register

Name: US_MR

Access Type: Read/Write

Reset Value: 0

Offset: 0x04

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

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

| USCLKS | | Selected Clock | Code Label |
|--------|---|----------------|--------------|
| | | | US_CLKS |
| 0 | 0 | MCK | US_CLKS_MCK |
| 0 | 1 | MCK/8 | US_CLKS_MCK8 |
| 1 | X | External (SCK) | US_CLKS_SCK |

- CHRL: Character Length**

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

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

- SYNC: Synchronous Mode Select (Code Label `US_SYNC`)**

0 = USART operates in Asynchronous Mode.

1 = USART operates in Synchronous Mode.

- PAR: Parity Type**

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

- NBSTOP: Number of Stop Bits**

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

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

- CHMODE: Channel Mode**

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

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

0 = CHRL defines character length.

1 = 9-bit character length.

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

0 = The USART does not drive the SCK pin.

1 = The USART drives the SCK pin if USCLKS[1] is 0.

USART Interrupt Enable Register

Name: US_IER

Access Type: Write Only

Offset: 0x08

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

- **RXRDY: Enable RXRDY Interrupt (Code Label US_RXRDY)**

0 = No effect.

1 = Enables RXRDY Interrupt.

- **TXRDY: Enable TXRDY Interrupt (Code Label US_TXRDY)**

0 = No effect.

1 = Enables TXRDY Interrupt.

- **RXBRK: Enable Receiver Break Interrupt (Code Label US_RXBRK)**

0 = No effect.

1 = Enables Receiver Break Interrupt.

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

0 = No effect.

1 = Enables End of Receive Transfer Interrupt.

- **ENDTX: Enable End of Transmit Interrupt (Code Label US_ENDTX)**

0 = No effect.

1 = Enables End of Transmit Interrupt.

- **OVRE: Enable Overrun Error Interrupt (Code Label US_OVRE)**

0 = No effect.

1 = Enables Overrun Error Interrupt.

- **FRAME: Enable Framing Error Interrupt (Code Label US_FRAME)**

0 = No effect.

1 = Enables Framing Error Interrupt.

- **PARE: Enable Parity Error Interrupt (Code Label US_PARE)**

0 = No effect.

1 = Enables Parity Error Interrupt.

- **TIMEOUT: Enable Time-out Interrupt (Code Label US_TIMEOUT)**

0 = No effect.

1 = Enables Reception Time-out Interrupt.

- **TXEMPTY: Enable TXEMPTY Interrupt (Code Label US_TXEMPTY)**

0 = No effect.

1 = Enables TXEMPTY Interrupt.

USART Interrupt Disable Register

Name: US_IDR

Access Type: Write Only

Offset: 0x0C

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

- **RXRDY: Disable RXRDY Interrupt (Code Label US_RXRDY)**

0 = No effect.

1 = Disables RXRDY Interrupt.

- **TXRDY: Disable TXRDY Interrupt (Code Label US_TXRDY)**

0 = No effect.

1 = Disables TXRDY Interrupt.

- **RXBRK: Disable Receiver Break Interrupt (Code Label US_RXBRK)**

0 = No effect.

1 = Disables Receiver Break Interrupt.

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

0 = No effect.

1 = Disables End of Receive Transfer Interrupt.

- **ENDTX: Disable End of Transmit Interrupt (Code Label US_ENDTX)**

0 = No effect.

1 = Disables End of Transmit Interrupt.

- **OVRE: Disable Overrun Error Interrupt (Code Label US_OVRE)**

0 = No effect.

1 = Disables Overrun Error Interrupt.

- **FRAME: Disable Framing Error Interrupt (Code Label US_FRAME)**

0 = No effect.

1 = Disables Framing Error Interrupt.

- **PARE: Disable Parity Error Interrupt (Code Label US_PARE)**

0 = No effect.

1 = Disables Parity Error Interrupt.

- **TIMEOUT: Disable Time-out Interrupt (Code Label US_TIMEOUT)**

0 = No effect.

1 = Disables Receiver Time-out Interrupt.

- **TXEMPTY: Disable TXEMPTY Interrupt (Code Label US_TXEMPTY)**

0 = No effect.

1 = Disables TXEMPTY Interrupt.

USART Interrupt Mask Register

Name: US_IMR

Access Type: Read Only

Reset Value: 0

Offset: 0x10

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

- **RXRDY: Mask RXRDY Interrupt (Code Label US_RXRDY)**

0 = RXRDY Interrupt is Disabled

1 = RXRDY Interrupt is Enabled

- **TXRDY: Mask TXRDY Interrupt (Code Label US_TXRDY)**

0 = TXRDY Interrupt is Disabled

1 = TXRDY Interrupt is Enabled

- **RXBRK: Mask Receiver Break Interrupt (Code Label US_RXBRK)**

0 = Receiver Break Interrupt is Disabled

1 = Receiver Break Interrupt is Enabled

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

0 = End of Receive Transfer Interrupt is Disabled

1 = End of Receive Transfer Interrupt is Enabled

- **ENDTX: Mask End of Transmit Interrupt (Code Label US_ENDTX)**

0 = End of Transmit Interrupt is Disabled

1 = End of Transmit Interrupt is Enabled

- **OVRE: Mask Overrun Error Interrupt (Code Label US_OVRE)**

0 = Overrun Error Interrupt is Disabled

1 = Overrun Error Interrupt is Enabled

- **FRAME: Mask Framing Error Interrupt (Code Label US_FRAME)**

0 = Framing Error Interrupt is Disabled

1 = Framing Error Interrupt is Enabled

- **PARE: Mask Parity Error Interrupt (Code Label US_PARE)**

0 = Parity Error Interrupt is Disabled

1 = Parity Error Interrupt is Enabled

- **TIMEOUT: Mask Time-out Interrupt (Code Label US_TIMEOUT)**

0 = Receive Time-out Interrupt is Disabled

1 = Receive Time-out Interrupt is Enabled

- **TXEMPTY: Mask TXEMPTY Interrupt (Code Label US_TXEMPTY)**

0 = TXEMPTY Interrupt is Disabled.

1 = TXEMPTY Interrupt is Enabled.

USART Channel Status Register

Name: US_CSR

Access Type: Read Only

Reset Value: 0x18

Offset: 0x14

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

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

0 = No complete character has been received since the last read of the US_RHR or the receiver is disabled.

1 = At least one complete character has been received and the US_RHR has not yet been read.

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

0 = US_THR contains a character waiting to be transferred to the Transmit Shift Register, or an STTBRK command has been requested.

1 = US_THR is empty and there is no Break request pending TSR availability.

Equal to zero when the USART is disabled or at reset. Transmitter Enable command (in US_CR) sets this bit to one.

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

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

1 = Break Received or End of Break has been detected since the last “Reset Status Bits” command in the Control Register.

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

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

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

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

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

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

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

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

1 = At least one byte has been transferred from the Receive Shift Register to the US_RHR when RxRDY was asserted since the last “Reset Status Bits” command.

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

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

1 = At least one stop bit has been detected low since the last “Reset Status Bits” command.

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

1 = At least one parity bit has been detected false (or a parity bit high in Multi-drop Mode) since the last “Reset Status Bits” command.

0 = No parity bit has been detected false (or a parity bit high in Multi-drop Mode) since the last “Reset Status Bits” command.

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

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

1 = There has been a time-out since the last “Start Time-out” command.

- **TXEMPTY: Transmitter Empty (Code Label `US_TXEMPTY`)**

0 = There are characters in either `US_THR` or the Transmit Shift Register or a Break is being transmitted.

1 = There are no characters in `US_THR` and the Transmit Shift Register and Break is not active.

Equal to zero when the USART is disabled or at reset. Transmitter Enable command (in `US_CR`) sets this bit to one.

USART Receiver Holding Register

Name: US_RHR
Access Type: Read Only
Reset Value: 0
Offset: 0x18

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

- **RXCHR: Received Character**

Last character received if RXRDY is set. When number of data bits is less than 8 bits, the bits are right-aligned. All non-significant bits read zero.

USART Transmitter Holding Register

Name: US_THR
Access Type: Write Only
Offset: 0x1C

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

- **TXCHR: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set. When number of data bits is less than 8 bits, the bits are right-aligned.

USART Baud Rate Generator Register

Name: US_BRGR

Access Type: Read/Write

Reset Value: 0

Offset: 0x20

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

• CD: Clock Divisor

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

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

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

USART Receiver Time-out Register

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

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

• **TO: Time-out Value**

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

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

Time-out duration = TO x 4 x Bit period

USART Transmitter Time-guard Register

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

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

• **TG: Time-guard Value**

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

Time-guard duration = TG x Bit period

USART Receive Pointer Register

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

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

- **RXPTR: Receive Pointer**

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

USART Receive Counter Register

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

| | | | | | | | |
|-------|----|----|------|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 4920 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RXCTR | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RXCTR | | | | | | | |

- **RXCTR: Receive Counter**

RXCTR must be loaded with the size of the receive buffer.

0: Stop Peripheral Data Transfer dedicated to the receiver.

1 - 65535: Start Peripheral Data transfer if RXRDY is active.

USART Transmit Pointer Register

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

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

- **TXPTR: Transmit Pointer**

TXPTR must be loaded with the address of the transmit buffer.

USART Transmit Counter Register

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

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

- **TXCTR: Transmit Counter**

TXCTR must be loaded with the size of the transmit buffer.

0: Stop Peripheral Data Transfer dedicated to the transmitter.

1 - 65535: Start Peripheral Data transfer if TXRDY is active.

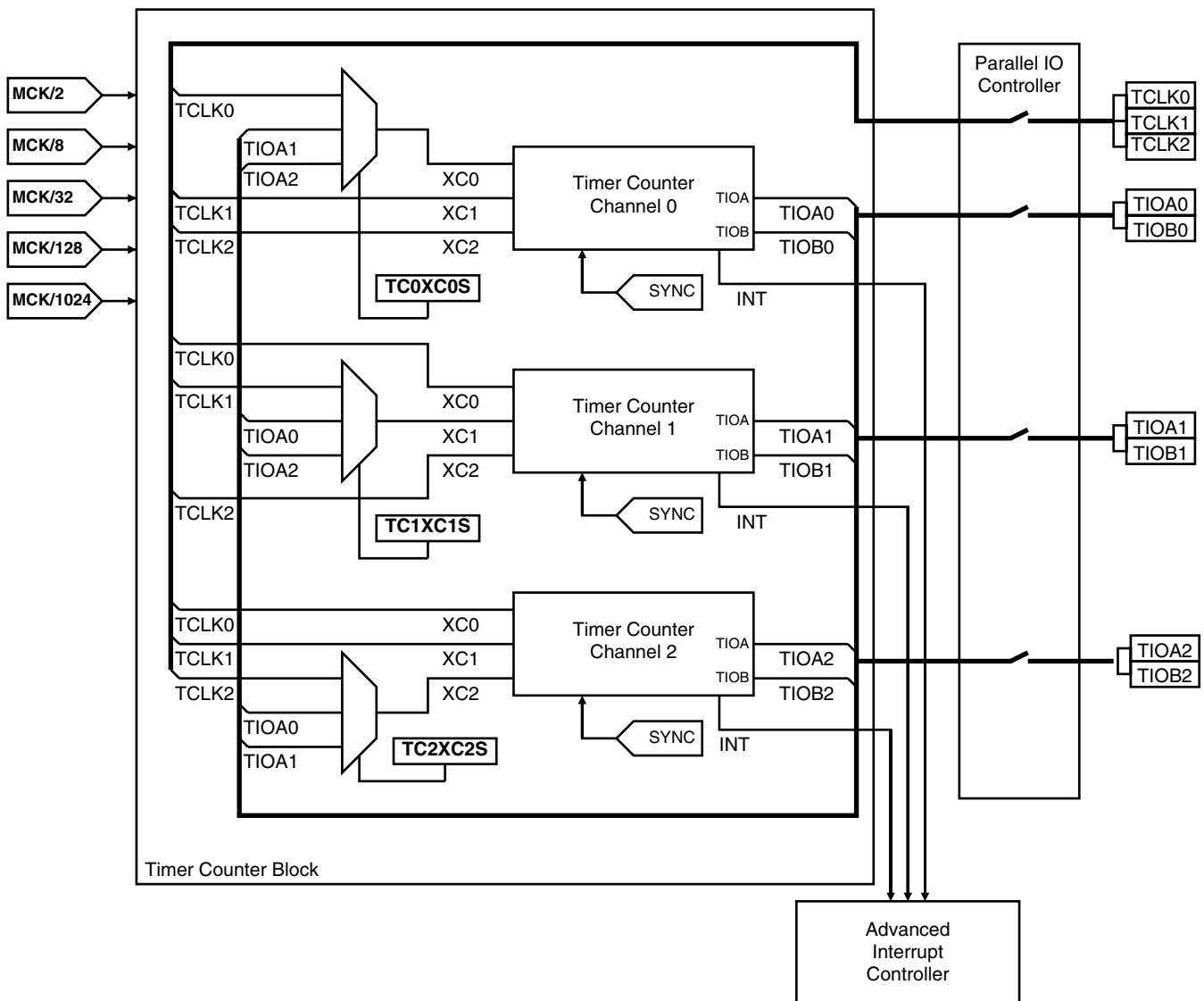
TC: Timer Counter

The AT91X40 Series features a Timer Counter block which 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 Timer Counter channel has 3 external clock inputs, 5 internal clock inputs, and 2 multi-purpose input/output signals which can be configured by the user. Each channel drives an internal interrupt signal which can be programmed to generate processor interrupts via the AIC (Advanced Interrupt Controller).

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

Figure 43. TC Block Diagram



Signal Name Description

| Channel Signal | Description |
|---------------------|--|
| XC0, XC1, XC2 | External Clock Inputs |
| TIOA | Capture Mode: General Purpose Input Waveform Mode: General Purpose Output |
| TIOB | Capture Mode: General Purpose Input Waveform Mode: General Purpose Input/Output |
| INT | Interrupt Signal Output |
| SYNC | Synchronization Input Signal |
| Block Signals | Description |
| 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 |

Note: After a hardware reset, the Timer Counter block pins are controlled by the PIO Controller. They must be configured to be controlled by the peripheral before being used.

Timer Counter Description

The three Timer Counter channels are independent and identical in operation. The registers for channel programming are listed in Table 17.

Counter

Each Timer Counter 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 bit COVFS in TC_SR (Status Register) is set.

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

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

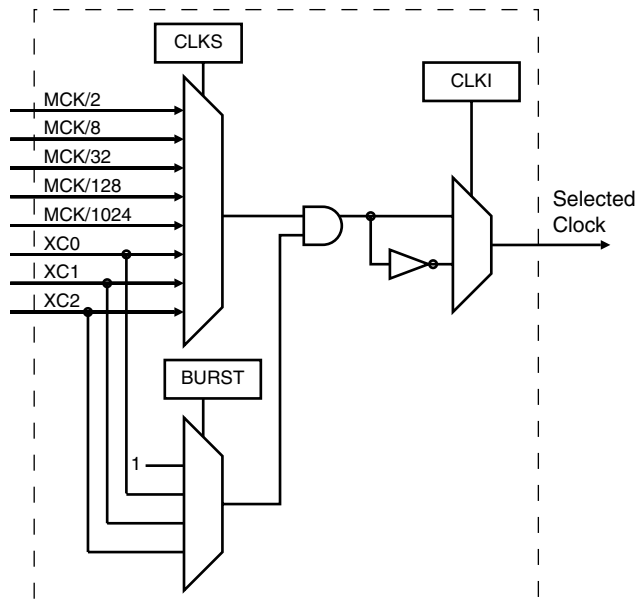
- Internal clock signals: MCK/2, MCK/8, MCK/32, MCK/128, MCK/1024
- External clock signals: XC0, XC1 or XC2

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

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

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

Figure 44. Clock Selection

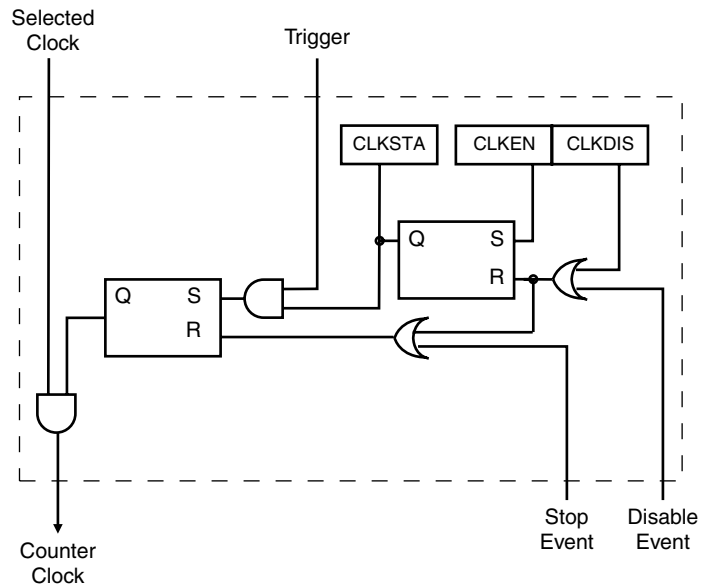


Clock Control

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

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

Figure 45. Clock Control



Timer Counter Operating Modes

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

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

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

Trigger

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

The following triggers are common to both modes:

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

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

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

Whatever the trigger used, it will be taken into account at the following active edge of the selected clock. This means that the counter value may not read 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 inputs.

Figure 46 shows the configuration of the TC Channel when programmed in Capture Mode.

Capture Registers A and B (RA and RB)

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

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

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

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

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

Trigger Conditions

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

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

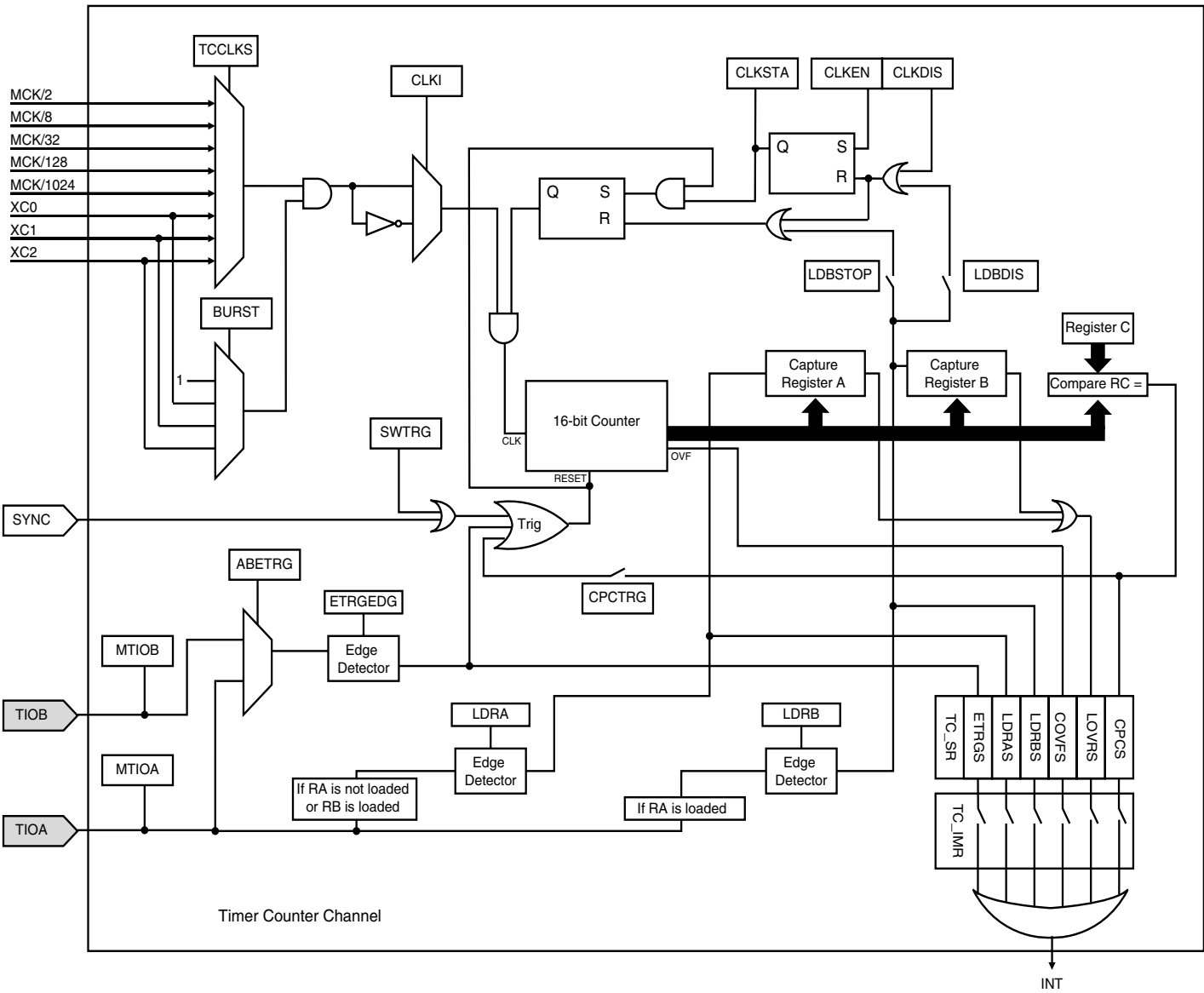
Status Register

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

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

Note: All the status bits are set when the corresponding event occurs and they are automatically cleared when the Status Register is read.

Figure 46. Capture Mode



Waveform Operating Mode

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

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

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

Figure 47 shows the configuration of the TC Channel when programmed in Waveform Operating Mode.

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

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

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

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

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

External Event/Trigger Conditions

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

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

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

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

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

Output Controller

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

The following events control TIOA and TIOB: software trigger, external event and RC compare. RA compare controls TIOA and RB compare controls TIOB. Each of these events can be programmed to set, clear or toggle the output as defined in the corresponding parameter in TC_CMCR.

The tables below show which parameter in TC_CMCR is used to define the effect of each event.

| Parameter | TIOA Event |
|-----------|------------------|
| ASWTRG | Software Trigger |
| AEEVT | External Event |
| ACPC | RC Compare |
| ACPA | RA Compare |

| Parameter | TIOB Event |
|-----------|------------------|
| BSWTRG | Software Trigger |
| BEEVT | External Event |
| BCPC | RC Compare |
| BCPB | RB Compare |

If two or more events occur at the same time, the priority level is defined as follows:

1. Software Trigger
2. External Event
3. RC Compare
4. RA or RB Compare

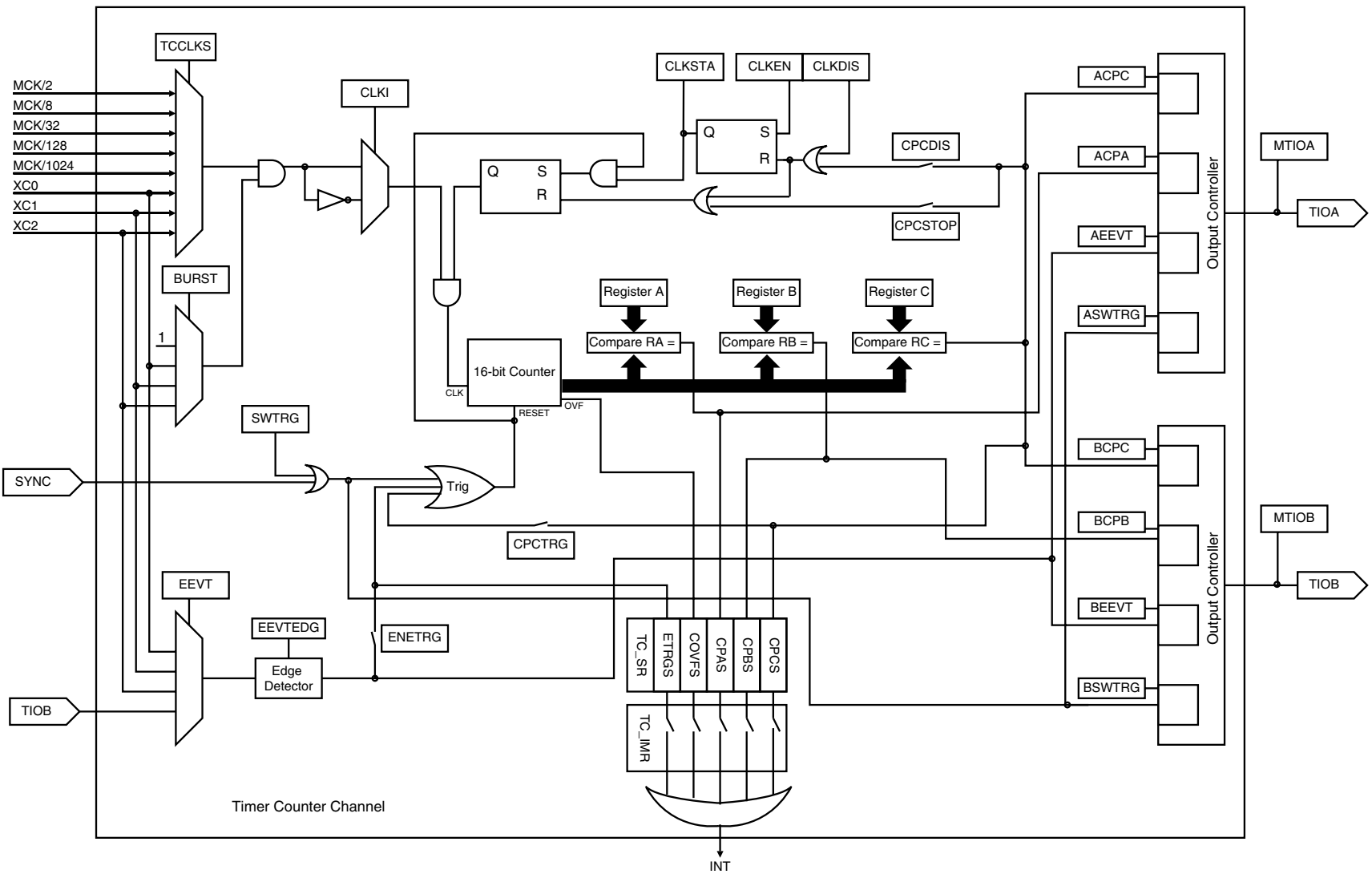
Status

The following bits in the status register are significant in Waveform Mode:

- CPAS: RA Compare Status
There has been a RA Compare match at least once since the last read of the status
- CPBS: RB Compare Status
There has been a RB Compare match at least once since the last read of the status
- CPCS: RC Compare Status
There has been a RC Compare match at least once since the last read of the status
- COVFS: Counter Overflow
Counter has attempted to count past \$FFFF since the last read of the status
- ETRGS: External Trigger
External trigger has been detected since the last read of the status

Note: All the status bits are set when the corresponding event occurs and they are automatically cleared when the Status Register is read.

Figure 47. Waveform Mode



TC User Interface

TC Base Address: 0xFFFFE0000 (Code Label TC_BASE)

Table 16. TC Global Memory Map

| Offset | Channel/Register | Name | Access | Reset State |
|--------|---------------------------|--------|--------------|-------------|
| 0x00 | TC Channel 0 | | See Table 17 | |
| 0x40 | TC Channel 1 | | See Table 17 | |
| 0x80 | TC Channel 2 | | See Table 17 | |
| 0xC0 | TC Block Control Register | TC_BCR | Write Only | – |
| 0xC4 | TC Block Mode Register | TC_BMR | Read/Write | 0 |

TC_BCR (Block Control Register) and TC_BMR (Block Mode Register) control the TC block. TC Channels are controlled by the registers listed in Table 17. The offset of each of the Channel registers in Table 17 is in relation to the offset of the corresponding channel as mentioned in Table 16.

Table 17. TC Channel Memory Map

| Offset | Register | Name | Access | Reset State |
|--------|----------------------------|--------|---------------------------|-------------|
| 0x00 | Channel Control Register | TC_CCR | Write Only | – |
| 0x04 | Channel Mode Register | TC_CMR | Read/Write | 0 |
| 0x08 | Reserved | | | – |
| 0x0C | Reserved | | | – |
| 0x10 | Counter Value | TC_CV | Read/Write | 0 |
| 0x14 | Register A | TC_RA | Read/Write ⁽¹⁾ | 0 |
| 0x18 | Register B | TC_RB | Read/Write ⁽¹⁾ | 0 |
| 0x1C | Register C | TC_RC | Read/Write | 0 |
| 0x20 | Status Register | TC_SR | Read Only | 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 |

Note: Read Only if WAVE = 0

TC Block Control Register

Register Name: TC_BCR

Access Type: Write only

Offset: 0xC0

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

- **SYNC: Synchro Command**

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

Reset Value: 0

Offset: 0xC4

| | | | | | | | |
|----|----|---------|----|---------|----|---------|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | – | – | – |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| – | – | TC2XC2S | | TC1XC1S | | TC0XC0S | |

- **TC0XC0S: External Clock Signal 0 Selection**

| TC0XC0S | | Signal Connected to XC0 |
|---------|---|-------------------------|
| 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

Offset: 0x00

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

- **CLKEN: Counter Clock Enable Command (Code Label TC_CLKEN)**

0 = No effect.

1 = Enables the clock if CLKDIS is not 1.

- **CLKDIS: Counter Clock Disable Command (Code Label TC_CLKDIS)**

0 = No effect.

1 = Disables the clock.

- **SWTRG: Software Trigger Command (Code Label TC_SWTRG)**

0 = No effect.

1 = A software trigger is performed: the counter is reset and clock is started.

TC Channel Mode Register: Capture Mode

Register Name: TC_CMR

Access Type: Read/Write

Reset Value: 0

Offset: 0x04

| | | | | | | | |
|----------|---------|-------|----|------|--------|---------|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | LDRB | | LDRA | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| WAVE = 0 | CPCTRG | – | – | – | ABETRG | ETRGEDG | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LDBDIS | LDBSTOP | BURST | | CLKI | TCCLKS | | |

• TCCLKS: Clock Selection

| TCCLKS | | | Clock Selected | Code Label |
|--------|---|---|----------------|-----------------|
| | | | | TC_CLKS |
| 0 | 0 | 0 | MCK/2 | TC_CLKS_MCK2 |
| 0 | 0 | 1 | MCK/8 | TC_CLKS_MCK8 |
| 0 | 1 | 0 | MCK/32 | TC_CLKS_MCK32 |
| 0 | 1 | 1 | MCK/128 | TC_CLKS_MCK128 |
| 1 | 0 | 0 | MCK/1024 | TC_CLKS_MCK1024 |
| 1 | 0 | 1 | XC0 | TC_CLKS_XC0 |
| 1 | 1 | 0 | XC1 | TC_CLKS_XC1 |
| 1 | 1 | 1 | XC2 | TC_CLKS_XC2 |

• CLKI: Clock Invert (Code Label TC_CLKI)

0 = Counter is incremented on rising edge of the clock.

1 = Counter is incremented on falling edge of the clock.

• BURST: Burst Signal Selection

| BURST | | Selected BURST | Code Label |
|-------|---|--|---------------|
| | | | TC_BURST |
| 0 | 0 | The clock is not gated by an external signal | TC_BURST_NONE |
| 0 | 1 | XC0 is ANDed with the selected clock | TC_BURST_XC0 |
| 1 | 0 | XC1 is ANDed with the selected clock | TC_BURST_XC1 |
| 1 | 1 | XC2 is ANDed with the selected clock | TC_BURST_XC2 |

• LDBSTOP: Counter Clock Stopped with RB Loading (Code Label TC_LDBSTOP)

0 = Counter clock is not stopped when RB loading occurs.

1 = Counter clock is stopped when RB loading occurs.

- **LDBDIS: Counter Clock Disable with RB Loading (Code Label TC_LDBDIS)**

0 = Counter clock is not disabled when RB loading occurs.

1 = Counter clock is disabled when RB loading occurs.

- **ETRGEDG: External Trigger Edge Selection**

| ETRGEDG | | Edge | Code Label |
|---------|---|--------------|-------------------------|
| | | | TC_ETRGEDG |
| 0 | 0 | None | TC_ETRGEDG_EDGE_NONE |
| 0 | 1 | Rising Edge | TC_ETRGEDG_RISING_EDGE |
| 1 | 0 | Falling Edge | TC_ETRGEDG_FALLING_EDGE |
| 1 | 1 | Each Edge | TC_ETRGEDG_BOTH_EDGE |

- **ABETRG: TIOA or TIOB External Trigger Selection**

| ABETRG | Selected ABETRG | Code Label |
|--------|--------------------------------------|----------------|
| | | TC_ABETRG |
| 0 | TIOB is used as an external trigger. | TC_ABETRG_TIOB |
| 1 | TIOA is used as an external trigger. | TC_ABETRG_TIOA |

- **CPCTRG: RC Compare Trigger Enable (Code Label TC_CPCTRG)**

0 = RC Compare has no effect on the counter and its clock.

1 = RC Compare resets the counter and starts the counter clock.

- **WAVE = 0 (Code Label TC_WAVE)**

0 = Capture Mode is enabled.

1 = Capture Mode is disabled (Waveform Mode is enabled).

- **LDRA: RA Loading Selection**

| LDRA | | Edge | Code Label |
|------|---|----------------------|----------------------|
| | | | TC_LDRA |
| 0 | 0 | None | TC_LDRA_EDGE_NONE |
| 0 | 1 | Rising edge of TIOA | TC_LDRA_RISING_EDGE |
| 1 | 0 | Falling edge of TIOA | TC_LDRA_FALLING_EDGE |
| 1 | 1 | Each edge of TIOA | TC_LDRA_BOTH_EDGE |

- **LDRB: RB Loading Selection**

| LDRB | | Edge | Code Label |
|------|---|----------------------|----------------------|
| | | | TC_LDRB |
| 0 | 0 | None | TC_LDRB_EDGE_NONE |
| 0 | 1 | Rising edge of TIOA | TC_LDRB_RISING_EDGE |
| 1 | 0 | Falling edge of TIOA | TC_LDRB_FALLING_EDGE |
| 1 | 1 | Each edge of TIOA | TC_LDRB_BOTH_EDGE |

TC Channel Mode Register: Waveform Mode

Register Name: TC_CMR

Access Type: Read/Write

Reset Value: 0

Offset: 0x04

| | | | | | | | |
|----------|---------|-------|--------|------|--------|---------|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| BSWTRG | | BEEVT | | BCPC | | BCPB | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ASWTRG | | AEEVT | | ACPC | | ACPA | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| WAVE = 1 | CPCTRG | – | ENETRГ | EEVT | | EEVTEDG | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CPCDIS | CPCSTOP | BURST | | CLKI | TCCLKS | | |

- TCCLKS: Clock Selection**

| TCCLKS | | | Clock Selected | Code Label |
|--------|---|---|----------------|-----------------|
| | | | | TC_CLKS |
| 0 | 0 | 0 | MCK/2 | TC_CLKS_MCK2 |
| 0 | 0 | 1 | MCK/8 | TC_CLKS_MCK8 |
| 0 | 1 | 0 | MCK/32 | TC_CLKS_MCK32 |
| 0 | 1 | 1 | MCK/128 | TC_CLKS_MCK128 |
| 1 | 0 | 0 | MCK/1024 | TC_CLKS_MCK1024 |
| 1 | 0 | 1 | XC0 | TC_CLKS_XC0 |
| 1 | 1 | 0 | XC1 | TC_CLKS_XC1 |
| 1 | 1 | 1 | XC2 | TC_CLKS_XC2 |

- CLKI: Clock Invert (Code Label TC_CLKI)**

0 = Counter is incremented on rising edge of the clock.

1 = Counter is incremented on falling edge of the clock.

- BURST: Burst Signal Selection**

| BURST | | Selected BURST | Code Label |
|-------|---|---|---------------|
| | | | TC_BURST |
| 0 | 0 | The clock is not gated by an external signal. | TC_BURST_NONE |
| 0 | 1 | XC0 is ANDed with the selected clock. | TC_BURST_XC0 |
| 1 | 0 | XC1 is ANDed with the selected clock. | TC_BURST_XC1 |
| 1 | 1 | XC2 is ANDed with the selected clock. | TC_BURST_XC2 |

- CPCSTOP: Counter Clock Stopped with RC Compare (Code Label TC_CPCSTOP)**

0 = Counter clock is not stopped when counter reaches RC.

1 = Counter clock is stopped when counter reaches RC.

- **CPCDIS: Counter Clock Disable with RC Compare (Code Label TC_CPCDIS)**

0 = Counter clock is not disabled when counter reaches RC.

1 = Counter clock is disabled when counter reaches RC.

- **EEVTEDG: External Event Edge Selection**

| EEVTEDG | | Edge | Code Label |
|---------|---|--------------|-------------------------|
| | | | TC_EEVTEDG |
| 0 | 0 | None | TC_EEVTEDG_EDGE_NONE |
| 0 | 1 | Rising edge | TC_EEVTEDG_RISING_EDGE |
| 1 | 0 | Falling edge | TC_EEVTEDG_FALLING_EDGE |
| 1 | 1 | Each edge | TC_EEVTEDG_BOTH_EDGE |

- **EEVT: External Event Selection**

| EEVT | | Signal Selected as External Event | TIOB Direction | Code Label |
|------|---|-----------------------------------|----------------------|--------------|
| | | | | TC_EEVT |
| 0 | 0 | TIOB | Input ⁽¹⁾ | TC_EEVT_TIOB |
| 0 | 1 | XC0 | Output | TC_EEVT_XC0 |
| 1 | 0 | XC1 | Output | TC_EEVT_XC1 |
| 1 | 1 | XC2 | Output | TC_EEVT_XC2 |

Note: If TIOB is chosen as the external event signal, it is configured as an input and no longer generates waveforms.

- **ENETRГ: External Event Trigger Enable (Code Label TC_ENETRГ)**

0 = The external event has no effect on the counter and its clock. In this case, the selected external event only controls the TIOA output.

1 = The external event resets the counter and starts the counter clock.

- **CPCTRГ: RC Compare Trigger Enable (Code Label TC_CPCTRГ)**

0 = RC Compare has no effect on the counter and its clock.

1 = RC Compare resets the counter and starts the counter clock.

- **WAVE = 1 (Code Label TC_WAVE)**

0 = Waveform Mode is disabled (Capture Mode is enabled).

1 = Waveform Mode is enabled.

- **ACPA: RA Compare Effect on TIOA**

| ACPA | | Effect | Code Label |
|------|---|--------|-----------------------|
| | | | TC_ACPA |
| 0 | 0 | None | TC_ACPA_OUTPUT_NONE |
| 0 | 1 | Set | TC_ACPA_SET_OUTPUT |
| 1 | 0 | Clear | TC_ACPA_CLEAR_OUTPUT |
| 1 | 1 | Toggle | TC_ACPA_TOGGLE_OUTPUT |

• **ACPC: RC Compare Effect on TIOA**

| ACPC | | Effect | Code Label |
|------|---|--------|-----------------------|
| | | | TC_ACPC |
| 0 | 0 | None | TC_ACPC_OUTPUT_NONE |
| 0 | 1 | Set | TC_ACPC_SET_OUTPUT |
| 1 | 0 | Clear | TC_ACPC_CLEAR_OUTPUT |
| 1 | 1 | Toggle | TC_ACPC_TOGGLE_OUTPUT |

• **AEEVT: External Event Effect on TIOA**

| AEEVT | | Effect | Code Label |
|-------|---|--------|------------------------|
| | | | TC_AEEVT |
| 0 | 0 | None | TC_AEEVT_OUTPUT_NONE |
| 0 | 1 | Set | TC_AEEVT_SET_OUTPUT |
| 1 | 0 | Clear | TC_AEEVT_CLEAR_OUTPUT |
| 1 | 1 | Toggle | TC_AEEVT_TOGGLE_OUTPUT |

• **ASWTRG: Software Trigger Effect on TIOA**

| ASWTRG | | Effect | Code Label |
|--------|---|--------|-------------------------|
| | | | TC_ASWTRG |
| 0 | 0 | None | TC_ASWTRG_OUTPUT_NONE |
| 0 | 1 | Set | TC_ASWTRG_SET_OUTPUT |
| 1 | 0 | Clear | TC_ASWTRG_CLEAR_OUTPUT |
| 1 | 1 | Toggle | TC_ASWTRG_TOGGLE_OUTPUT |

• **BCPB: RB Compare Effect on TIOB**

| BCPB | | Effect | Code Label |
|------|---|--------|-----------------------|
| | | | TC_BCPB |
| 0 | 0 | None | TC_BCPB_OUTPUT_NONE |
| 0 | 1 | Set | TC_BCPB_SET_OUTPUT |
| 1 | 0 | Clear | TC_BCPB_CLEAR_OUTPUT |
| 1 | 1 | Toggle | TC_BCPB_TOGGLE_OUTPUT |

• **BCPC: RC Compare Effect on TIOB**

| BCPC | | Effect | Code Label |
|------|---|--------|-----------------------|
| | | | TC_BCPC |
| 0 | 0 | None | TC_BCPC_OUTPUT_NONE |
| 0 | 1 | Set | TC_BCPC_SET_OUTPUT |
| 1 | 0 | Clear | TC_BCPC_CLEAR_OUTPUT |
| 1 | 1 | Toggle | TC_BCPC_TOGGLE_OUTPUT |

- **BEEVT: External Event Effect on TIOB**

| BEEVT | | Effect | Code Label |
|-------|---|--------|------------------------|
| | | | TC_BEEVT |
| 0 | 0 | None | TC_BEEVT_OUTPUT_NONE |
| 0 | 1 | Set | TC_BEEVT_SET_OUTPUT |
| 1 | 0 | Clear | TC_BEEVT_CLEAR_OUTPUT |
| 1 | 1 | Toggle | TC_BEEVT_TOGGLE_OUTPUT |

- **BSWTRG: Software Trigger Effect on TIOB**

| BSWTRG | | Effect | Code Label |
|--------|---|--------|-------------------------|
| | | | TC_BSWTRG |
| 0 | 0 | None | TC_BSWTRG_OUTPUT_NONE |
| 0 | 1 | Set | TC_BSWTRG_SET_OUTPUT |
| 1 | 0 | Clear | TC_BSWTRG_CLEAR_OUTPUT |
| 1 | 1 | Toggle | TC_BSWTRG_TOGGLE_OUTPUT |

TC Counter Value Register

Register Name: TC_CVR
Access Type: Read Only
Reset Value: 0
Offset: 0x10

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

- CV: Counter Value (Code Label TC_CV)**

CV contains the counter value in real-time.

TC Register A

Register Name: TC_RA
Access Type: Read Only if WAVE = 0, Read/Write if WAVE = 1
Reset Value: 0
Offset: 0x14

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

- RA: Register A (Code Label TC_RA)**

RA contains the Register A value in real-time.

TC Register B

Register Name: TC_RB

Access Type: Read Only if WAVE = 0, Read/Write if WAVE = 1

Reset Value: 0

Offset: 0x18

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

- **RB: Register B (Code Label TC_RB)**

RB contains the Register B value in real-time.

TC Register C

Register Name: TC_RC

Access Type: Read/Write

Reset Value: 0

Offset: 0x1C

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

- **RC: Register C (Code Label TC_RC)**

RC contains the Register C value in real-time.

TC Status Register

Register Name: TC_SR

Access Type: Read Only

Reset Value: 0

Offset: 0x20

| | | | | | | | |
|-------|-------|-------|------|------|-------|-------|--------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| – | – | – | – | – | – | – | – |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | – | – | – | – | MTIOB | MTIOA | CLKSTA |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| – | – | – | – | – | – | – | – |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ETRGS | LDRBS | LDRAS | CPCS | CPBS | CPAS | LOVRS | COVFS |

- **COVFS: Counter Overflow Status (Code Label TC_COVFS)**

0 = No counter overflow has occurred since the last read of the Status Register.

1 = A counter overflow has occurred since the last read of the Status Register.

- **LOVRS: Load Overrun Status (Code Label TC_LOVRS)**

0 = Load overrun has not occurred since the last read of the Status Register or WAVE = 1.

1 = RA or RB have been loaded at least twice without any read of the corresponding register since the last read of the Status Register, if WAVE = 0.

- **CPAS: RA Compare Status (Code Label TC_CPAS)**

0 = RA Compare has not occurred since the last read of the Status Register or WAVE = 0.

1 = RA Compare has occurred since the last read of the Status Register, if WAVE = 1.

- **CPBS: RB Compare Status (Code Label TC_CPBS)**

0 = RB Compare has not occurred since the last read of the Status Register or WAVE = 0.

1 = RB Compare has occurred since the last read of the Status Register, if WAVE = 1.

- **CPCS: RC Compare Status (Code Label TC_CPCS)**

0 = RC Compare has not occurred since the last read of the Status Register.

1 = RC Compare has occurred since the last read of the Status Register.

- **LDRAS: RA Loading Status (Code Label TC_LDRAS)**

0 = RA Load has not occurred since the last read of the Status Register or WAVE = 1.

1 = RA Load has occurred since the last read of the Status Register, if WAVE = 0.

- **LDRBS: RB Loading Status (Code Label TC_LDRBS)**

0 = RB Load has not occurred since the last read of the Status Register or WAVE = 1.

1 = RB Load has occurred since the last read of the Status Register, if WAVE = 0.

- **ETRGS: External Trigger Status (Code Label TC_ETRGS)**

0 = External trigger has not occurred since the last read of the Status Register.

1 = External trigger has occurred since the last read of the Status Register.

- **CLKSTA: Clock Enabling Status (Code Label TC_CLKSTA)**

0 = Clock is disabled.

1 = Clock is enabled.

- **MTIOA: TIOA Mirror (Code Label TC_MTIOA)**

0 = TIOA is low. If WAVE = 0, this means that TIOA pin is low. If WAVE = 1, this means that TIOA is driven low.

1 = TIOA is high. If WAVE = 0, this means that TIOA pin is high. If WAVE = 1, this means that TIOA is driven high.

- **MTIOB: TIOB Mirror (Code Label TC_MTIOB)**

0 = TIOB is low. If WAVE = 0, this means that TIOB pin is low. If WAVE = 1, this means that TIOB is driven low.

1 = TIOB is high. If WAVE = 0, this means that TIOB pin is high. If WAVE = 1, this means that TIOB is driven high.

TC Interrupt Enable Register

Register Name: TC_IER

Access Type: Write only

Offset: 0x24

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

- **COVFS: Counter Overflow (Code Label TC_COVFS)**

0 = No effect.

1 = Enables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun (Code Label TC_LOVRS)**

0 = No effect.

1: Enables the Load Overrun Interrupt.

- **CPAS: RA Compare (Code Label TC_CPAS)**

0 = No effect.

1 = Enables the RA Compare Interrupt.

- **CPBS: RB Compare (Code Label TC_CPBS)**

0 = No effect.

1 = Enables the RB Compare Interrupt.

- **CPCS: RC Compare (Code Label TC_CPCS)**

0 = No effect.

1 = Enables the RC Compare Interrupt.

- **LDRAS: RA Loading (Code Label TC_LDRAS)**

0 = No effect.

1 = Enables the RA Load Interrupt.

- **LDRBS: RB Loading (Code Label TC_LDRBS)**

0 = No effect.

1 = Enables the RB Load Interrupt.

- **ETRGS: External Trigger (Code Label TC_ETRGS)**

0 = No effect.

1 = Enables the External Trigger Interrupt.

TC Interrupt Disable Register

Register Name: TC_IDR

Access Type: Write only

Offset: 0x28

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

- **COVFS: Counter Overflow (Code Label TC_COVFS)**

0 = No effect.

1 = Disables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun (Code Label TC_LOVRS)**

0 = No effect.

1 = Disables the Load Overrun Interrupt (if WAVE = 0).

- **CPAS: RA Compare (Code Label TC_CPAS)**

0 = No effect.

1 = Disables the RA Compare Interrupt (if WAVE = 1).

- **CPBS: RB Compare (Code Label TC_CPBS)**

0 = No effect.

1 = Disables the RB Compare Interrupt (if WAVE = 1).

- **CPCS: RC Compare (Code Label TC_CPCS)**

0 = No effect.

1 = Disables the RC Compare Interrupt.

- **LDRAS: RA Loading (Code Label TC_LDRAS)**

0 = No effect.

1 = Disables the RA Load Interrupt (if WAVE = 0).

- **LDRBS: RB Loading (Code Label TC_LDRBS)**

0 = No effect.

1 = Disables the RB Load Interrupt (if WAVE = 0).

- **ETRGS: External Trigger (Code Label TC_ETRGS)**

0 = No effect.

1 = Disables the External Trigger Interrupt.

TC Interrupt Mask Register

Register Name: TC_IMR

Access Type: Read Only

Reset Value: 0

Offset: 0x2C

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

- **COVFS: Counter Overflow (Code Label TC_COVFS)**

0 = The Counter Overflow Interrupt is disabled.

1 = The Counter Overflow Interrupt is enabled.

- **LOVRS: Load Overrun (Code Label TC_LOVRS)**

0 = The Load Overrun Interrupt is disabled.

1 = The Load Overrun Interrupt is enabled.

- **CPAS: RA Compare (Code Label TC_CPAS)**

0 = The RA Compare Interrupt is disabled.

1 = The RA Compare Interrupt is enabled.

- **CPBS: RB Compare (Code Label TC_CPBS)**

0 = The RB Compare Interrupt is disabled.

1 = The RB Compare Interrupt is enabled.

- **CPCS: RC Compare (Code Label TC_CPCS)**

0 = The RC Compare Interrupt is disabled.

1 = The RC Compare Interrupt is enabled.

- **LDRAS: RA Loading (Code Label TC_LDRAS)**

0 = The Load RA Interrupt is disabled.

1 = The Load RA Interrupt is enabled.

- **LDRBS: RB Loading (Code Label TC_LDRBS)**

0 = The Load RB Interrupt is disabled.

1 = The Load RB Interrupt is enabled.

- **ETRGS: External Trigger (Code Label TC_ETRGS)**

0 = The External Trigger Interrupt is disabled.

1 = The External Trigger Interrupt is enabled.



Table of Contents

| | |
|--|-----------|
| Features..... | 1 |
| Description..... | 1 |
| Pin Configuration..... | 2 |
| Block Diagram..... | 4 |
| Architectural Overview..... | 5 |
| Memories..... | 5 |
| Peripherals..... | 5 |
| Associated Documentation | 7 |
| Product Overview | 8 |
| Power Supply..... | 8 |
| Input/Output Considerations | 8 |
| Master Clock..... | 8 |
| Reset | 8 |
| Emulation Function | 8 |
| Memory Controller | 9 |
| External Bus Interface | 11 |
| Peripherals | 11 |
| System Peripherals..... | 12 |
| User Peripherals | 14 |
| Memory Map..... | 15 |
| Peripheral Memory Map | 17 |
| EBI: External Bus Interface..... | 18 |
| External Memory Mapping..... | 18 |
| External Bus Interface Pin Description | 20 |
| Chip Select Lines..... | 21 |
| Data Bus Width..... | 23 |
| Byte Write or Byte Select Access | 24 |
| Boot on NCS0..... | 26 |
| Read Protocols | 27 |
| Write Data Hold Time | 28 |
| Wait States | 29 |
| Memory Access Waveforms | 33 |
| EBI User Interface | 45 |
| EBI Chip Select Register | 46 |
| EBI Remap Control Register | 48 |
| EBI Memory Control Register | 49 |

| | |
|---|-----------|
| PS: Power-saving | 50 |
| Peripheral Clocks..... | 50 |
| PS User Interface | 51 |
| PS Control Register | 52 |
| PS Peripheral Clock Enable Register | 53 |
| PS Peripheral Clock Disable Register | 54 |
| PS Peripheral Clock Status Register | 55 |
| | |
| AIC: Advanced Interrupt Controller | 56 |
| Hardware Interrupt Vectoring..... | 58 |
| Priority Controller | 58 |
| Interrupt Handling | 58 |
| Interrupt Masking | 58 |
| Interrupt Clearing and Setting..... | 59 |
| Fast Interrupt Request | 59 |
| Software Interrupt | 59 |
| Spurious Interrupt | 59 |
| Protect Mode | 60 |
| AIC User Interface | 61 |
| AIC Source Mode Register | 62 |
| AIC Source Vector Register..... | 63 |
| AIC Interrupt Vector Register..... | 64 |
| AIC FIQ Vector Register | 64 |
| AIC Interrupt Status Register..... | 65 |
| AIC Interrupt Pending Register | 65 |
| AIC Interrupt Mask Register | 66 |
| AIC Core Interrupt Status Register | 67 |
| AIC Interrupt Enable Command Register | 68 |
| AIC Interrupt Disable Command Register | 68 |
| AIC Interrupt Clear Command Register | 69 |
| AIC Interrupt Set Command Register | 69 |
| AIC End of Interrupt Command Register | 70 |
| AIC Spurious Vector Register | 70 |
| Standard Interrupt Sequence..... | 71 |
| | |
| PIO: Parallel I/O Controller..... | 74 |
| Multiplexed I/O Lines | 74 |
| Output Selection | 74 |
| I/O Levels..... | 74 |
| Filters | 74 |
| Interrupts..... | 75 |
| User Interface | 75 |
| PIO User Interface | 78 |
| PIO Enable Register | 79 |
| PIO Disable Register | 79 |
| PIO Status Register | 80 |

| | |
|--|-----------|
| PIO Output Enable Register | 81 |
| PIO Output Disable Register | 81 |
| PIO Output Status Register | 82 |
| PIO Input Filter Enable Register | 83 |
| PIO Input Filter Disable Register | 83 |
| PIO Input Filter Status Register | 84 |
| PIO Set Output Data Register | 85 |
| PIO Clear Output Data Register | 85 |
| PIO Output Data Status Register..... | 86 |
| PIO Pin Data Status Register | 86 |
| PIO Interrupt Enable Register..... | 87 |
| PIO Interrupt Disable Register..... | 87 |
| PIO Interrupt Mask Register | 88 |
| PIO Interrupt Status Register..... | 88 |
| | |
| WD: Watchdog Timer | 89 |
| WD User Interface | 90 |
| WD Overflow Mode Register | 90 |
| WD Clock Mode Register | 91 |
| WD Control Register..... | 92 |
| WD Status Register | 92 |
| WD Enabling Sequence..... | 93 |
| | |
| SF: Special Function Registers..... | 94 |
| Chip Identification | 94 |
| SF User Interface..... | 94 |
| Chip ID Register | 95 |
| Chip ID Extension Register..... | 96 |
| Reset Status Register..... | 97 |
| SF Memory Mode Register..... | 98 |
| SF Protect Mode Register | 98 |
| | |
| USART: Universal Synchronous/Asynchronous Receiver/Transmitter..... | 99 |
| Pin Description..... | 99 |
| Baud Rate Generator..... | 100 |
| Receiver..... | 101 |
| Transmitter..... | 103 |
| Break | 104 |
| Peripheral Data Controller | 105 |
| Interrupt Generation..... | 105 |
| Channel Modes..... | 105 |
| USART User Interface | 107 |
| USART Control Register..... | 108 |
| USART Mode Register | 110 |
| USART Interrupt Enable Register..... | 112 |

| | |
|--|------------|
| USART Interrupt Disable Register | 113 |
| USART Interrupt Mask Register | 114 |
| USART Channel Status Register..... | 115 |
| USART Receiver Holding Register..... | 117 |
| USART Transmitter Holding Register..... | 117 |
| USART Baud Rate Generator Register | 118 |
| USART Receiver Time-out Register..... | 119 |
| USART Transmitter Time-guard Register..... | 119 |
| USART Receive Pointer Register | 120 |
| USART Receive Counter Register | 120 |
| USART Transmit Pointer Register..... | 121 |
| USART Transmit Counter Register | 121 |
| TC: Timer Counter | 122 |
| Signal Name Description | 123 |
| Timer Counter Description..... | 123 |
| Capture Operating Mode | 126 |
| Waveform Operating Mode..... | 128 |
| TC User Interface | 131 |
| TC Block Control Register | 132 |
| TC Block Mode Register..... | 133 |
| TC Channel Control Register..... | 134 |
| TC Channel Mode Register: Capture Mode | 135 |
| TC Channel Mode Register: Waveform Mode..... | 137 |
| TC Counter Value Register..... | 141 |
| TC Register A | 141 |
| TC Register B | 142 |
| TC Register C | 142 |
| TC Status Register | 143 |
| TC Interrupt Enable Register | 145 |
| TC Interrupt Disable Register | 146 |
| TC Interrupt Mask Register..... | 147 |
| Table of Contents | i |



Atmel Headquarters

Corporate Headquarters

2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 487-2600

Europe

Atmel Sarl
Route des Arsenaux 41
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 Tsimhatsui
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
TEL 1(408) 441-0311
FAX 1(408) 436-4314

Microcontrollers

2325 Orchard Parkway
San Jose, CA 95131
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
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
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

e-mail

literature@atmel.com

Web Site

<http://www.atmel.com>



© Atmel Corporation 2002.

Atmel Corporation makes no warranty for the use of its products, other than those expressly contained in the Company's standard warranty which is detailed in Atmel's Terms and Conditions located on the Company's web site. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Atmel are granted by the Company in connection with the sale of Atmel products, expressly or by implication. Atmel's products are not authorized for use as critical components in life support devices or systems.

Atmel® is the registered trademark of Atmel.

ARM®, Thumb® and ARM Powered® are registered trademarks of ARM Ltd. ARM7TDMI™ and AMBA™ are the trademarks of ARM Ltd. Other terms and product names may be the trademarks of others.



Printed on recycled paper.