Features

- Utilizes the ARM7TDMI™ ARM® Thumb® Processor Core
  - High-performance 32-bit RISC Architecture
  - High-density 16-bit Instruction Set
  - Leader in MIPS/Watt
  - Embedded ICE (In-Circuit Emulation)
- 2K Bytes Internal RAM
- Fully-programmable External Bus Interface (EBI)
  - Maximum External Address Space of 64M Bytes
  - Up to 8 Chip Selects
  - Software Programmable 8/16-bit External Data Bus
- Multi-processor Interface (MPI)
  - High-performance External Processor Interface
  - 512 x 16-bit Dual-port RAM
- 8-channel Peripheral Data Controller
- 8-level Priority, Individually-maskable, Vectored Interrupt Controller
  - 5 External Interrupts, including a High-priority, Low-latency Interrupt Request
- 58 Programmable I/O Lines
- 6-channel 16-bit Timer/Counter
  - 6 External Clock Inputs
  - 2 Multi-purpose I/O Pins per Channel
- 3 USARTs
  - 2 Dedicated Peripheral Data Controller (PDC) Channels per USART
  - Support for up to 9-bit Data Transfers
- Master/Slave SPI Interface
  - 2 Dedicated Peripheral Data Controller (PDC) Channels
  - 8- to 16-bit Programmable Data Length
  - 4 External Slave Chip Selects
- Programmable Watchdog Timer
- Power Management Controller (PMC)
  - CPU and Peripherals can be Deactivated Individually
- IEEE 1149.1 JTAG Boundary Scan on all Active Pins
- Fully Static Operation: 0 Hz to 25 MHz (12 MHz @ 1.8V)
- 1.8V to 3.6V Core Operating Voltage Range
- 2.7V to 5.5V I/O Operating Voltage Range
- -40° to +85°C Operating Temperature Range
- Available in a 176-lead TQFP Package

Description

The AT91M63200 is a member of the Atmel AT91 16/32-bit Microcontroller family, which is based on the ARM7TDMI processor core.

This processor has a high-performance 32-bit RISC architecture with a high-density 16-bit instruction set and very low power consumption. In addition, a large number of internally banked registers result in very fast exception handling, making the device ideal for real-time control applications. The AT91 ARM-based MCU family also features Atmel's high-density, in-system programmable, nonvolatile memory technology.

The AT91M63200 has a direct connection to off-chip memory, including Flash, through the External Bus Interface.

The Multi-processor Interface (MPI) provides a high-performance interface with an external co-processor or a high-bandwidth peripheral.

The AT91M63200 is manufactured using Atmel's high-density CMOS technology. By combining the ARM7TDMI microcontroller core with on-chip SRAM, a multi-processor interface and a wide range of peripheral functions on a monolithic chip, the AT91M63200 provides a highly-flexible and cost-effective solution to many compute-intensive multi-processor applications.

AT91 ARM® Thumb® Microcontrollers

AT91M63200

Rev. 1028A–11/99

Pin Configuration

Table 1. AT91M63200 Pinout

Pin	AT91M63200
1	GND
2	GND
3	NCS0
4	NCS1
5	NCS2
6	NCS3
7	NLB/A0
8	A1
9	A2
10	A3
11	A4
12	A5
13	A6
14	A7
15	VDDIO
16	GND
17	A8
18	A9
19	A10
20	A11
21	A12
22	A13
23	A14
24	A15
25	A16
26	A17
27	A18
28	A19
29	VDDIO
30	GND
31	A20/CS7
32	A21/CS6
33	A22/CS5
34	A23/CS4
35	D0
36	D1
37	D2
38	D3
39	D4
40	D5
41	D6
42	D7
43	VDDCORE
44	VDDIO

Pin	AT91M63200
45	GND
46	GND
47	D8
48	D9
49	D10
50	D11
51	D12
52	D13
53	D14
54	D15
55	PB19/TCLK0
56	PB20/TIOA0
57	PB21/TIOB0
58	PB22/TCLK1
59	VDDIO
60	GND
61	PB23/TIOA1
62	PB24/TIOB1
63	PB25/TCLK2
64	PB26/TIOA2
65	PB27/TIOB2
66	PA0/TCLK3
67	PA1/TIOA3
68	PA2/TIOB3
69	PA3/TCLK4
70	PA4/TIOA4
71	PA5/TIOB4
72	PA6/TCLK5
73	VDDIO
74	GND
75	PA7/TIOA5
76	PA8/TIOB5
77	PA9/IRQ0
78	PA10/IRQ1
79	PA11/IRQ2
80	PA12/IRQ3
81	PA13/FIQ
82	PA14/SCK0
83	PA15/TXD0
84	PA16/RXD0
85	PA17/SCK1
86	PA18/TXD1/NTRI
87	VDDCORE
88	VDDIO

Pin	AT91M63200
89	GND
90	GND
91	PA19/RXD1
92	PA20/SCK2
93	PA21/TXD2
94	PA22/RXD2
95	PA23/SPCK
96	PA24/MISO
97	PA25/MOSI
98	PA26/NPCS0/NSS
99	PA27/NPCS1
100	PA28/NPCS2
101	PA29/NPCS3
102	MPI_A1
103	VDDIO
104	GND
105	MPI_A2
106	MPI_A3
107	MPI_A4
108	MPI_A5
109	MPI_A6
110	MPI_A7
111	MPI_A8
112	MPI_A9
113	MPI_NCS
114	MPI_RNW
115	MPI_BR
116	MPI_BG
117	VDDIO
118	GND
119	MPI_D0
120	MPI_D1
121	MPI_D2
122	MPI_D3
123	MPI_D4
124	MPI_D5
125	MPI_D6
126	MPI_D7
127	MPI_D8
128	MPI_D9
129	MPI_D10
130	MPI_D11
131	VDDCORE
132	VDDIO

Pin	AT91M63200
133	GND
134	GND
135	MPI_D12
136	MPI_D13
137	MPI_D14
138	MPI_D15
139	PB0/MPI_NOE
140	PB1/MPI_NLB
141	PB2/MPI_NUB
142	PB3
143	PB4
144	PB5
145	PB6
146	PB7
147	VDDIO
148	GND
149	PB8
150	PB9
151	PB10
152	PB11
153	PB12
154	PB13
155	PB14
156	PB15
157	PB16
158	PB17/MCKO
159	NWDOVF
160	MCKI
161	VDDIO
162	GND
163	PB18/BMS
164	JTAGSEL
165	TMS
166	TDI
167	TDO
168	TCK
169	NTRST
170	NRST
171	NWAIT
172	NOE/NRD
173	NWE/NWR0
174	NUB/NWR1
175	VDDCORE
176	VDDIO

Pin Description

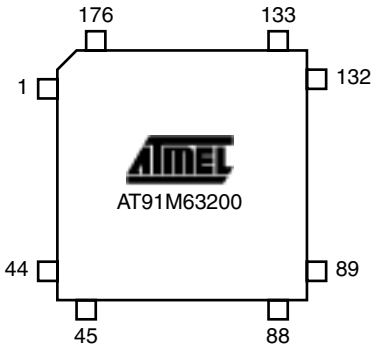
Table 2. AT91M63200 Pin Description

Module	Name	Function	Type	Active Level	Comments
EBI	A0 - A23	Address Bus	Output	–	All valid after reset
	D0 - D15	Data Bus	I/O	–	
	CS4 - CS7	Chip Select	Output	High	A23 - A20 after reset
	NCS0 - NCS3	Chip Select	Output	Low	
	NWR0	Lower Byte 0 Write Signal	Output	Low	Used in Byte Write option
	NWR1	Lower Byte 1 Write Signal	Output	Low	Used in Byte Write option
	NRD	Read Signal	Output	Low	Used in Byte Write option
	NWE	Write Enable	Output	Low	Used in Byte Select option
	NOE	Output Enable	Output	Low	Used in Byte Select option
	NUB	Upper Byte Select (16-bit SRAM)	Output	Low	Used in Byte Select option
	NLB	Lower Byte Select (16-bit SRAM)	Output	Low	Used in Byte Write option
	NWAIT	Wait Input	Input	Low	
	BMS	Boot Mode Select	Input	–	Sampled during reset
MPI	MPI_NCS	Chip Select	Input	Low	
	MPI_RNW	Read Not Write Signal	Input	–	
	MPI_BR	Bus Request from External Processor	Input	High	
	MPI_BG	Bus Grant to External Processor	Output	High	
	MPI_NOE	Output Enable	Input	Low	
	MPI_NLB	Lower Byte Select	Input	Low	
	MPI_NUB	Upper Byte Select	Input	Low	
	MPI_A1 - MPI_A9	Address Bus	Input	–	
	MPI_D0 - MPI_D15	Data Bus	I/O	–	
AIC	IRQ0 - IRQ3	External Interrupt Request	Input	–	PIO-controlled after reset
	FIQ	Fast External Interrupt Request	Input	–	PIO-controlled after reset
Timer	TCLK0 - TCLK5	Timer External Clock	Input	–	PIO-controlled after reset
	TIOA0 - TIOA5	Multi-purpose Timer I/O Pin A	I/O	–	PIO-controlled after reset
	TIOB0 - TIOB5	Multi-purpose Timer I/O Pin B	I/O	–	PIO-controlled after reset
USART	SCK0 - SCK2	External Serial Clock	I/O	–	PIO-controlled after reset
	TXD0 - TXD2	Transmit Data Output	Output	–	PIO-controlled after reset
	RXD0 - RXD2	Receive Data Input	Input	–	PIO-controlled after reset
SPI	SPCK	SPI Clock	I/O	–	PIO-controlled after reset
	MISO	Master In Slave Out	I/O	–	PIO-controlled after reset
	MOSI	Master Out Slave In	I/O	–	PIO-controlled after reset
	NSS	Slave Select	Input	Low	PIO-controlled after reset
	NPCS0 - NPCS3	Peripheral Chip Select	Output	Low	PIO-controlled after reset
PIO	PA0 - PA29	Programmable I/O Port A	I/O	–	Input after reset
	PB0 - PB27	Programmable I/O Port B	I/O	–	Input after reset
WD	NWDOVF	Watchdog Timer Overflow	Output	Low	Open drain
Clock	MCKI	Master Clock Input	Input	–	Schmitt trigger
	MCKO	Master Clock Output	Output	–	
Reset	NRST	Hardware Reset Input	Input	Low	Schmitt trigger, internal pull-up

Table 2. AT91M63200 Pin Description (Continued)

Module	Name	Function	Type	Active Level	Comments
JTAG/ICE	JTAGSEL	Selects between JTAG and ICE Mode	Input		High enables IEEE 1149.1 JTAG boundary scan Low enables ARM Standard ICE debug
	TMS	Test Mode Select	Input	–	Schmitt trigger, internal pull-up
	TDI	Test Data In	Input	–	Schmitt trigger, internal pull-up
	TDO	Test Data Out	Output	–	
	TCK	Test Clock	Input	–	Schmitt trigger, internal pull-up
	NTRST	Test Reset Input	Input	Low	Schmitt trigger, internal pull-up
Power	VDDIO	I/O Power	Power	–	3V or 5V nominal supply
	VDDCORE	Core Power	Power	–	2.0V or 3V nominal supply
	GND	Ground	Ground	–	
Emulation	NTRI	Tristate Mode Enable	Input	Low	Sampled during reset

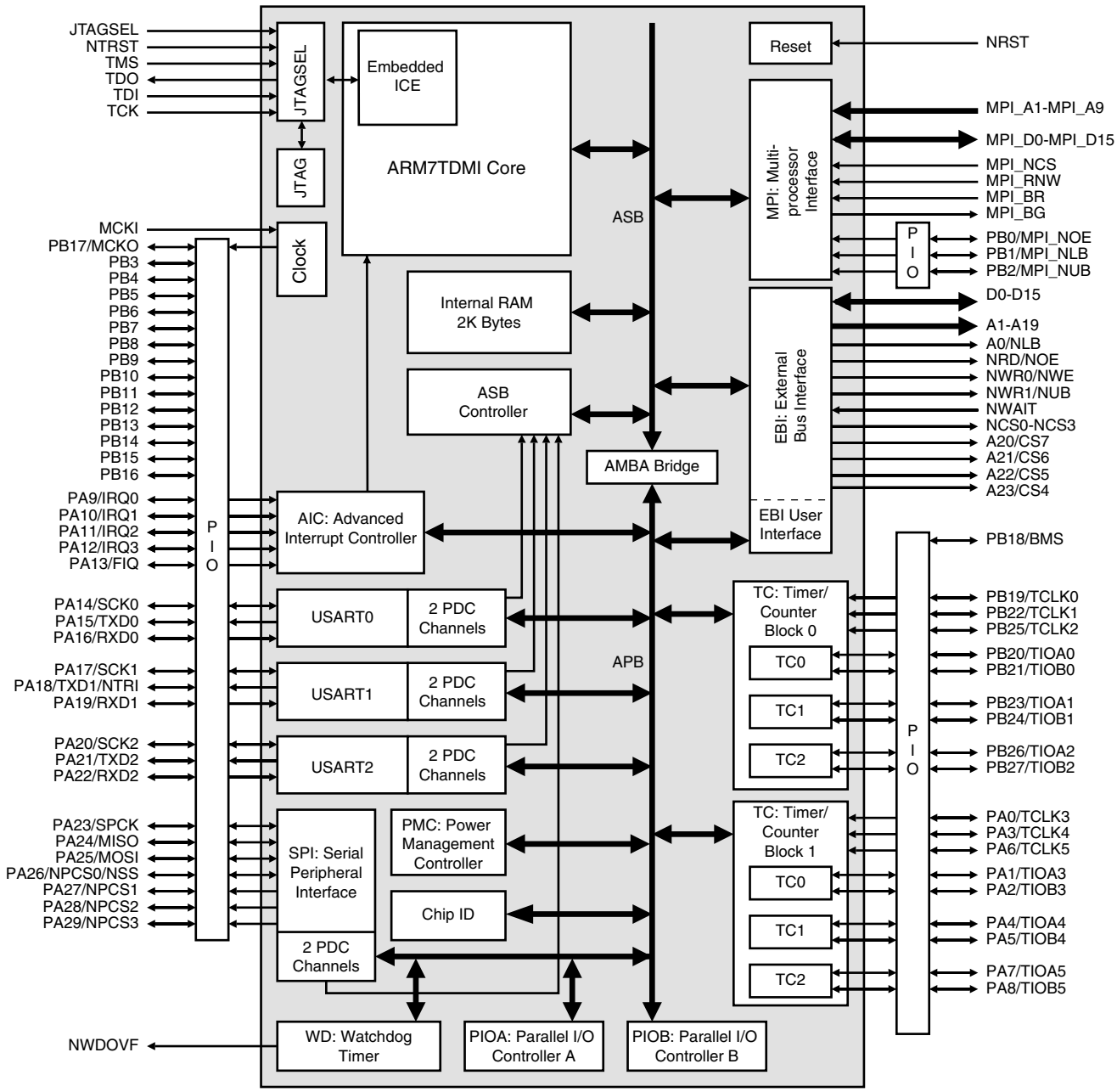
Figure 1. Pin Configuration (Top View)



176-lead TQFP

Block Diagram

Figure 2. AT91M63200



Architectural Overview

The AT91M63200 architecture consists of two main buses, the Advanced System Bus (ASB) and the Advanced Peripheral Bus (APB). The ASB is designed for maximum performance. It interfaces the processor with the on-chip 32-bit memories and the external memories and devices by means of the External Bus Interface (EBI). The APB is designed for accesses to on-chip peripherals and is optimized for low power consumption. The AMBA Bridge provides an interface between the ASB and the APB.

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

The AT91M63200 peripherals are designed to be easily programmable with a minimum number of instructions. Each peripheral has a 16K byte address space allocated in the upper 3M bytes of the 4G byte address space. Except for the interrupt controller, the peripheral base address is the lowest address of its memory space. The peripheral register set is composed of control, mode, data, status and interrupt registers.

To maximize the efficiency of bit manipulation, frequently-written registers are mapped into three memory locations. The first address is used to set the individual register bits, the second resets the bits and the third address reads the value stored in the register. A bit can be set or reset by writing a one to the corresponding position at the appropriate address. Writing a zero has no effect. Individual bits can thus be modified without having to use costly read-modify-write and complex bit-manipulation instructions.

All of the external signals of the on-chip peripherals are under the control of the Parallel I/O Controller. The PIO Controller can be programmed to insert an input filter on each pin or generate an interrupt on a signal change. After reset, the user must carefully program the PIO Controller in order to define which peripheral signals are connected with off-chip logic.

The ARM7TDMI processor operates in little-endian mode in the AT91M63200 microcontroller. The processor's internal architecture and the ARM and Thumb instruction sets are described in the ARM7TDMI datasheet. The memory map and the on-chip peripherals are described in the subsequent sections of this datasheet. Electrical and mechanical characteristics are documented in a separate datasheet entitled "AT91M63200 Electrical and Mechanical Characteristics" (Literature No. 1090).

The ARM standard In-Circuit Emulation debug interface is supported via the ICE port of the AT91M63200 via the JTAG/ICE port when JTAGSEL is low. IEEE JTAG boundary scan is supported via the JTAG/ICE port when JTAGSEL is high.

PDC: Peripheral Data Controller

The AT91M63200 has an 8-channel PDC dedicated to the three on-chip USARTs and to the SPI. One PDC channel is connected to the receiving channel and one to the transmitting channel of each peripheral.

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

Power Supplies

The AT91M63200 has two kinds of power supply pins:

- VDDCORE pins, which power the chip core
- VDDIO pins, which power the I/O lines

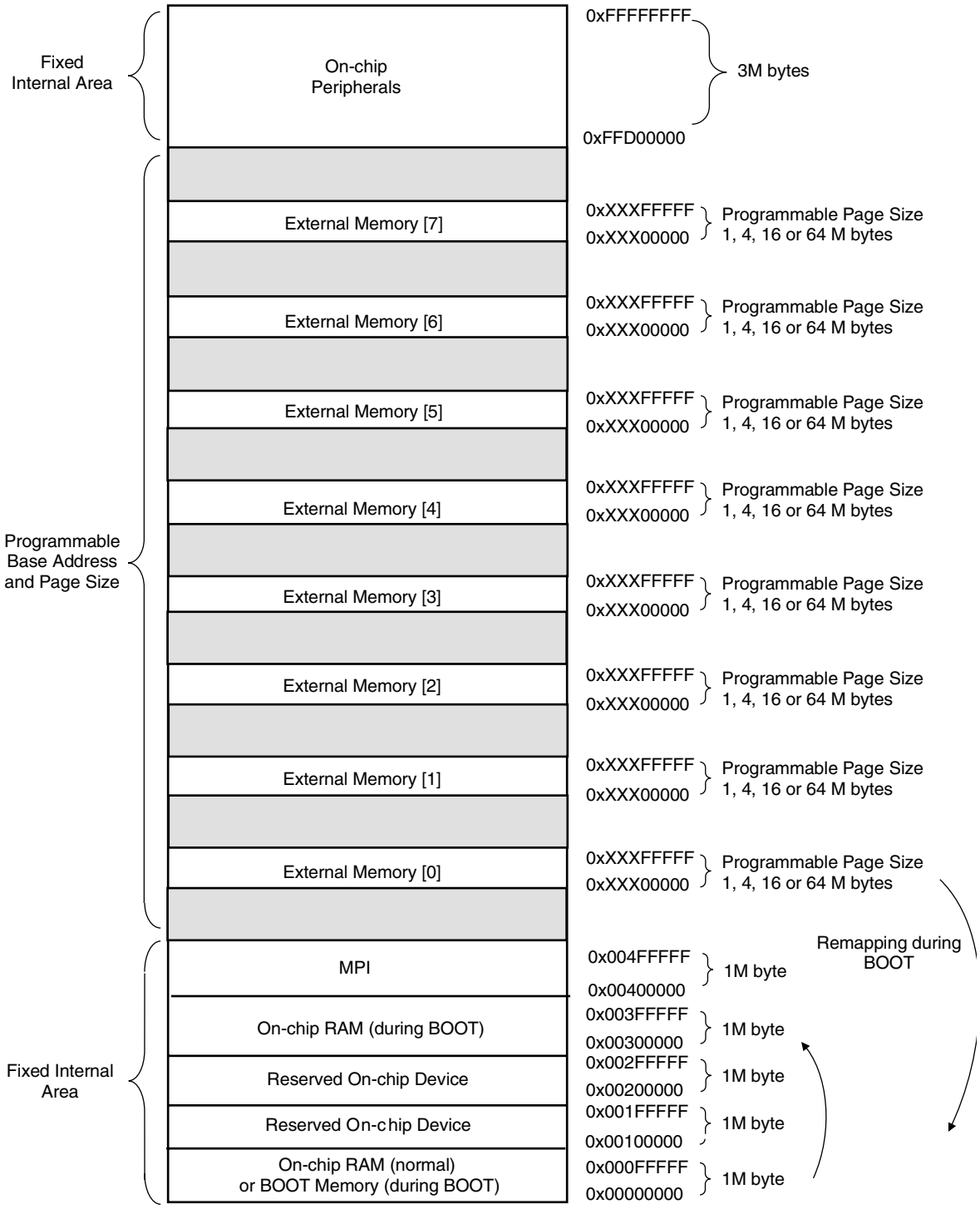
This allows core power consumption to be reduced by supplying it with a lower voltage than the I/O lines. The VDDCORE pins must never be powered at a voltage greater than the supply voltage applied to the VDDIO pins.

Typical supported voltage combinations are shown in the following table:

Pins	Typical Supply Voltages		
VDDCORE	3.0V or 3.3V	3.0V or 3.3V	2.0V
VDDIO	5.0V	3.0V or 3.3V	3.0V or 3.3V

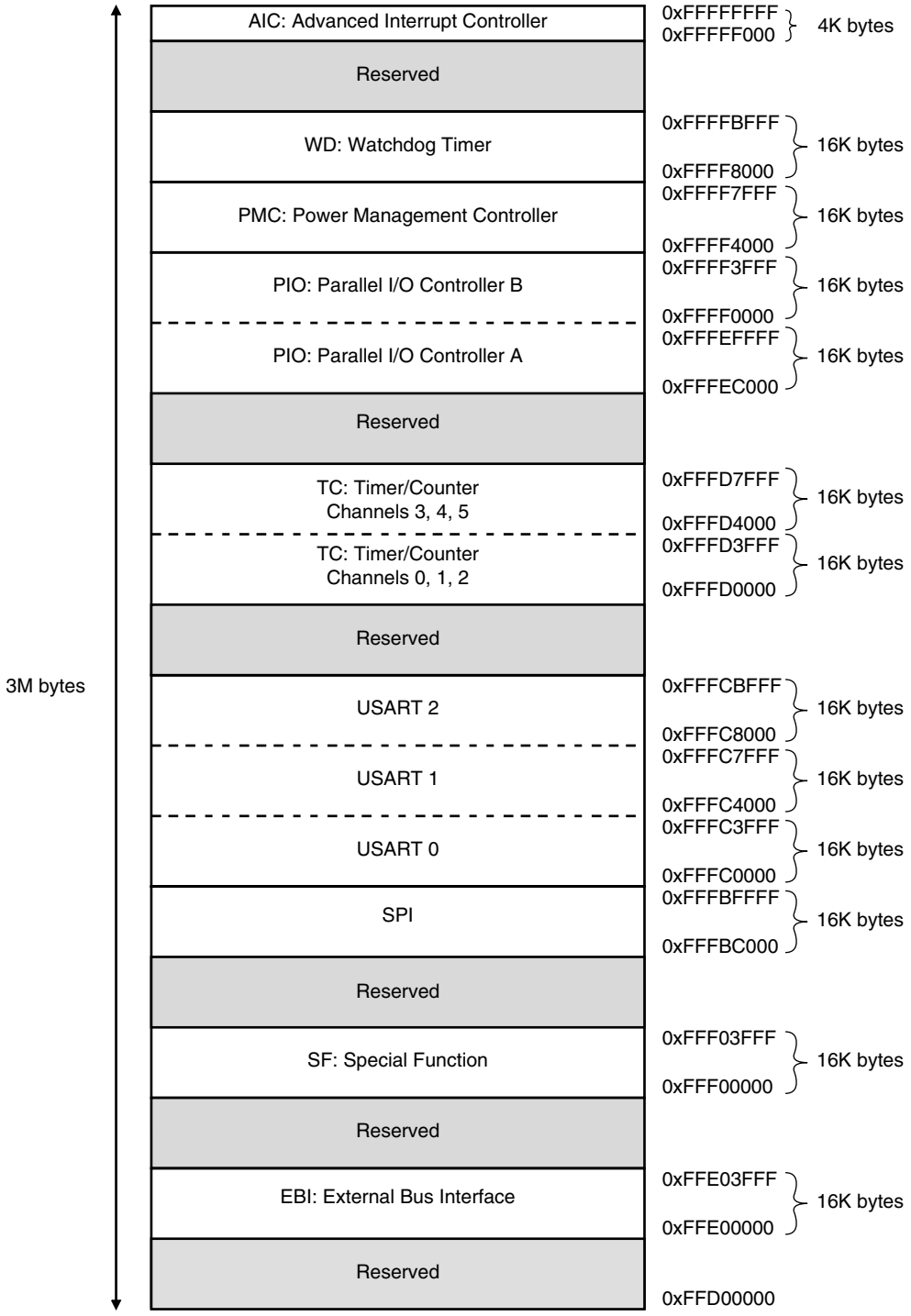
Memory Map

Figure 3. AT91M63200 Memory Map



Peripheral Memory Map

Figure 4. AT91M63200 Peripheral Memory Map



Initialization

Reset

Reset initializes the user interface registers to their default states as defined in the peripheral sections of this datasheet and forces the ARM7TDMI to perform the next instruction fetch from address zero. Except for the program counter, the ARM core registers do not have defined reset states. When reset is active, the inputs of the AT91M63200 must be held at valid logic levels. The EBI address lines drive low during reset. All the peripheral clocks are disabled during reset to save power (see "PMC: Power Management Controller" on page 139).

NRST Pin

NRST is the active low reset input. It is asserted asynchronously, but exit from reset is synchronized internally to MCKI. MCKI must be active within specification for a minimum of 10 clock cycles up to the rising edge of NRST to ensure correct operation.

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

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

Watchdog Reset

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

Boot Mode Select

The input level on the BMS pin during the last 10 clock cycles before the rising edge of NRST selects the type of boot memory. Boot operation is described on page 14. BMS must be driven to a valid logic value during reset.

The boot mode depends on BMS and whether the device has on-chip nonvolatile memory (NVM). See Table 3 below.

The correct logic level on BMS can be ensured with a resistor (pull-up or pull-down). See “AT91M63200 Electrical and Mechanical Characteristics” for the resistor value specification.

The BMS pin is multiplexed with parallel I/O PB18, which can be programmed after reset like any standard PIO.

Table 3. Boot Mode Select

BMS	Architecture	Boot Mode
1	No NVM	External 8-bit memory on NCS0
	NVM on-chip	Internal 32-bit NVM
0	All	External 16-bit memory on NCS0

Emulation Functions

Tristate Mode

The AT91M63200 provides a tristate mode, which is used for debug purposes in order to connect an emulator probe to an application board. In tristate mode the AT91M63200 continues to function, but all the output pin drivers are tristated.

To enter tristate 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 400KΩ. NTRI must be driven to a valid logic value during reset.

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

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

Embedded ICE

ARM standard embedded In-Circuit Emulation is supported via the JTAG/ICE port. It is connected to a host computer via an embedded ICE interface.

Embedded ICE mode is selected when JTAGSEL is low.

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

IEEE 1149.1 JTAG Boundary Scan

IEEE 1149.1 JTAG Boundary Scan is enabled when JTAGSEL is high. The functions SAMPLE, EXTEST and BYPASS are implemented.

In ICE Debug mode, the ARM core responds with a non-JTAG chip ID, which identifies the core to the ICE system. This is not IEEE 1149.1 JTAG compliant. See "SF: Special Function Registers" on page 145 for details on chip ID.

It is not possible to switch directly between JTAG and ICE operations. A chip reset must be performed (NRST and NTRST) after JTAGSEL is changed.

EBI: External Bus Interface

The EBI generates the signals which 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 eight 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 31.

External Memory Mapping

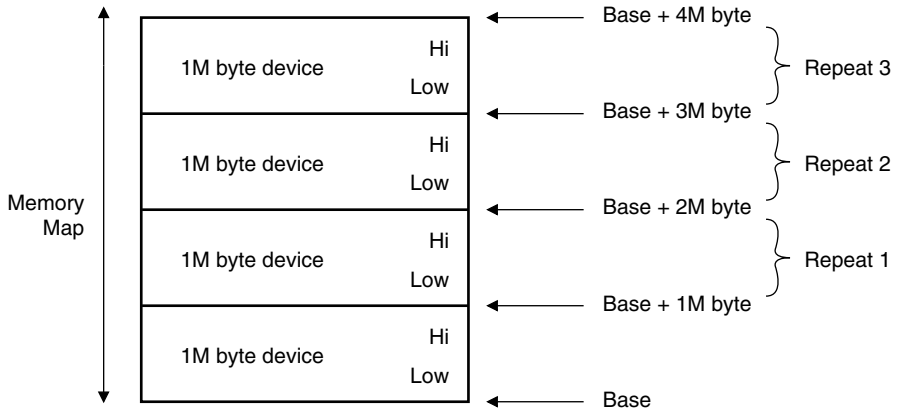
The memory map associates the internal 32-bit address space with the external 24-bit address bus.

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 5).

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).

Figure 5. External Memory Smaller than Page Size



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:

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:

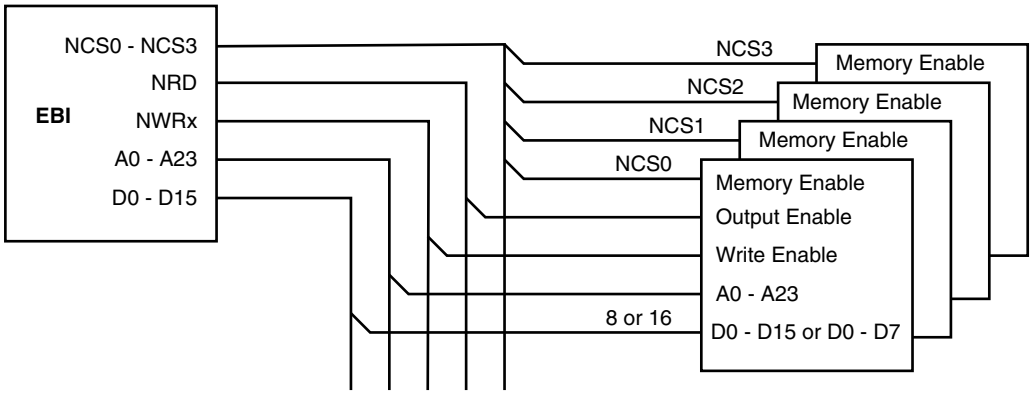
- Chip select lines NCS0 - NCS3 are dedicated to the EBI (not multiplexed).
- Chip select lines CS4 - CS7 are multiplexed with the top four address lines A23 - A20.

By exchanging address lines for chip select lines, the user can optimize the EBI to suit his 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:

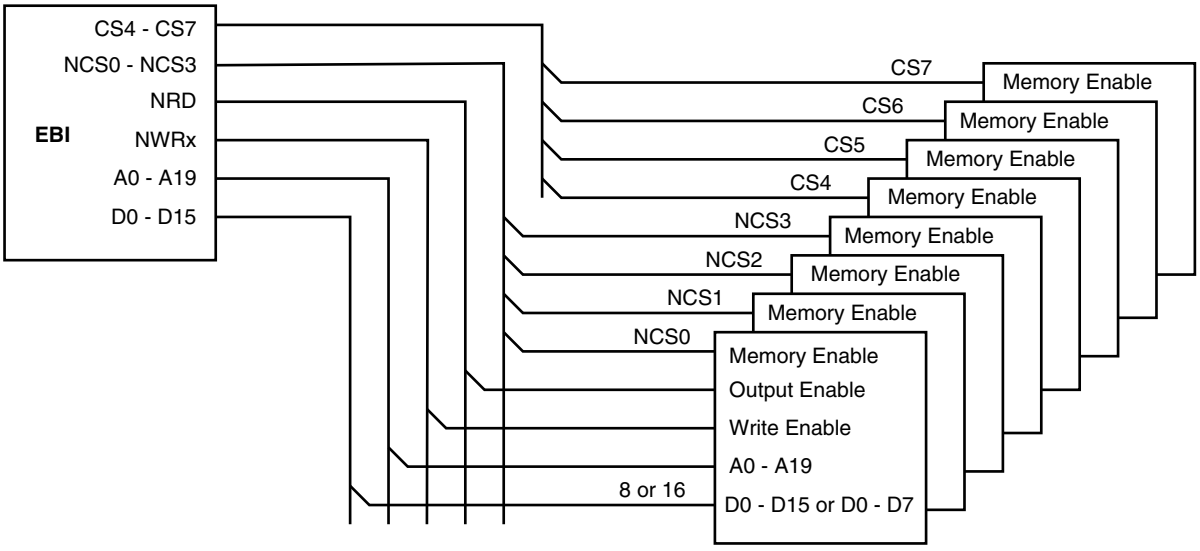
A20, A21, A22, A23 (configuration by default)
A20, A21, A22, CS4
A20, A21, CS5, CS4
A20, CS6, CS5, CS4
CS7, CS6, CS5, CS4

Figure 6. Memory Connections for Four External Devices



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

Figure 7. 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 8 shows how to connect a 512K x 8-bit memory on NCS2.

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

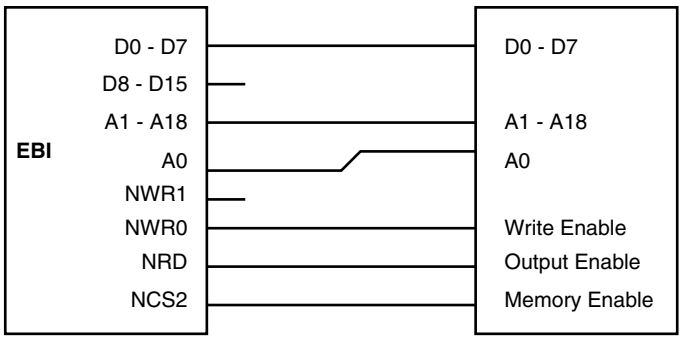
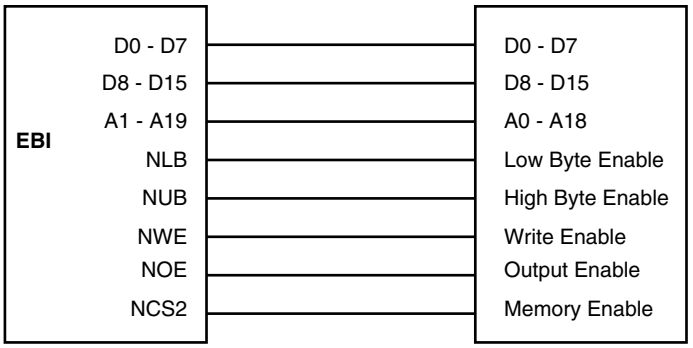


Figure 9 shows how to connect a 512K x 16-bit memory on NCS2.

Figure 9. 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.

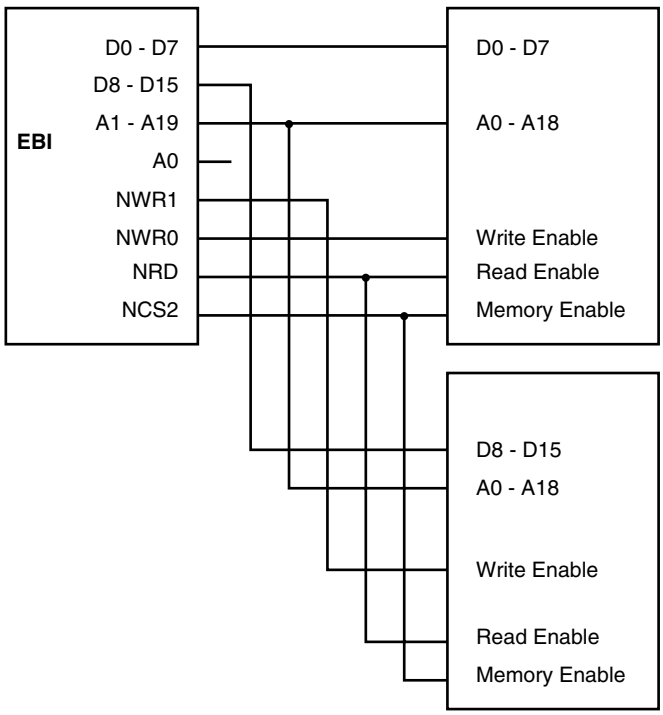
This option is controlled by the BAT field in the EBI_CSR (Chip Select Register) for the corresponding chip select.

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

Figure 10. 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 11 shows how to connect a 16-bit device with byte and half-word access (e.g. 16-bit SRAM) on NCS2.

Figure 11. Connection for a 16-bit Data Bus with Byte and Half-word Access

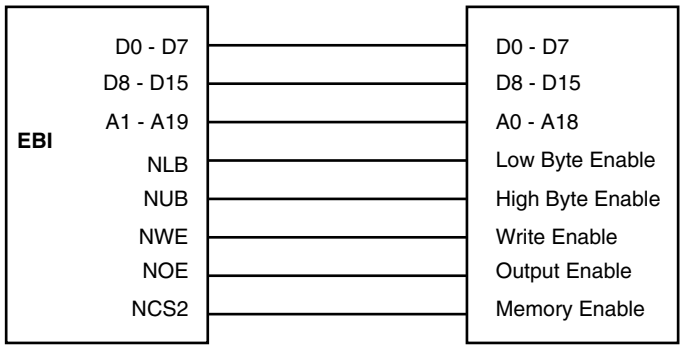
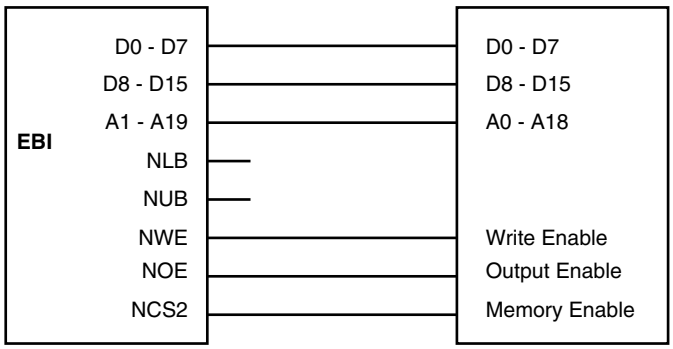


Figure 12 shows how to connect a 16-bit device without byte access (e.g. Flash) on NCS2.

Figure 12. Connection for a 16-bit Data Bus without Byte Write Capability



Boot

Conventional program operation requires RAM memory at page zero to support dynamic exception vectors. However, it is necessary to boot from nonvolatile memory at page zero.

When the AT91M63200 is reset, the memory map is modified to place NVM at page zero. The on-chip RAM is remapped to address 0x00300000 and either on-chip 32-bit NVM or off-chip 8/16-bit NVM is remapped to address 0x00000000. The off-chip NVM is selected on NCS0.

The boot memory type is selected by the BMS pin when NRST is active (see “Boot Mode Select” on page 9). Watchdog reset does not change the boot memory selection but does perform a full reboot from the previously-selected memory.

The memory map is returned to its conventional configuration by writing 1 to the RCB bit of the EBI_RCR (Remap Control Register). This cancels the remapping and enables normal operation of the EBI, as programmed (see page 34). It is not possible to remap the memory by writing 0 to the RCB bit in EBI_RCR.

During boot, the number of external devices (number of active chip selects) and their configurations must be programmed as described in the EBI user interface (see page 31). The chip select addresses which are programmed take effect when memory remapping is cancelled. Only NCS0 is active while the memory is remapped. Wait states take effect immediately when they are programmed to allow boot program execution to be optimized.

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 13). NWE is the same in both protocols. NWE always goes low in the second half of the master clock cycle (see Figure 14).

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 15). 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 13. Standard Read Protocol

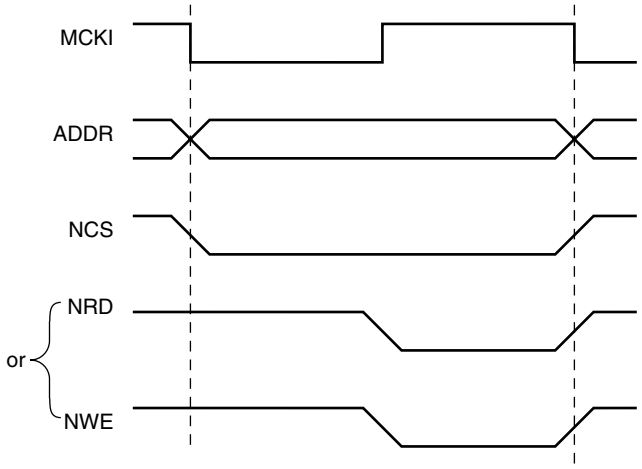


Figure 14. Early Read Protocol

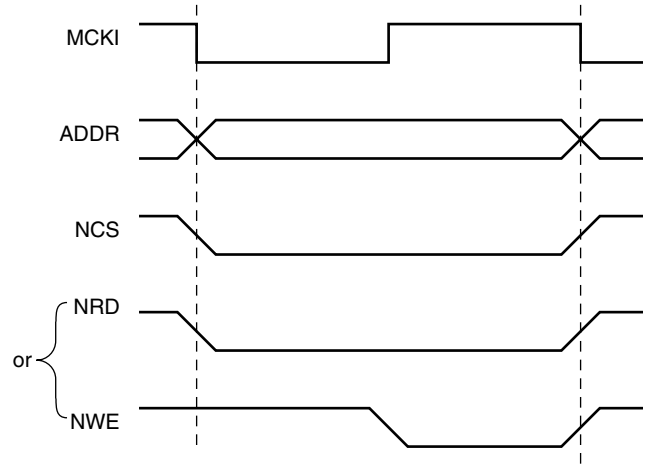
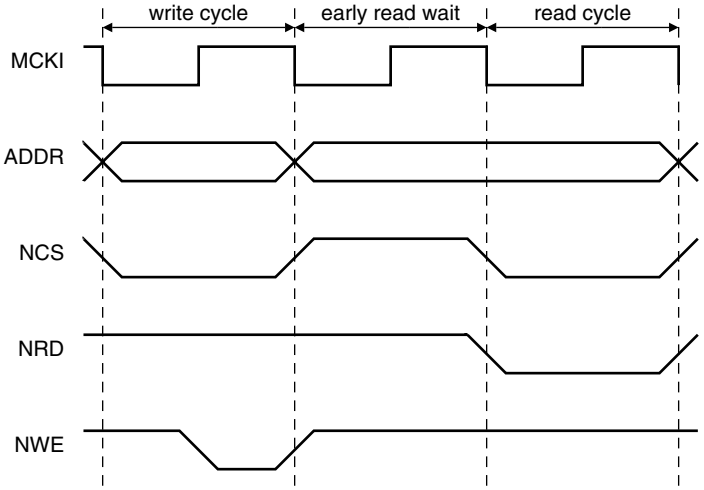


Figure 15. Early Read Wait State

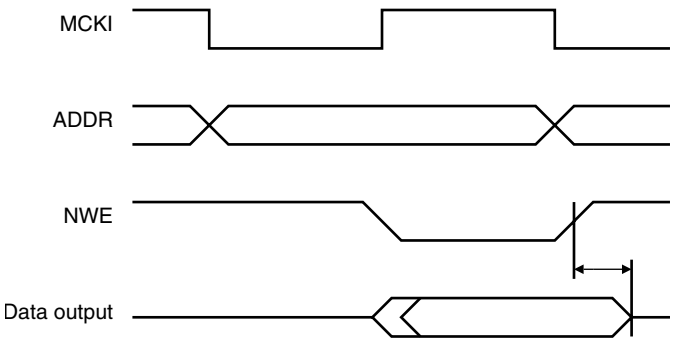


Write Data Hold Time

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 the figure below. 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 16. Data Hold Time



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

Wait States

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

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

Standard Wait States

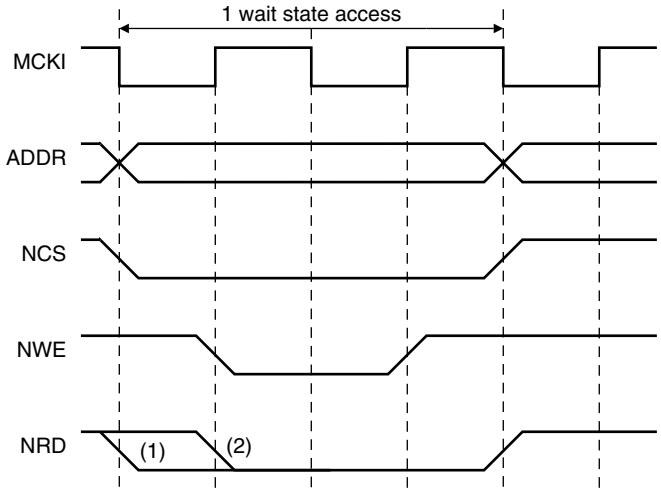
Each chip select can be programmed to insert one or more wait states during an access on the corresponding device. This is done by setting the WSE field in the corresponding EBI_CSR. The number of cycles to insert is programmed in the NWS field in the same register.

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 17. One Wait State Access



Notes: 1. Early read protocol
2. Standard read protocol

Data Float Wait State

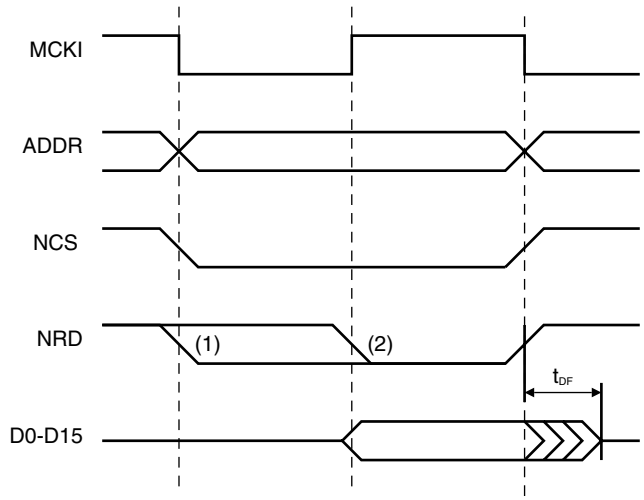
Some memory devices are slow to release the external bus. For such devices it is necessary to add wait states (data float waits) after a read access before starting a write access or a read access to a different external memory.

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.

Figure 18. Data Float Output Time



Notes: 1. Early read protocol
2. Standard read protocol

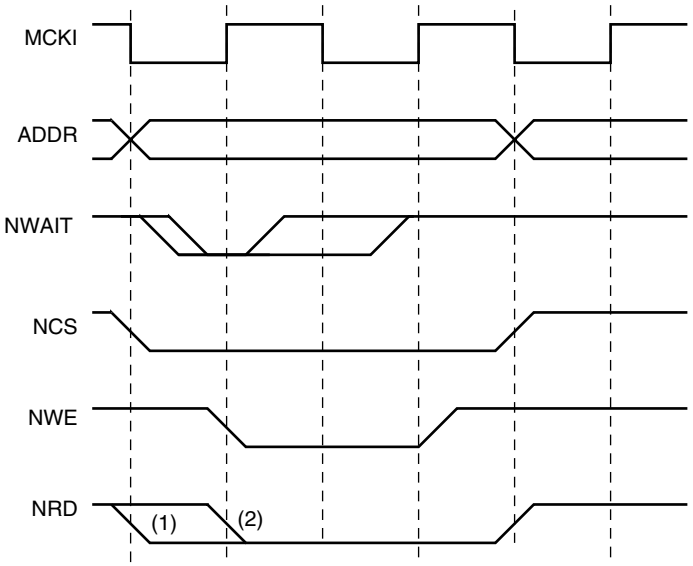
Internal memory accesses and consecutive accesses to the same external memory do not have added data float wait states.

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.

Figure 19. External Wait



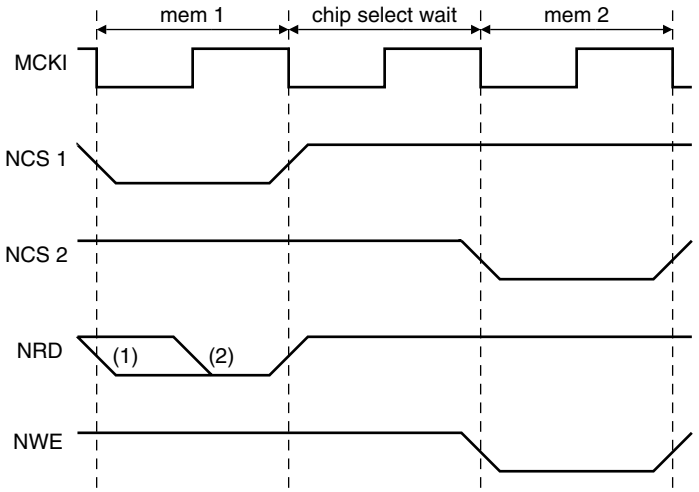
Notes: 1. Early read protocol
2. Standard read protocol

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

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 20. Chip Select Wait



Notes: 1. Early read protocol
2. Standard read protocol

Memory Access Waveforms

Figures 21 through 24 show examples of the two alternative protocols for external memory read access.

Figure 21. Standard Read Protocol with No t_{DF}

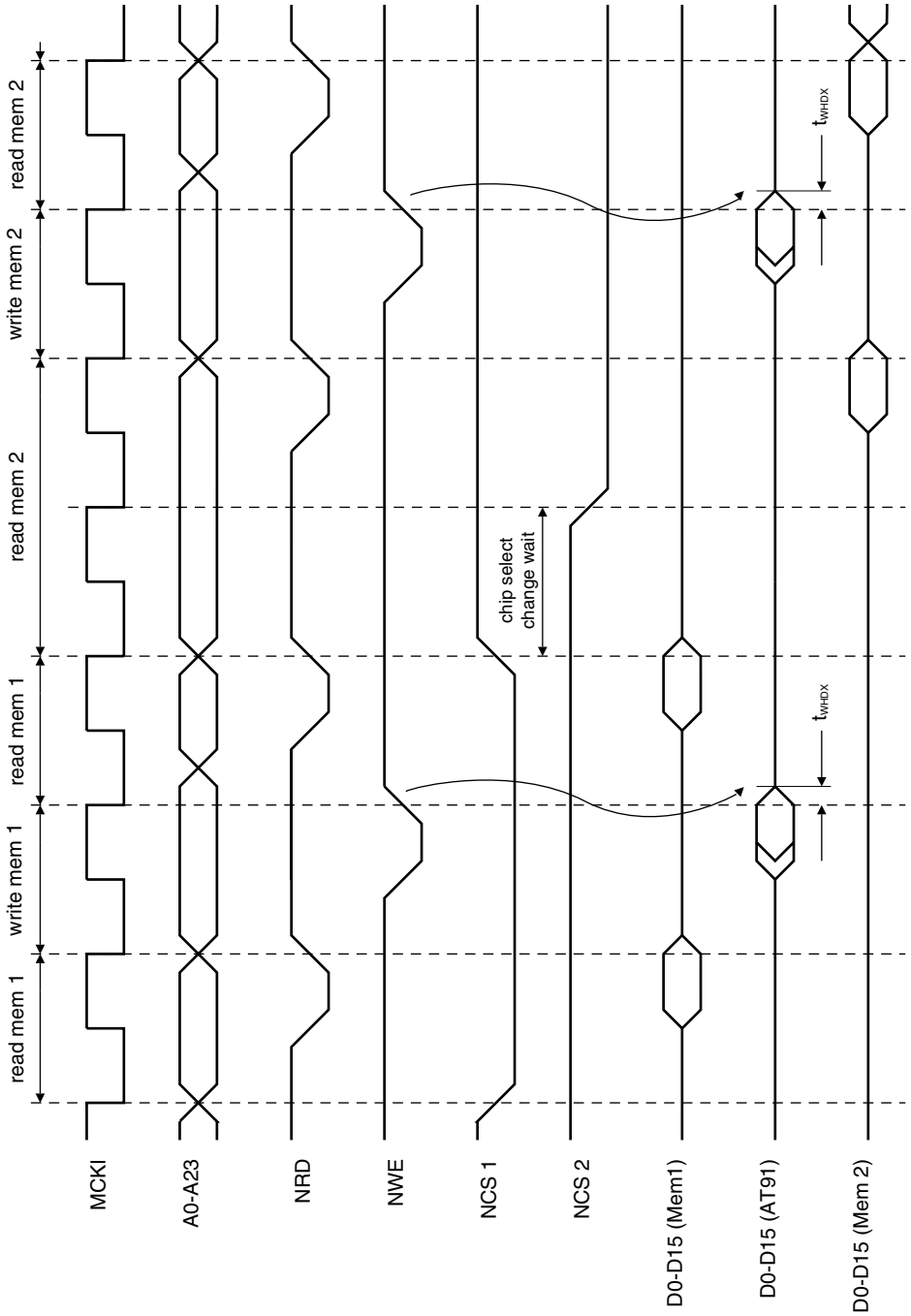


Figure 22. Early Read Protocol with No t_{DF}

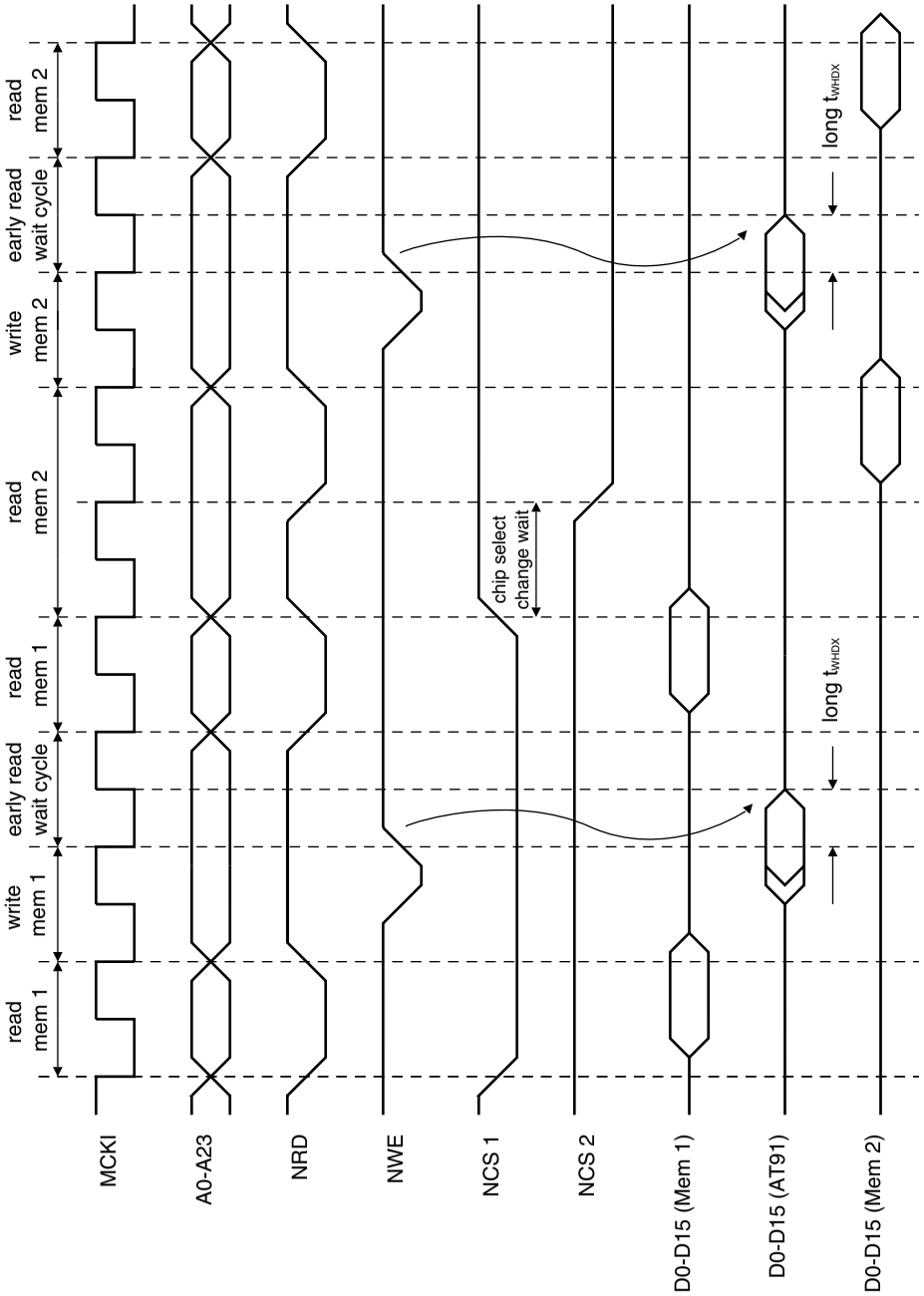


Figure 23. Standard Read Protocol with t_{DF}

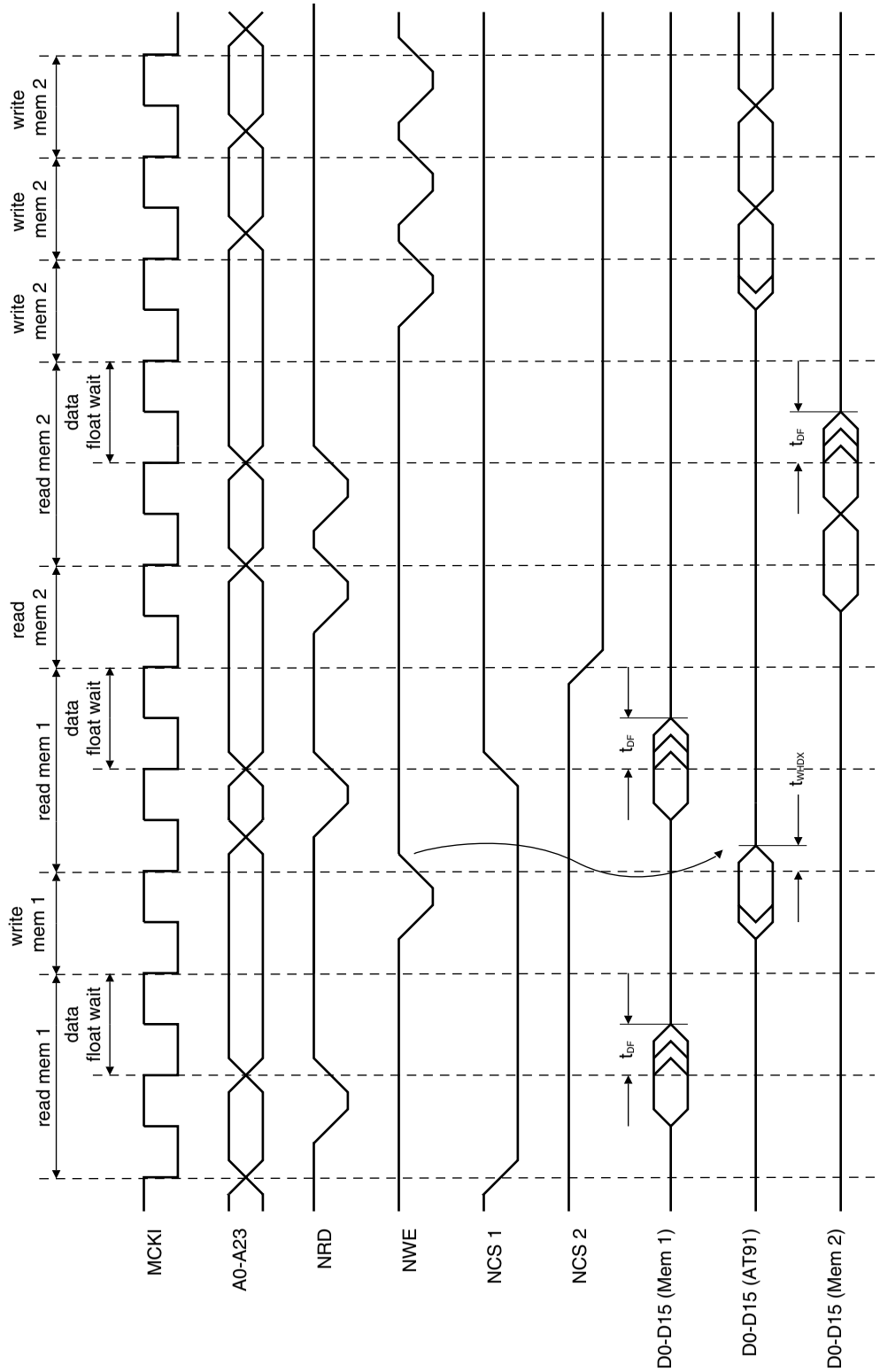
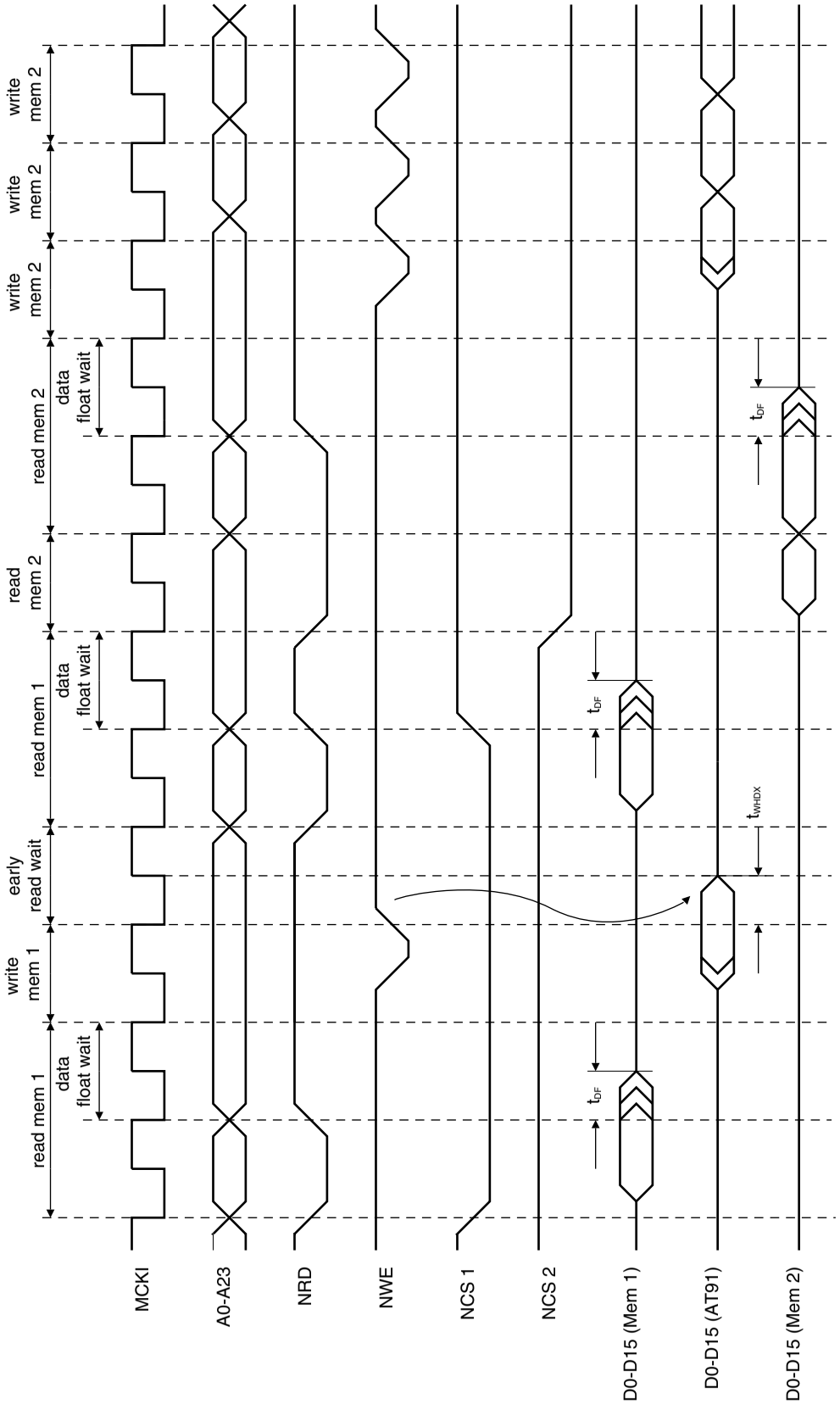


Figure 24. Early Read Protocol with t_{DF}



Figures 25 through 31 show the timing cycles and wait states for read and write access to the various AT91M63200 external memory devices. The configurations described are as follows:

Table 4. Memory Access Waveforms

Figure Number	Number of Wait States	Bus Width	Size of Data Transfer
25	0	16	Word
26	1	16	Word
27	1	16	Half-Word
28	0	8	Word
29	1	8	Half-Word
30	1	8	Byte
31	0	16	Byte

Figure 25. 0 Wait States, 16-bit Bus Width, Word Transfer

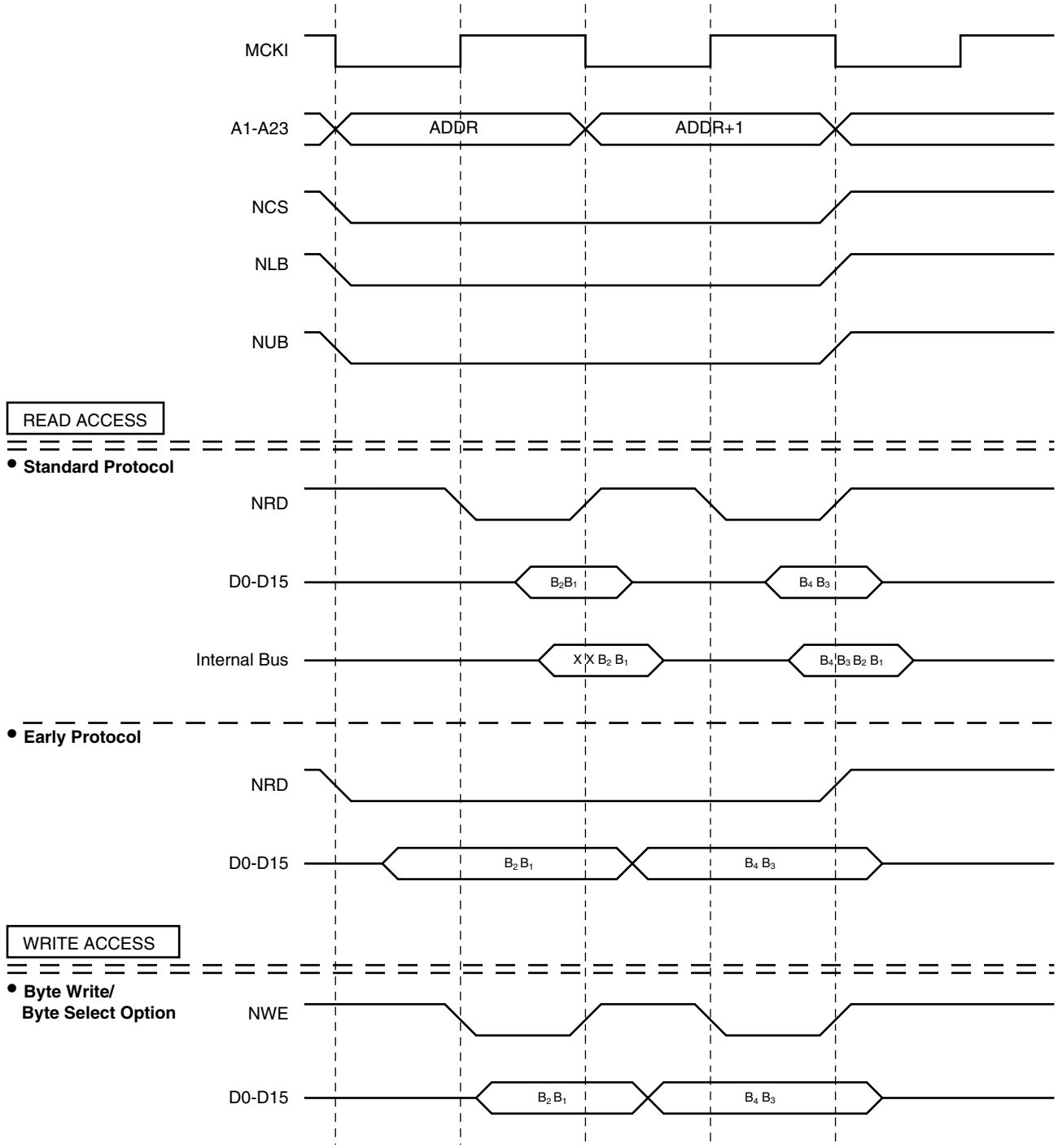


Figure 26. 1 Wait State, 16-bit Bus Width, Word Transfer

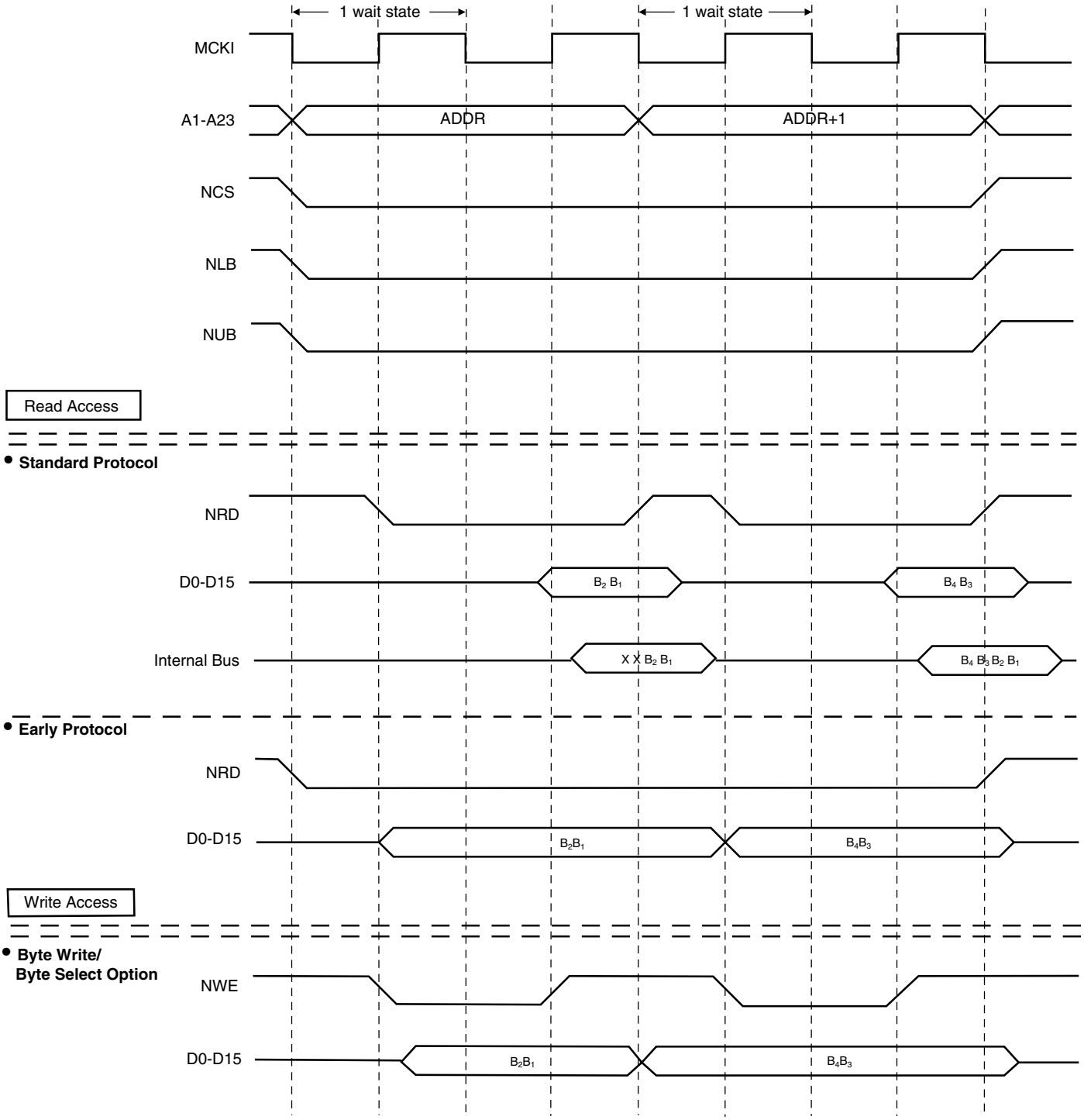


Figure 27. 1 Wait State, 16-bit Bus Width, Half-word Transfer

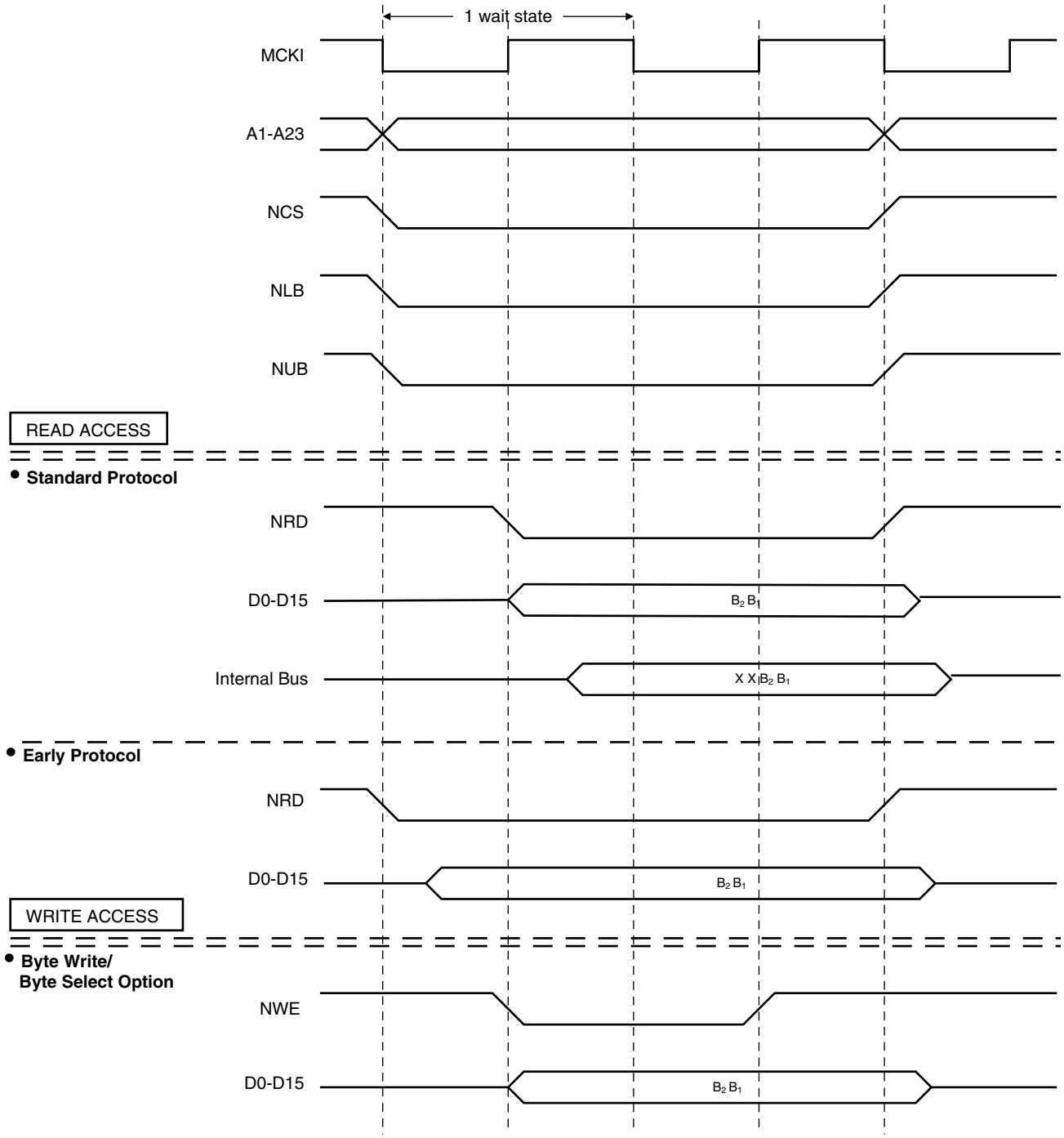


Figure 28. 0 Wait States, 8-bit Bus Width, Word Transfer

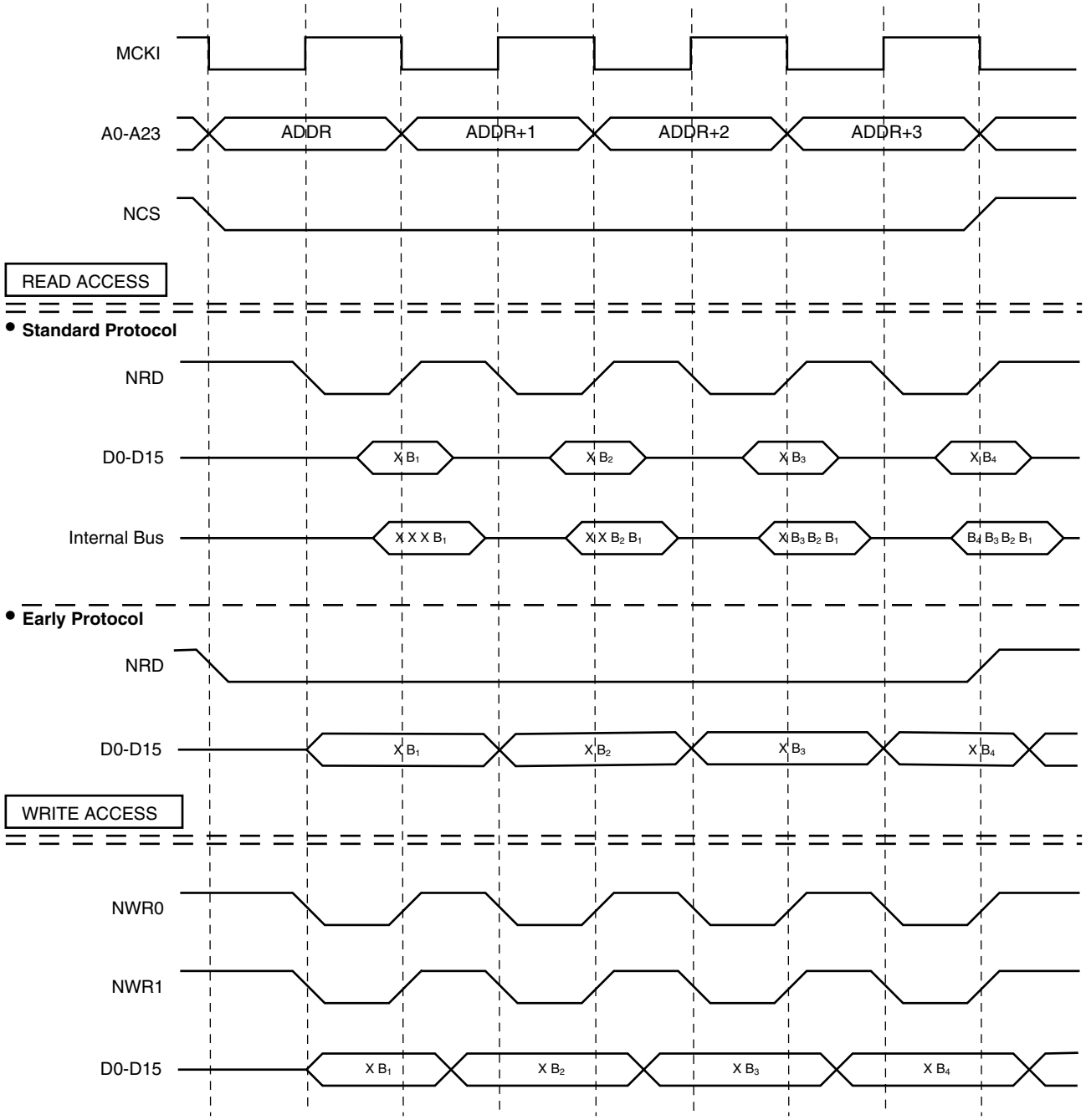


Figure 29. 1 Wait State, 8-bit Bus Width, Half-word Transfer

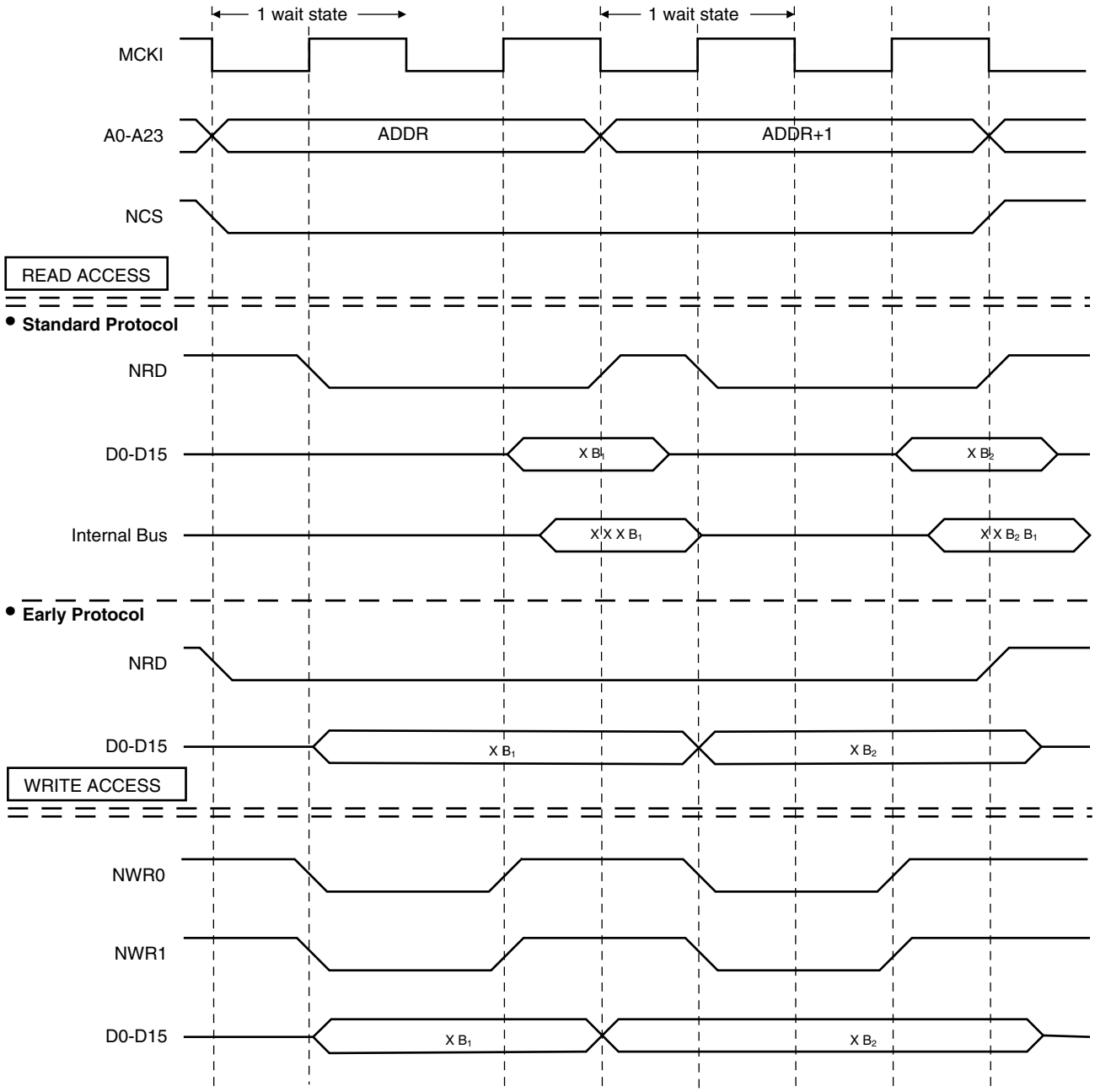


Figure 30. 1 Wait State, 8-bit Bus Width, Byte Transfer

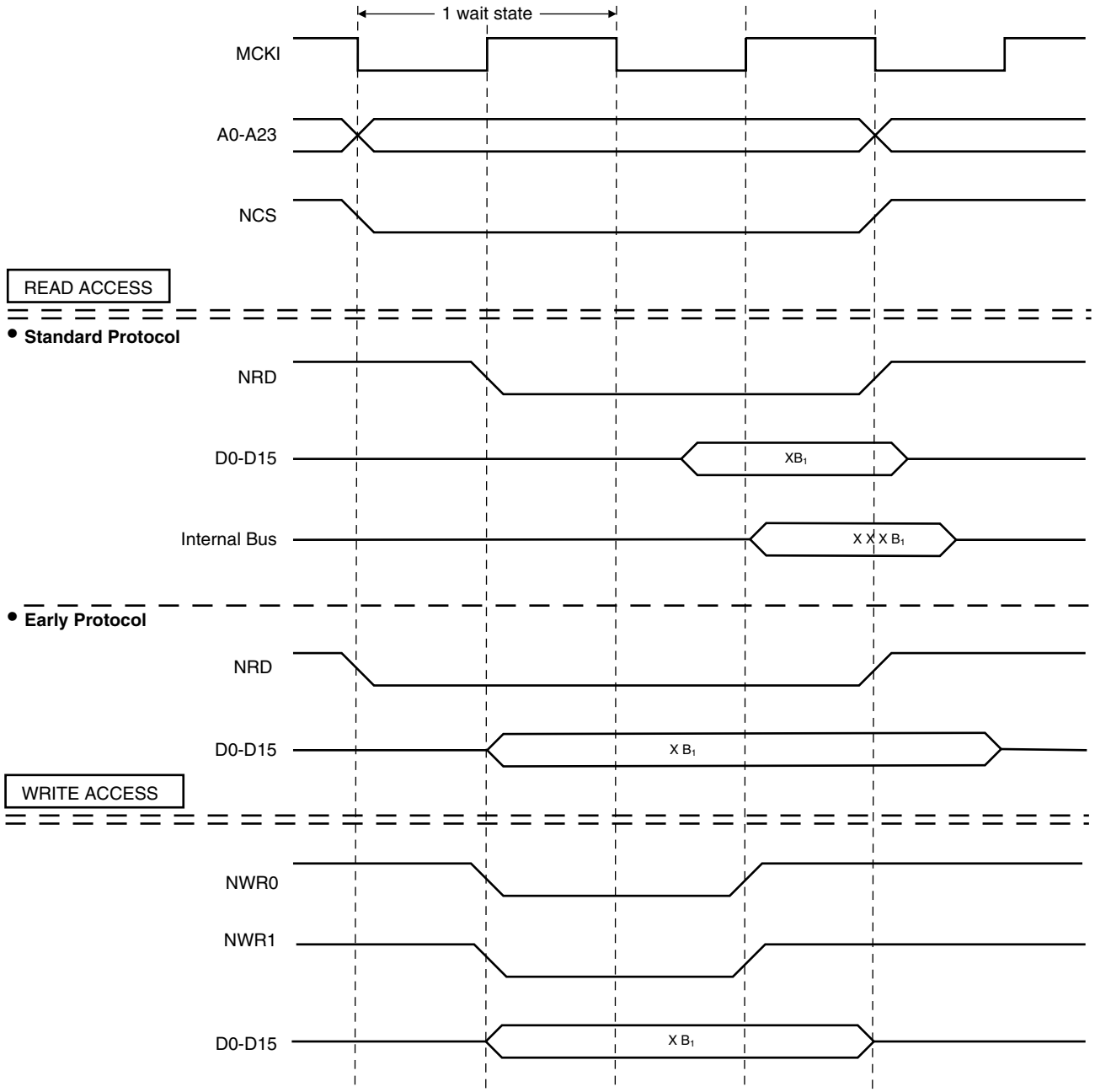
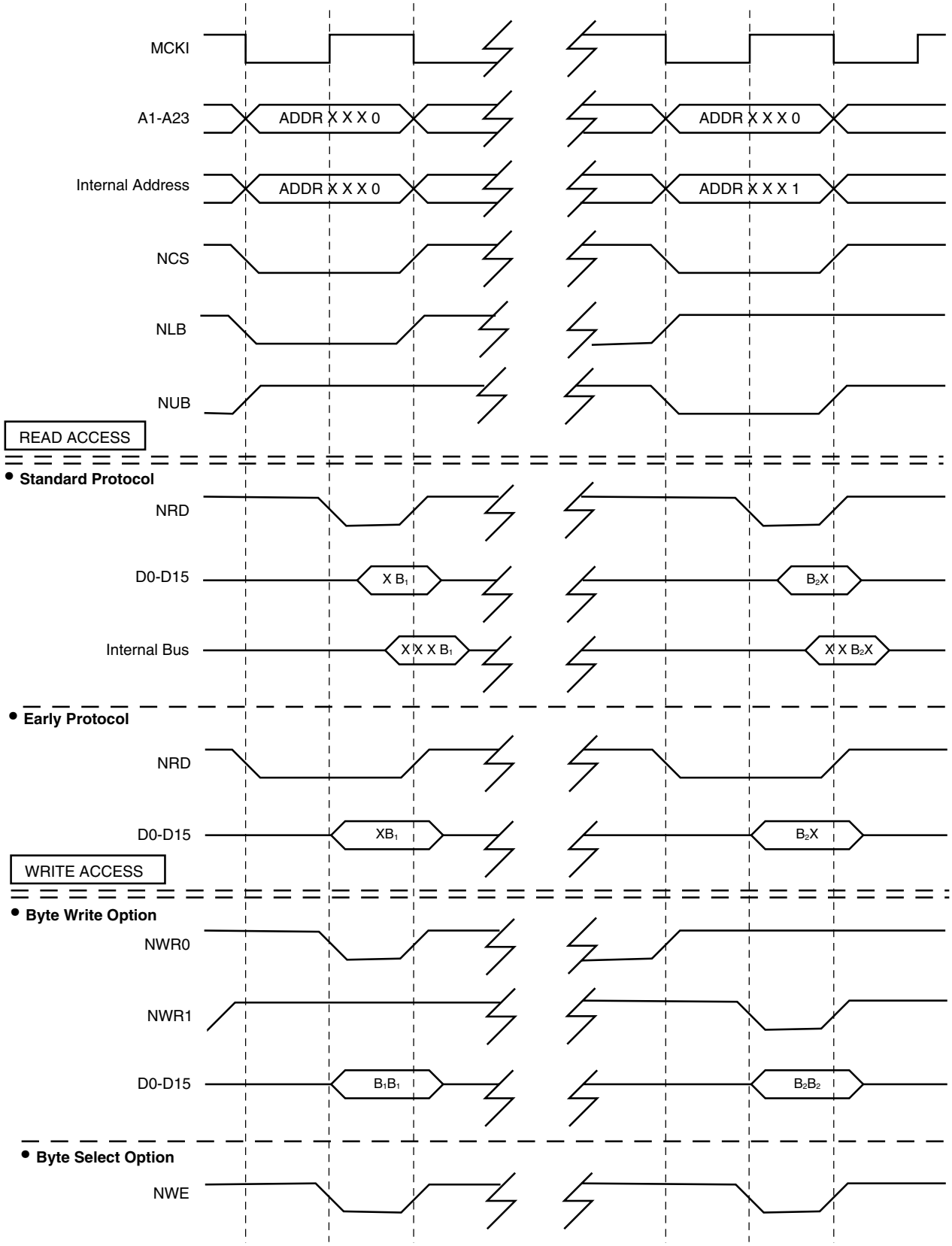


Figure 31. 0 Wait States, 16-bit Bus Width, Byte Transfer



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 page 14). The Memory Control Register (EBI_MCR) is used to program the number of active chip selects and data read protocol.

Base Address: 0xFFE00000

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.

Table 5. 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 5
Absolute Address: 0xFFE00000 - 0xFFE0001C

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

- **DBW: Data Bus Width**

DBW		Data Bus Width
0	0	Reserved
0	1	16-bit data bus width
1	0	8-bit data bus width
1	1	Reserved

- **NWS: Number of Wait States**
  This field is valid only if WSE is set.

NWS			Number of Standard Wait States
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

- **WSE: Wait State Enable**
  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
0	0	1M byte	12 bits (31-20)
0	1	4M bytes	10 bits (31-22)
1	0	16M bytes	8 bits (31-24)
1	1	64M bytes	6 bits (31-26)

- **TDF: Data Float Output Time**

TDF			Number of Cycles Added after the Transfer
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

- **BAT: Byte Access Type**
  0 = Byte write access type.
  1 = Byte select access type.

- **CSEN: Chip Select Enable**
  0 = Chip select is disabled.
  1 = Chip select is enabled.

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

EBI Remap Control Register

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

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

- **RCB: Remap Command Bit**
  0 = No effect.
  1 = Cancels the remapping (performed at reset) of the page zero memory devices.
  –

EBI Memory Control Register

Register Name: EBI_MCR–
Access Type: Read/Write
Reset Value: See Table 5
Absolute Address: 0xFFE00024

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

- **ALE: Address Line Enable**
  This field determines the number of valid address lines and the number of valid chip select lines.

ALE			Valid Address Bits	Maximum Addressable Space	Valid Chip Select
0	X	X	A20, A21, A22, A23	16M bytes	none
1	0	0	A20, A21, A22	8M bytes	CS4
1	0	1	A20, A21	4M bytes	CS4, CS5
1	1	0	A20	2M bytes	CS4, CS5, CS6
1	1	1	none	1M bytes	CS4, CS5, CS6, CS7

- **DRP: Data Read Protocol**
  0 = Standard read protocol for all external memory devices enabled.
  1 = Early read protocol for all external memory devices enabled.

MPI: Multi-processor Interface

The AT91M63200 family features a second bus interface which is dedicated to parallel data transfers with an external processing device. The MPI is based on a 1K byte Dual-port RAM (DPRAM) and an arbiter. Both the ARM core and the external processor can read and write to any location in the DPRAM.

In order to avoid conflicts when the ARM core or external processor is accessing the DPRAM, an arbiter is present. The external processor makes a bus request (MPI_BR) and waits until the bus grant (MPI_BG) is asserted before a read or write access to the DPRAM is made. The external bus request is synchronized on the main clock of the AT91M63200 microcontroller before being processed. The deactivation of the external bus grant is asynchronous and results from the deactivation of the external bus request. See Figure 35.

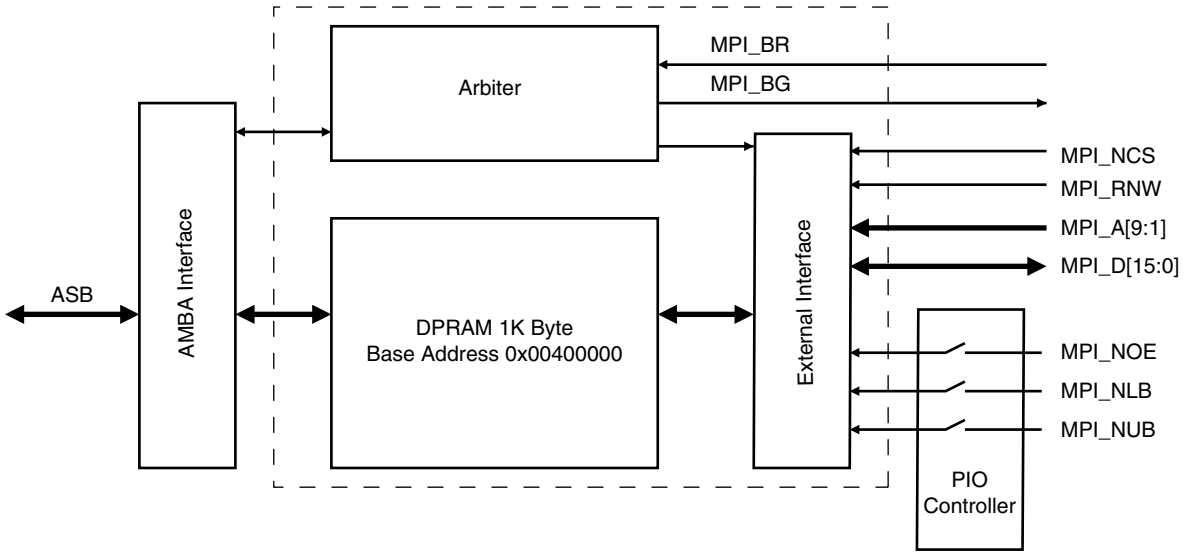
The arbiter always gives priority to the external processor over the ARM core. If the ARM core is accessing the DPRAM when an external bus request is made, the ARM core access is suspended and finished after the bus request has been removed. Care must be taken that the ARM core is not halted longer than is critical for the application.

The external processor accesses the DPRAM like a standard 16-bit SRAM once the bus grant is active. The control signals MPI_NOE, MPI_NLB and MPI_NUB are multiplexed, respectively, with the PIO signals PB0, PB1 and PB2. These signals are not mandatory for proper use of the MPI.

If one or more of these signals are not used, the PIO function must be selected on the respective pins. Consequently, the PIO controller will drive an active level on the MPI control signals. As an example, if all three control signals are not used, the external processor can only perform 16-bit accesses to the DPRAM and the MPI_NCS and MPI_RNW signals determine the data bus direction. Special care must be taken if the MPI_RNW is changed within the active period of the MPI_NCS. External bus conflicts may occur in this case.

The ARM core has single cycle 8-, 16-, and 32-bit access to the DPRAM.

Figure 32. MPI Block Diagram



Note: *For detailed timing values, see the corresponding datasheet "M63200 Electrical and Mechanical Characteristics" (Literature No. 1090).*

Note: *After a hardware reset, pins MPI_NOE, MPI_NLB and MPI_NUB are not enabled by default (see "PIO: Parallel I/O Controller" on page 55). The user must configure the PIO Controller to enable the corresponding pins for their MPI function.*

Pin Description

Pin Name	Mnemonic	Function	Type
MPI Bus Request	MPI_BR	Active high bus request input	Input
MPI Bus Grant	MPI_BG	Active high bus grant output	Output
MPI Chip Select	MPI_NCS	Active low chip select input	Input
MPI Read/Write	MPI_RNW	Active high read and active low write input	Input
MPI Output Enable	MPI_NOE	Active low output enable	Input
MPI Lower Byte Select	MPI_NLB	Active low lower byte select	Input
MPI Upper Byte Select	MPI_NUB	Active low upper byte select	Input
MPI Address Bus	MPI_A[9:1]	9-bit address bus	Input
MPI Data Bus	MPI_D[15:0]	16-bit data bus	I/O

Table 6. MPI Function Table

MPI_BG	MPI_NCS	MPI_NOE	MPI_RNW	MPI_NLB	MPI_NUB	MPI_D[7:0]	MPI_D[15:8]	Ref. cycle
L	X	X	X	X	X	High-Z	High-Z	–
H	H	X	X	X	X	High-Z	High-Z	–
H	L	H	X	X	X	High-Z	High-Z	–
H	L	L	H	L	L	Output	Output	Read cycle (16-bit)
H	L	L	H	L	H	Output	High-Z	Read cycle (lower byte)
H	L	L	H	H	L	High-Z	Output	Read cycle (upper byte)
H	L	L	H	H	H	High-Z	High-Z	–
H	L	X	L	L	L	Input	Input	Write cycle (16-bit)
H	L	X	L	L	H	Input	High-Z	Write cycle (lower byte)
H	L	X	L	H	L	High-Z	Input	Write cycle (upper byte)
H	L	X	L	H	H	High-Z	High-Z	–

Note: X: H or L

MPI Connection

The MPI must be connected to the bus of the external processor as a 1K byte 16-bit memory.

As illustrated in Figure 33, below, the MPI only supports 16-bit data transfers by connecting the basic signals: MPI_A9 to MPI_A1, MPI_D15 to MPI_D0, MPI_NCS and MPI_RNW.

Connection of the MPI_NOE signal is not mandatory, but gives the advantage of driving the bus only during the data access while the chip select line is active, in order to avoid any risk of contention on the data bus.

The connection of the MPI_NUB and MPI_NLB signals is not mandatory, but allows the external processor to perform byte accesses.

The connection with the interface with an external processor varies depending on the bus of the processor. Figure 34, below, shows how to connect the MPI to the External Bus Interface of an AT91 as an additional example.

Figure 33. MPI Connection to External Processor

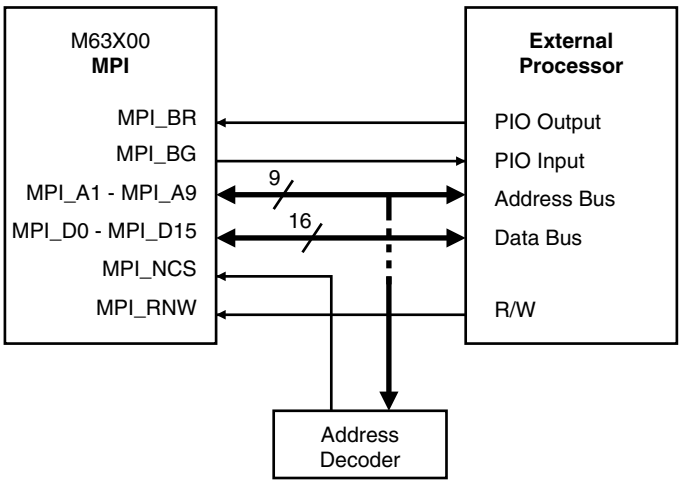
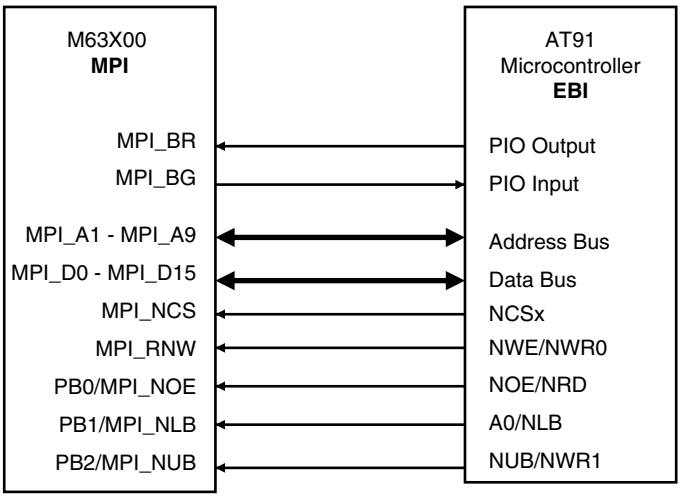


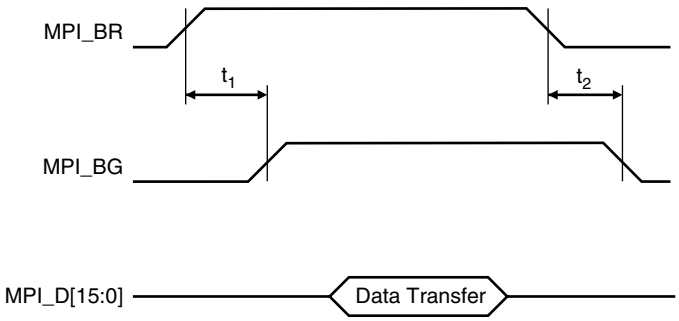
Figure 34. MPI Connection to an AT91 Microcontroller



MPI Arbitration

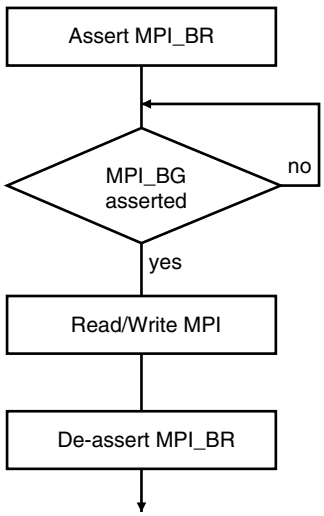
Figure 35 below shows the hardware protocol on the bus request and bus grant signals.

Figure 35. External Arbitration



The following diagram shows the actions which must be performed by the external processor to take control of the MPI.

Figure 36. MPI Control



The MPI guarantees a maximum delay to assert the bus grant signal. Any AT91M63200 instruction execution in progress (even Swap and Load/Store Multiple) is stopped. If the ARM core is not accessing the DPRAM, the delay t_1 is one MCKI cycle plus the propagation time. If the ARM core is accessing the DPRAM, the delay can be up to four MCKI cycles to allow the current instruction to be stopped cleanly (Swap or Load/Store Multiple instructions).

As the de-assertion of the bus grant signal is asynchronous, t_2 is in all cases less than one MCKI cycle.

Note that the read/write MPI sequence must be as short as possible in order to reduce to a minimum the risk of stopping the AT91 in case it needs to access the MPI, or to reduce the time during which it is stopped.

After having performed these actions, the external processor must inform the AT91M63200 that data has been read or written. This may be done by positioning an external interrupt signal NIRQ0-NIRQ3 or NFIQ, or by flagging. In the last case, a memory space in the MPI must be reserved for this flag and the AT91M63200 application software must poll it to detect an update by the external processor. The flag must be de-asserted after treatment by the AT91M63200 application software.

AIC: Advanced Interrupt Controller

The AT91M63200 has an 8-level priority, individually-maskable, vectored interrupt controller. This feature substantially reduces the software and real-time overhead in handling internal and external interrupts.

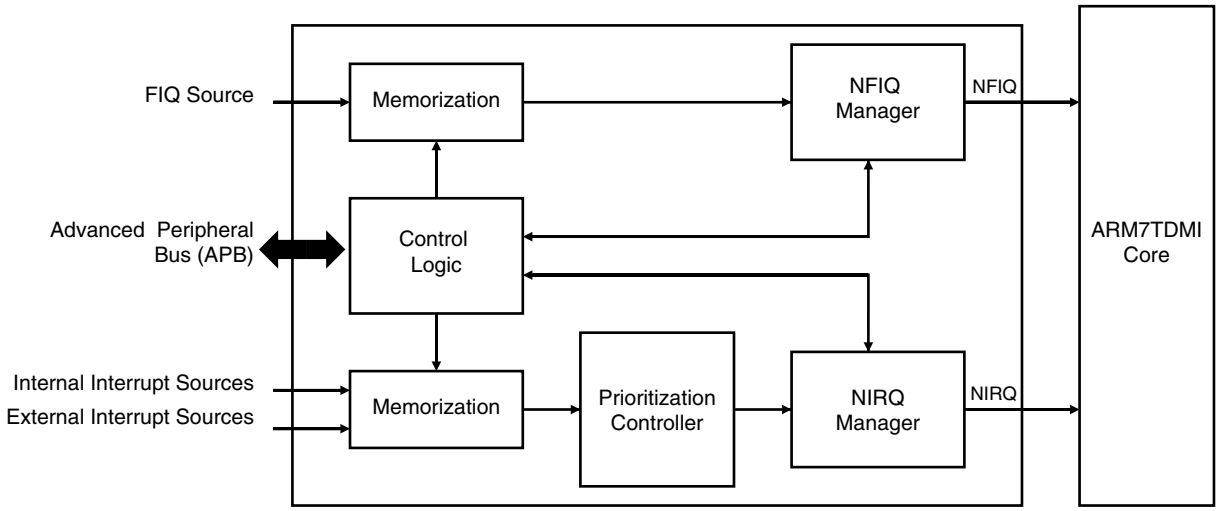
The interrupt controller is connected to the NFIQ (fast interrupt request) and the NIRQ (standard interrupt request) inputs of the ARM7TDMI processor. The processor's NFIQ line can only be asserted by the external fast interrupt request input: FIQ. The NIRQ line can be asserted by the interrupts generated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

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

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

The interrupt sources are listed in Table 7 and the AIC programmable registers in Table 8.

Figure 37. 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 7. AIC Interrupt Sources

Interrupt Source	Interrupt Name	Interrupt Description
0	FIQ	Fast interrupt
1	SWIRQ	Soft interrupt (generated by the AIC)
2	US0IRQ	USART Channel 0 interrupt
3	US1IRQ	USART Channel 1 interrupt
4	US2IRQ	USART Channel 2 interrupt
5	SPIRQ	SPI interrupt
6	TC0IRQ	Timer Channel 0 interrupt
7	TC1IRQ	Timer Channel 1 interrupt
8	TC2IRQ	Timer Channel 2 interrupt
9	TC3IRQ	Timer Channel 3 interrupt
10	TC4IRQ	Timer Channel 4 interrupt
11	TC5IRQ	Timer Channel 5 interrupt
12	WDIRQ	Watchdog interrupt
13	PIOAIRQ	Parallel I/O Controller A interrupt
14	PIOBIRQ	Parallel I/O Controller B interrupt
15	---	Reserved
16	---	Reserved
17	---	Reserved
18	---	Reserved
19	---	Reserved
20	---	Reserved
21	---	Reserved
22	---	Reserved
23	---	Reserved
24	---	Reserved
25	---	Reserved
26	---	Reserved
27	---	Reserved
28	IRQ3	External interrupt 3
29	IRQ2	External interrupt 2
30	IRQ1	External interrupt 1
31	IRQ0	External interrupt 0

Hardware Interrupt Vectoring

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

```
ldrPC,[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 7) 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.

```
ldrPC,[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 can be read. This has the following consequences in normal mode:

- If an enabled interrupt with a higher priority than the current one is pending, it will be stacked.
- If there is no enabled pending interrupt, the spurious vector will 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 145).

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 2 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. Note that software which 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

Table 8. 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: 1. The reset value of this register depends on the level of the external IRQ lines. All other sources are cleared at reset.

AIC Source Mode Register

Register Name: AIC_SMR0...AIC_SMR31
Access Type: Read/Write
Reset Value: 0

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

- **PRIOR: Priority Level**
  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-edge triggered or positive- or negative-level sensitive.
  The active level or edge is not programmable for the internal sources.

SRCTYPE		Internal Sources	External Sources
0	0	Level Sensitive	Low-Level Sensitive
0	1	Edge Triggered	Negative-Edge Triggered
1	0	Level Sensitive	High-Level Sensitive
1	1	Edge Triggered	Positive-Edge Triggered

AIC Source Vector Register

Register Name: AIC_SVR0...AIC_SVR31
Access Type: Read/Write
Reset Value: 0

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

- **VECTOR: Interrupt Handler Address**
  The user may store in these registers the addresses of the corresponding handler for each interrupt source.

AIC Interrupt Vector Register

Register Name: AIC_IVR
Access Type: Read only
Reset Value: 0

31	30	29	28	27	26	25	24
IRQV							

23	22	21	20	19	18	17	16
IRQV							

15	14	13	12	11	10	9	8
IRQV							

7	6	5	4	3	2	1	0
IRQV							

- **IRQV: Interrupt Vector Register**
  The 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

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

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**
  The Interrupt Status Register returns the current interrupt source number.

AIC Interrupt Pending Register

Register Name: AIC_IPR
Access Type: Read only
Reset Value: Undefined

31	30	29	28	27	26	25	24
IRQ0	IRQ1	IRQ2	IRQ3	---	---	---	---
23	22	21	20	19	18	17	16
---	---	---	---	---	---	---	---
15	14	13	12	11	10	9	8
---	PIOBIRQ	PIOAIRQ	WDIRQ	TC5IRQ	TC4IRQ	TC3IRQ	TC2IRQ
7	6	5	4	3	2	1	0
TC1IRQ	TC0IRQ	SPIRQ	US2IRQ	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

31	30	29	28	27	26	25	24
IRQ0	IRQ1	IRQ2	IRQ3	---	---	---	---
23	22	21	20	19	18	17	16
---	---	---	---	---	---	---	---
15	14	13	12	11	10	9	8
---	PIOBIRQ	PIOAIRQ	WDIRQ	TC5IRQ	TC4IRQ	TC3IRQ	TC2IRQ
7	6	5	4	3	2	1	0
TC1IRQ	TC0IRQ	SPIRQ	US2IRQ	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

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

- **NFIQ: NFIQ Status**
  0 = NFIQ line inactive.
  1 = NFIQ line active.

- **NIRQ: NIRQ Status**
  0 = NIRQ line inactive.
  1 = NIRQ line active.

AIC Interrupt Enable Command Register

Register Name: AIC_IECR
Access Type: Write only

31	30	29	28	27	26	25	24
IRQ0	IRQ1	IRQ2	IRQ3	---	---	---	---
23	22	21	20	19	18	17	16
---	---	---	---	---	---	---	---
15	14	13	12	11	10	9	8
---	PIOBIRQ	PIOAIRQ	WDIRQ	TC5IRQ	TC4IRQ	TC3IRQ	TC2IRQ
7	6	5	4	3	2	1	0
TC1IRQ	TC0IRQ	SPIRQ	US2IRQ	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

31	30	29	28	27	26	25	24
IRQ0	IRQ1	IRQ2	IRQ3	---	---	---	---
23	22	21	20	19	18	17	16
---	---	---	---	---	---	---	---
15	14	13	12	11	10	9	8
---	PIOBIRQ	PIOAIRQ	WDIRQ	TC5IRQ	TC4IRQ	TC3IRQ	TC2IRQ
7	6	5	4	3	2	1	0
TC1IRQ	TC0IRQ	SPIRQ	US2IRQ	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

31	30	29	28	27	26	25	24
IRQ0	IRQ1	IRQ2	IRQ3	---	---	---	---
23	22	21	20	19	18	17	16
---	---	---	---	---	---	---	---
15	14	13	12	11	10	9	8
---	PIOBIRQ	PIOAIRQ	WDIRQ	TC5IRQ	TC4IRQ	TC3IRQ	TC2IRQ
7	6	5	4	3	2	1	0
TC1IRQ	TC0IRQ	SPIRQ	US2IRQ	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

31	30	29	28	27	26	25	24
IRQ0	IRQ1	IRQ2	IRQ3	---	---	---	---
23	22	21	20	19	18	17	16
---	---	---	---	---	---	---	---
15	14	13	12	11	10	9	8
---	PIOBIRQ	PIOAIRQ	WDIRQ	TC5IRQ	TC4IRQ	TC3IRQ	TC2IRQ
7	6	5	4	3	2	1	0
TC1IRQ	TC0IRQ	SPIRQ	US2IRQ	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

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

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 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 has 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:
   - Sets the current interrupt to be the pending one with the highest priority. The current level is the priority level of the current interrupt.
   - De-asserts the NIRQ line on the processor (Even if vectoring is not used, AIC_IVR must be read in order to de-assert NIRQ.)
   - Automatically clears the interrupt, if it has been programmed to be edge-triggered.
   - Pushes the current level on to the stack.
   - Returns the value written in the AIC_SVR corresponding to the current interrupt.
4. The previous step has the effect of branching to the corresponding interrupt service routine. This should start by saving the Link Register(r14_irq) and 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 the 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 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 the 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 the effect of branching 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 the 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 AT91M63200 has 58 programmable I/O lines. 14 pins on the AT91M63200 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 (see Tables 9 and 10). These lines are controlled by two separate and identical PIO Controllers called PIOA and PIOB. Each PIO controller also provides an internal interrupt signal to the Advanced Interrupt Controller.

Note: *After a hardware reset, the PIO clock is disabled by default (see "Power Management Controller" on page 139). The user must configure the Power Management Controller before any access to the user interface of the PIO.*

Multiplexed I/O Lines

Some I/O lines are multiplexed with an I/O signal of a peripheral. After reset, the pin is controlled by the PIO Controller and is in input mode.

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 38 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 an effect only if the pin is configured to be controlled by the PIO Controller.

I/O Levels

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

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

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

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

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

Filters

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

Interrupts

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

When PIO_ISR is read, the register is automatically cleared.

User Interface

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

Multi-driver (Open Drain)

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

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

Figure 38. Parallel I/O Multiplexed with a Bi-directional Signal

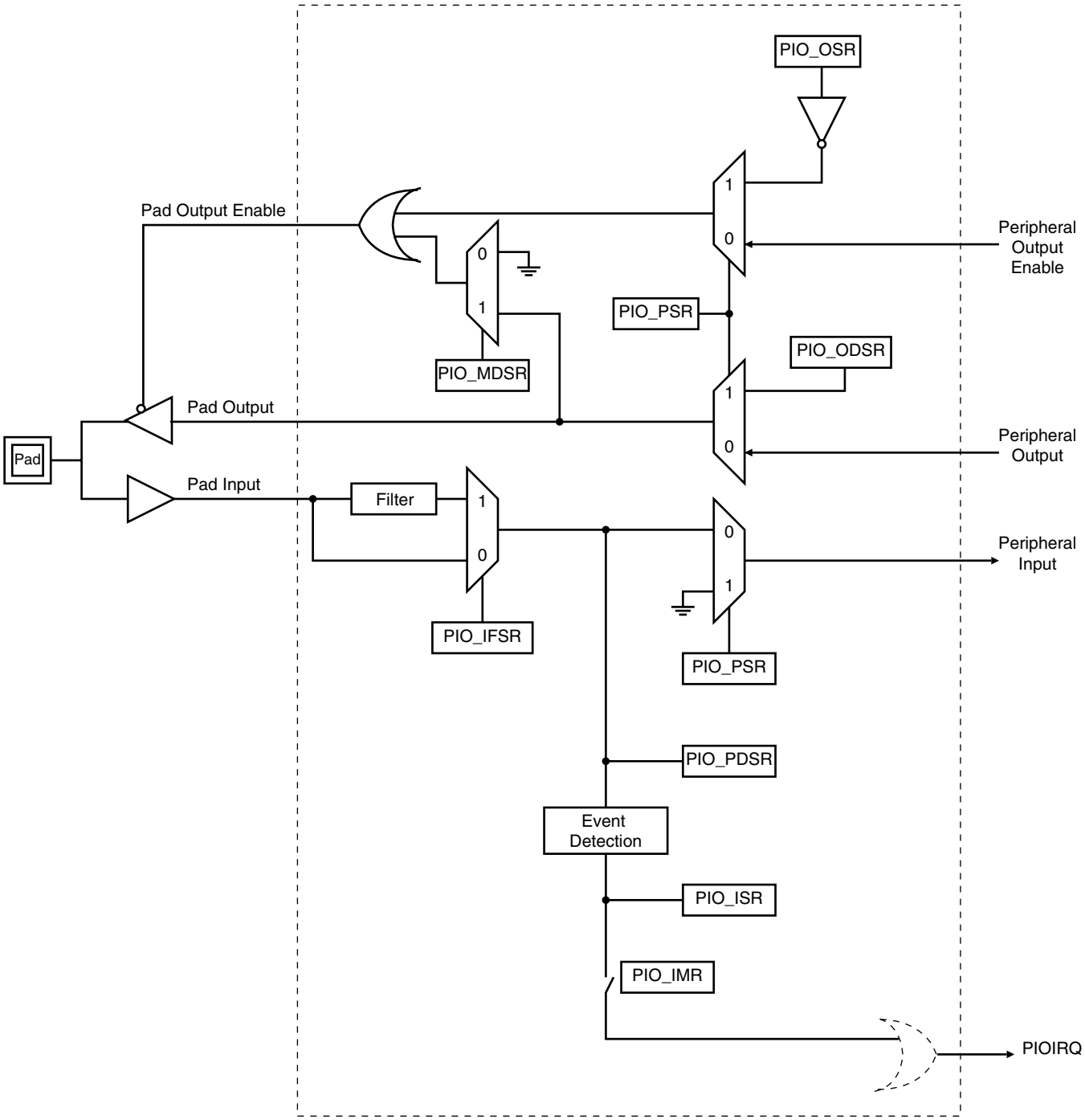


Table 9. PIO Controller A Connection Table

PIO Controller		Peripheral				
Bit Number[1]	Port Name	Port Name	Signal Description	Signal Direction	Reset State	Pin Number
0	PA0	TCLK3	Timer 3 Clock Signal	Input	PIO Input	66
1	PA1	TIOA3	Timer 3 Signal A	Bi-directional	PIO Input	67
2	PA2	TIOB3	Timer 3 Signal B	Bi-directional	PIO Input	68
3	PA3	TCLK4	Timer 4 Clock Signal	Input	PIO Input	69
4	PA4	TIOA4	Timer 4 Signal A	Bi-directional	PIO Input	70
5	PA5	TIOB4	Timer 4 Signal B	Bi-directional	PIO Input	71
6	PA6	TCLK5	Timer 5 Clock Signal	Input	PIO Input	72
7	PA7	TIOA5	Timer 5 Signal A	Bi-directional	PIO Input	75
8	PA8	TIOB5	Timer 5 Signal B	Bi-directional	PIO Input	76
9	PA9	IRQ0	External Interrupt 0	Input	PIO Input	77
10	PA10	IRQ1	External Interrupt 1	Input	PIO Input	78
11	PA11	IRQ2	External Interrupt 2	Input	PIO Input	79
12	PA12	IRQ3	External Interrupt 3	Input	PIO Input	80
13	PA13	FIQ	Fast Interrupt	Input	PIO Input	81
14	PA14	SCK0	USART 0 Clock Signal	Bi-directional	PIO Input	82
15	PA15	TXD0	USART 0 Transmit Data Signal	Output	PIO Input	83
16	PA16	RXD0	USART 0 Receive Data Signal	Input	PIO Input	84
17	PA17	SCK1	USART 1 Clock Signal	Bi-directional	PIO Input	85
18	PA18	TXD1	USART 1 Transmit Data Signal	Output	PIO Input	86
19	PA19	RXD1	USART 1 Receive Data Signal	Input	PIO Input	91
20	PA20	SCK2	USART 2 Clock Signal	Bi-directional	PIO Input	92
21	PA21	TXD2	USART 2 Transmit Data Signal	Output	PIO Input	93
22	PA22	RXD2	USART 2 Receive Data Signal	Input	PIO Input	94
23	PA23	SPCK	SPI Clock Signal	Bi-directional	PIO Input	95
24	PA24	MISO	SPI Master In Slave Out	Bi-directional	PIO Input	96
25	PA25	MOSI	SPI Master Out Slave In	Bi-directional	PIO Input	97
26	PA26	NPCS0	SPI Peripheral Chip Select 0	Bi-directional	PIO Input	98
27	PA27	NPCS1	SPI Peripheral Chip Select 1	Output	PIO Input	99
28	PA28	NPCS2	SPI Peripheral Chip Select 2	Output	PIO Input	100
29	PA29	NPCS3	SPI Peripheral Chip Select 3	Output	PIO Input	101
30	–	–	–	–	–	–
31	–	–	–	–	–	–

Note: 1. Bit number refers to the data bit which corresponds to this signal in each of the User Interface registers

Table 10. PIO Controller B Connection Table

PIO Controller		Peripheral				
Bit Number[1]	Port Name	Port Name	Signal Description	Signal Direction	Reset State	Pin Number
0	PB0	MPI_NOE	MPI Output Enable	Input	PIO Input	139
1	PB1	MPI_NLB	MPI Lower Byte Select	Input	PIO Input	140
2	PB2	MPI_NUB	MPI Upper Byte Select	Input	PIO Input	141
3	PB3	–	–	–	PIO Input	142
4	PB4	–	–	–	PIO Input	143
5	PB5	–	–	–	PIO Input	144
6	PB6	–	–	–	PIO Input	145
7	PB7	–	–	–	PIO Input	146
8	PB8	–	–	–	PIO Input	149
9	PB9	–	–	–	PIO Input	150
10	PB10	–	–	–	PIO Input	151
11	PB11	–	–	–	PIO Input	152
12	PB12	–	–	–	PIO Input	153
13	PB13	–	–	–	PIO Input	154
14	PB14	–	–	–	PIO Input	155
15	PB15	–	–	–	PIO Input	156
16	PB16	–	–	–	PIO Input	157
17	PB17	MCKO	Master Clock Output	Output	PIO Input	158
18	PB18	BMS	Boot Mode Select	Input	PIO Input	163
19	PB19	TCLK0	Timer 0 Clock Signal	Input	PIO Input	55
20	PB20	TIOA0	Timer 0 Signal A	Bi-directional	PIO Input	56
21	PB21	TIOB0	Timer 0 Signal B	Bi-directional	PIO Input	57
22	PB22	TCLK1	Timer 1 Clock Signal	Input	PIO Input	58
23	PB23	TIOA1	Timer 1 Signal A	Bi-directional	PIO Input	61
24	PB24	TIOB1	Timer 1 Signal B	Bi-directional	PIO Input	62
25	PB25	TCLK2	Timer 2 Clock Signal	Input	PIO Input	63
26	PB26	TIOA2	Timer 2 Signal A	Bi-directional	PIO Input	64
27	PB27	TIOB2	Timer 2 Signal B	Bi-directional	PIO Input	65
28	–	–	–	–	–	–
29	–	–	–	–	–	–
30	–	–	–	–	–	–
31	–	–	–	–	–	–

Note: 1. Bit number refers to the data bit which corresponds to this signal in each of the User Interface registers

PIO User Interface

PIO Controller A Base Address: 0xFFFEC000
PIO Controller B Base Address: 0xFFFF0000

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	0x3FFFFFFF (A) 0x0FFFFFFF (B)
0x0C	Reserved	–	–	–
0x10	Output Enable Register	PIO_OER	Write only	–
0x14	Output Disable Register	PIO_ODR	Write only	–
0x18	Output Status Register	PIO_OSR	Read only	0
0x1C	Reserved	–	–	–
0x20	Input Filter Enable Register	PIO_IFER	Write only	–
0x24	Input Filter Disable Register	PIO_IFDR	Write only	–
0x28	Input Filter Status Register	PIO_IFSR	Read only	0
0x2C	Reserved	–	–	–
0x30	Set Output Data Register	PIO_SODR	Write only	–
0x34	Clear Output Data Register	PIO_CODR	Write only	–
0x38	Output Data Status Register	PIO_ODSR	Read only	0
0x3C	Pin Data Status Register	PIO_PDSR	Read only	(see Note 1)
0x40	Interrupt Enable Register	PIO_IER	Write only	–
0x44	Interrupt Disable Register	PIO_IDR	Write only	–
0x48	Interrupt Mask Register	PIO_IMR	Read only	0
0x4C	Interrupt Status Register	PIO_ISR	Read only	(see Note 2)
0x50	Multi-driver Enable Register	PIO_MDER	Write only	–
0x54	Multi-driver Disable Register	PIO_MDDR	Write only	–
0x58	Multi-driver Status Register	PIO_MDSR	Read only	0
0x5C	Reserved	–	–	–

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

PIO Enable Register

Register Name: PIO_PER
Access Type: Write only

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

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

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

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: 0x3FFFFFFF (A)
0x0FFFFFFF (B)

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

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

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

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

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

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

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

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

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

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

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

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: Undefined

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

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

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

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 an 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

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 an 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

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

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

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

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

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

PIO Multi-drive Enable Register

Register Name: PIO_MDER
Access Type: Write only

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

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

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

PIO Multi-drive Disable Register

Register Name: PIO_MDDR
Access Type: Write only

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

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

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

PIO Multi-drive Status Register

Register Name: PIO_MDSR
Access Type: Read only

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

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

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

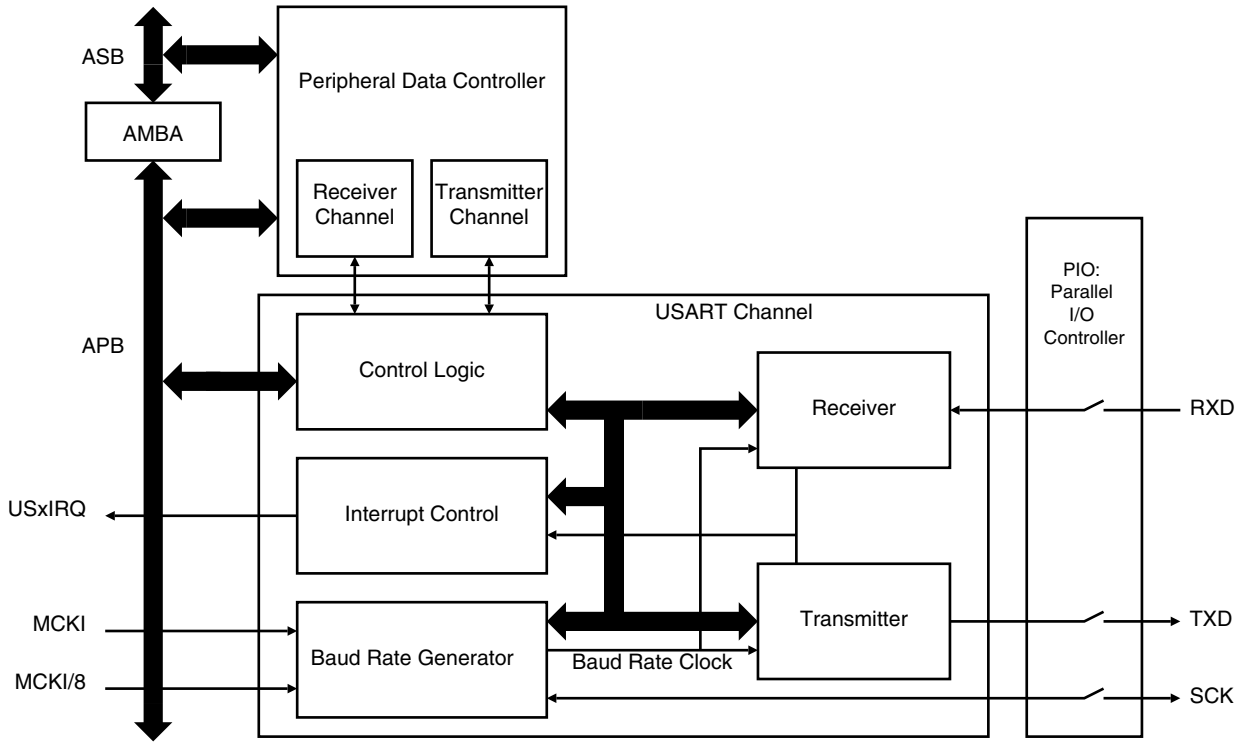
USART: Universal Synchronous/Asynchronous Receiver/Transmitter

The AT91M63200 provides three 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 39. 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

Note: *After a hardware reset, the USART clock is disabled by default (see "PMC: Power Management Controller" on page 139). The user must configure the Power Management Controller before any access to the user interface of the USART.*

Note: *After a hardware reset, the USART pins are deselected by default (see "PIO: Parallel I/O Controller" on page 55). The user must configure the PIO Controller before enabling the transmitter or receiver.*

If the user selects one of the internal clocks, SCK can be configured as a PIO.

Baud Rate Generator

The Baud Rate Generator provides the bit period clock (the baud rate clock) to both the receiver and the transmitter.

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

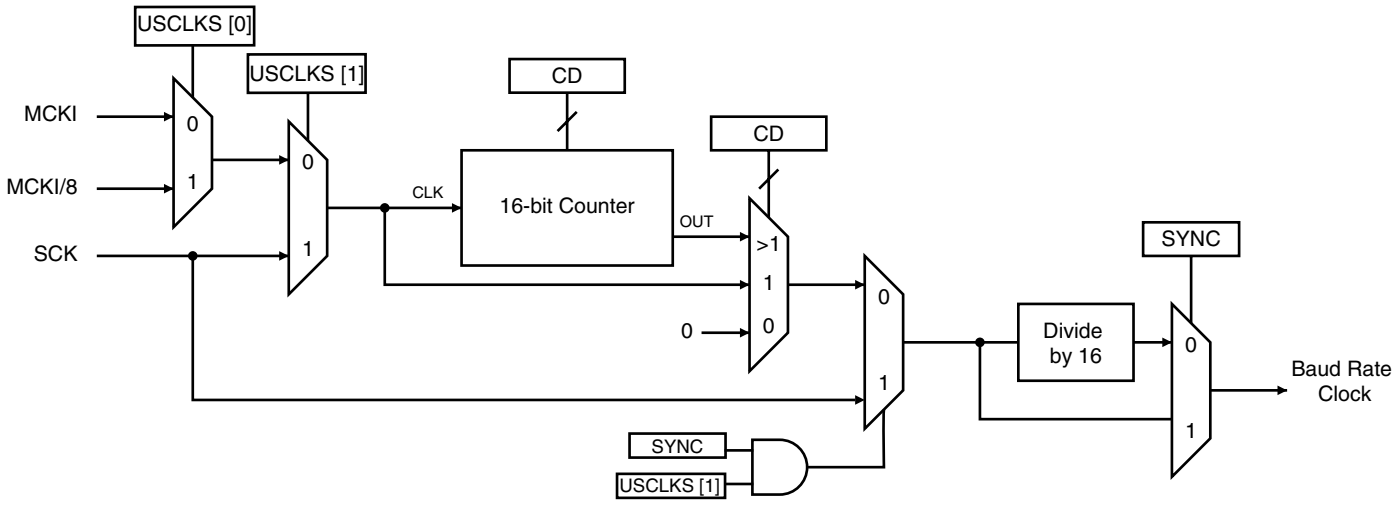
$$\text{Baud Rate} = \frac{\text{Selected Clock}}{16 \times \text{CD}}$$

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 40. 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 (1-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 41. Asynchronous Mode: Start Bit Detection

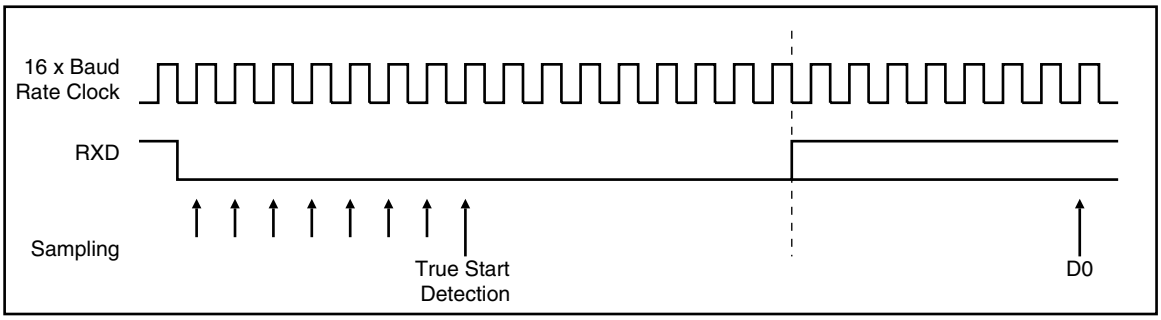
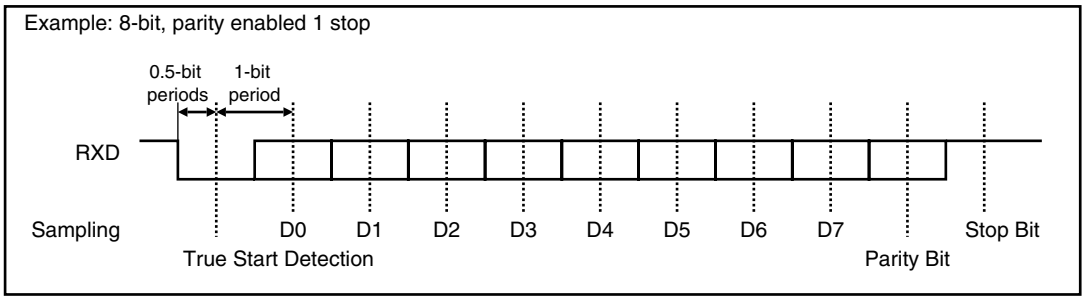


Figure 42. Asynchronous Mode: Character Reception



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 the example in Figure 43.

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.

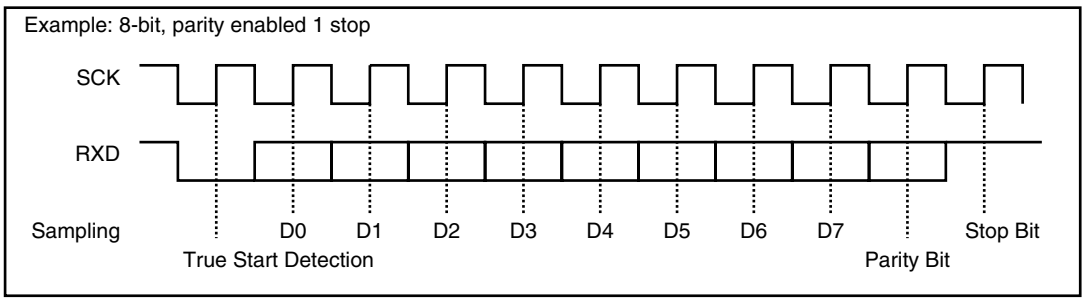
Time-out

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:

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

Figure 43. Synchronous Mode: Character Reception



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 the example in Figure 44.

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.

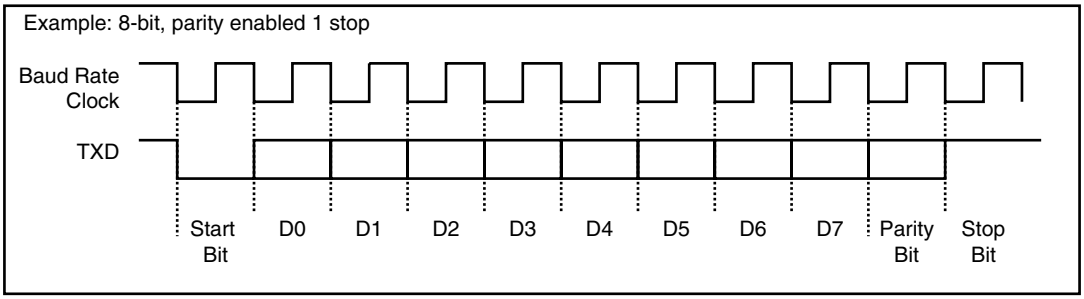
$$\text{Idle state duration between two characters} = \text{Time-guard value} \times \text{Bit period}$$

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 44. Synchronous and Asynchronous Modes: Character Transmission



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). US_CSR.TXEMPTY is cleared. The break blocks the transmitter shift register until it is completed (high level for at least 12 bit periods after the STPBRK command is requested).

In order to avoid unpredictable states:

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

The standard break transmission sequence is:

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

The next byte can then be sent:

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

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

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

Receive Break

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

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

Peripheral Data Controller

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

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) and can be programmed to generate an interrupt. Transfers are then disabled until a new non-zero counter value is programmed.

Interrupt Generation

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

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

Channel Modes

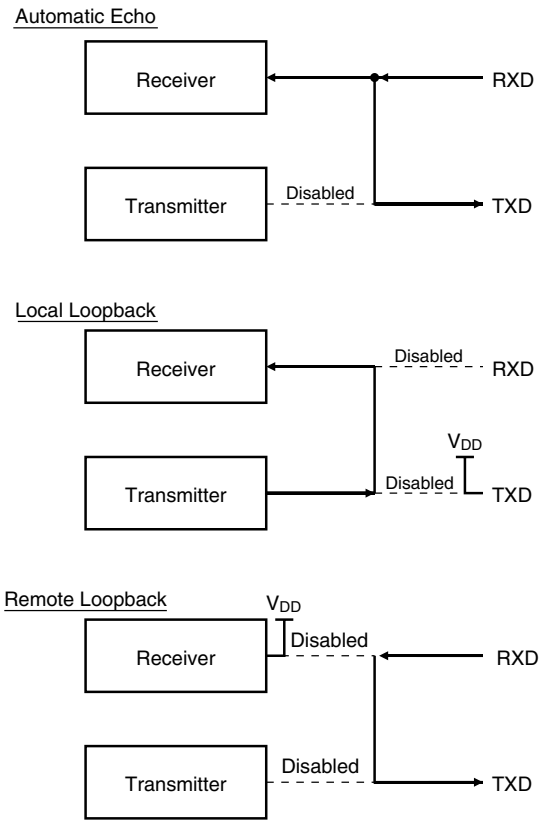
The USART can be programmed to operate in three different test modes, using the field CHMODE in US_MR.

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

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

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

Figure 45. Channel Modes



USART User Interface

Base Address USART0: 0xFFFC0000
Base Address USART1: 0xFFFC4000
Base Address USART2: 0xFFFC8000

Table 1. USART Memory Map

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

USART Control Register

Name: US_CR
Access Type: Write only

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**
  0 = No effect.
  1 = The receiver logic is reset.
- **RSTTX: Reset Transmitter**
  0 = No effect.
  1 = The transmitter logic is reset.
- **RXEN: Receiver Enable**
  0 = No effect.
  1 = The receiver is enabled if RXDIS is 0.
- **RXDIS: Receiver Disable**
  0 = No effect.
  1 = The receiver is disabled.
- **TXEN: Transmitter Enable**
  0 = No effect.
  1 = The transmitter is enabled if TXDIS is 0.
- **TXDIS: Transmitter Disable**
  0 = No effect.
  1 = The transmitter is disabled.
- **RSTSTA: Reset Status Bits**
  0 = No effect.
  1 = Resets the status bits PARE, FRAME, OVRE and RXBRK in the US_CSR.
- **STTBRK: Start Break**
  0 = No effect.
  1 = 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**
  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**
  0 = No effect.
  1 = Start waiting for a character before clocking the time-out counter.
- **SENDA: Send Address**
  0 = No effect.
  1 = In multi-drop mode only, the next character written to the US_THR is sent with the address bit set.

USART Mode Register

Name: US_MR
Access Type: Read/Write

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
0	0	MCKI
0	1	MCKI/8
1	X	External (SCK)

- **CHRL: Character Length**

CHRL		Character Length
0	0	Five bits
0	1	Six bits
1	0	Seven bits
1	1	Eight bits

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

- **SYNC: Synchronous Mode Select**
  0 = USART operates in asynchronous mode.
  1 = USART operates in synchronous mode.
- **PAR: Parity Type**

PAR			Parity Type
0	0	0	Even parity
0	0	1	Odd parity
0	1	0	Parity forced to 0 (space)
0	1	1	Parity forced to 1 (mark)
1	0	x	No parity
1	1	x	Multi-drop mode

- **NBSTOP: Number of Stop Bits**
  The interpretation of the number of stop bits depends on SYNC.

NBSTOP		Asynchronous (SYNC = 0)	Synchronous (SYNC = 1)
0	0	1 stop bit	1 stop bit
0	1	1.5 stop bits	Reserved
1	0	2 stop bits	2 stop bits
1	1	Reserved	Reserved

- **CHMODE: Channel Mode**

CHMODE		Mode Description
0	0	Normal Mode The USART Channel operates as an Rx/Tx USART.
0	1	Automatic Echo Receiver Data Input is connected to TXD pin.
1	0	Local Loopback Transmitter Output Signal is connected to Receiver Input Signal.
1	1	Remote Loopback RXD pin is internally connected to TXD pin.

- **MODE9: 9-bit Character Length**
  0 = CHRL defines character length.
  1 = 9-bit character length.
- **CKLO: Clock Output Select**
  0 = The USART does not drive the SCK pin.
  1 = The USART drives the SCK pin if USCLKS[1] is 0.

USART Interrupt Enable Register

Name: US_IER
Access Type: Write only

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

- **RXRDY: Enable RXRDY Interrupt**
  0 = No effect.
  1 = Enables RXRDY Interrupt.
- **TXRDY: Enable TXRDY Interrupt**
  0 = No effect.
  1 = Enables TXRDY Interrupt.
- **RXBRK: Enable Receiver Break Interrupt**
  0 = No effect.
  1 = Enables Receiver Break Interrupt.
- **ENDRX: Enable End of Receive Transfer Interrupt**
  0 = No effect.
  1 = Enables End of Receive Transfer Interrupt.
- **ENDTX: Enable End of Transmit Transfer Interrupt**
  0 = No effect.
  1 = Enables End of Transmit Transfer Interrupt.
- **OVRE: Enable Overrun Error Interrupt**
  0 = No effect.
  1 = Enables Overrun Error Interrupt.
- **FRAME: Enable Framing Error Interrupt**
  0 = No effect.
  1 = Enables Framing Error Interrupt.
- **PARE: Enable Parity Error Interrupt**
  0 = No effect.
  1 = Enables Parity Error Interrupt.
- **TIMEOUT: Enable Time-out Interrupt**
  0 = No effect.
  1 = Enables Reception Time-out Interrupt.
- **TXEMPTY: Enable TXEMPTY Interrupt**
  0 = No effect.
  1 = Enables TXEMPTY Interrupt.
- **COMMTX: Enable ARM7TDMI ICE Debug Communication Channel Transmit Interrupt**
  This bit is implemented for USART0 only.
  0 = No effect.
  1 = Enables COMMTX Interrupt.
- **COMMRX: Enable ARM7TDMI ICE Debug Communication Channel Receive Interrupt**
  This bit is implemented for USART0 only.
  0 = No effect.
  1 = Enables COMMRX Interrupt.

USART Interrupt Disable Register

Name: US_IDR
Access Type: Write only

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

- **RXRDY: Disable RXRDY Interrupt**
  0 = No effect.
  1 = Disables RXRDY Interrupt.
- **TXRDY: Disable TXRDY Interrupt**
  0 = No effect.
  1 = Disables TXRDY Interrupt.
- **RXBRK: Disable Receiver Break Interrupt**
  0 = No effect.
  1 = Disables Receiver Break Interrupt.
- **ENDRX: Disable End of Receive Transfer Interrupt**
  0 = No effect.
  1 = Disables End of Receive Transfer Interrupt.
- **ENDTX: Disable End of Transmit Transfer Interrupt**
  0 = No effect.
  1 = Disables End of Transmit Transfer Interrupt.
- **OVRE: Disable Overrun Error Interrupt**
  0 = No effect.
  1 = Disables Overrun Error Interrupt.
- **FRAME: Disable Framing Error Interrupt**
  0 = No effect.
  1 = Disables Framing Error Interrupt.
- **PARE: Disable Parity Error Interrupt**
  0 = No effect.
  1 = Disables Parity Error Interrupt.
- **TIMEOUT: Disable Time-out Interrupt**
  0 = No effect.
  1 = Disables Receiver Time-out Interrupt.
- **TXEMPTY: Disable TXEMPTY Interrupt**
  0 = No effect.
  1 = Disables TXEMPTY Interrupt.
- **COMMTX: Disable ARM7TDMI ICE Debug Communication Channel Transmit Interrupt**
  This bit is implemented for USART0 only.
  0 = No effect.
  1 = Disables COMMTX Interrupt.
- **COMMRX: Disable ARM7TDMI ICE Debug Communication Channel Receive Interrupt**
  This bit is implemented for USART0 only.
  0 = No effect.
  1 = Disables COMMRX Interrupt.

USART Interrupt Mask Register

Name: US_IMR
Access Type: Read only

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

- **RXRDY: RXRDY Interrupt Mask**
  0 = RXRDY Interrupt is disabled.
  1 = RXRDY Interrupt is enabled.
- **TXRDY: TXRDY Interrupt Mask**
  0 = TXRDY Interrupt is disabled.
  1 = TXRDY Interrupt is enabled.
- **RXBRK: Receiver Break Interrupt Mask**
  0 = Receiver Break Interrupt is disabled.
  1 = Receiver Break Interrupt is enabled.
- **ENDRX: End of Receive Transfer Interrupt Mask**
  0 = End of Receive Transfer Interrupt is disabled.
  1 = End of Receive Transfer Interrupt is enabled.
- **ENDTX: End of Transmit Transfer Interrupt Mask**
  0 = End of Transmit Transfer Interrupt is disabled.
  1 = End of Transmit Transfer Interrupt is enabled.
- **OVRE: Overrun Error Interrupt Mask**
  0 = Overrun Error Interrupt is disabled.
  1 = Overrun Error Interrupt is enabled.
- **FRAME: Framing Error Interrupt Mask**
  0 = Framing Error Interrupt is disabled.
  1 = Framing Error Interrupt is enabled.
- **PARE: Parity Error Interrupt Mask**
  0 = Parity Error Interrupt is disabled.
  1 = Parity Error Interrupt is enabled.
- **TIMEOUT: Time-out Interrupt Mask**
  0 = Receive Time-out Interrupt is disabled.
  1 = Receive Time-out Interrupt is enabled.
- **TXEMPTY: TXEMPTY Interrupt Mask**
  0 = TXEMPTY Interrupt is disabled.
  1 = TXEMPTY Interrupt is enabled.
- **COMMTX: ARM7TDMI ICE Debug Communication Channel Transmit Interrupt Mask**
  This bit is implemented for USART0 only.
  0 = COMMTX Interrupt is disabled.
  1 = COMMTX Interrupt is enabled.
- **COMMRX: ARM7TDMI ICE Debug Communication Channel Receive Interrupt Mask**
  This bit is implemented for USART0 only.
  0 = COMMRX Interrupt is disabled.
  1 = COMMRX Interrupt is enabled.

USART Channel Status Register

Name: US_CSR

Access Type: Read only

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

- **RXRDY: Receiver Ready**
  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**
  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**
  0 = No Break Received nor End of Break detected since the last “Reset Status Bits” command in the Control Register.
  1 = Break Received or End of Break detected since the last “Reset Status Bits” command in the Control Register.
- **ENDRX: End of Receive Transfer**
  0 = The End of Transfer signal from the Peripheral Data Controller channel dedicated to the receiver is inactive.
  1 = The End of Transfer signal from the Peripheral Data Controller channel dedicated to the receiver is active.
- **ENDTX: End of Transmit Transfer**
  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**
  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**
  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**
  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**
  0 = There has not been a time-out since the last “Start Time-out” command or the Time-out Register is 0.
  1 = There has been a time-out since the last “Start Time-out” command.
- **TXEMPTY: Transmitter Empty**
  0 = There are characters in either US_THR or the Transmit Shift Register or 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.
- **COMMTX: ARM7TDMI ICE Debug Communication Channel Transmit Status**
  For USART0 only. Refer to the ARM7TDMI datasheet for a complete description of this flag.
- **COMMRX: ARM7TDMI ICE Debug Communication Channel Receive Status**
  For USART0 only. Refer to the ARM7TDMI datasheet for a complete description of this flag.

USART Receiver Holding Register

Name: US_RHR
Access Type: Read only

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

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

USART Transmitter Holding Register

Name: US_THR
Access Type: Write only

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

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

USART Baud Rate Generator Register

Name: US_BRGR
Access Type: Read/Write

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

- **CD: Clock Divisor**
  This register has no effect if synchronous mode is selected with an external clock.

CD	
0	Disables clock
1	Clock Divisor bypass
2 to 65535	Baud Rate (asynchronous mode) = Selected clock / (16 x CD) Baud Rate (synchronous mode) = Selected clock / CD

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

USART Receiver Time-out Register

Name: US_RTOR
Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
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

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

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

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

- **RXCTR: Receive Counter**
  RXCTR must be loaded with the size of the receive buffer.
  0: Stop peripheral data transfer dedicated to the receiver.
  1-65535: Start peripheral data transfer if RXRDY is active.

USART Transmit Pointer Register

Name: US_TPR
Access Type: Read/Write

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

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.

SPI: Serial Peripheral Interface

The AT91M63200 includes an SPI which provides communication with external devices in master or slave mode.

Pin Description

Seven pins are associated with the SPI interface. When not needed for the SPI function, each of these pins can be configured as a PIO.

Support for an external master is provided by the PIO Controller multi-driver option. To configure an SPI pin as open-drain to support external drivers, set the corresponding bits in the PIO_MDSR register (see page 70).

An input filter can be enabled on the SPI input pins by setting the corresponding bits in the PIO_IFSR (see page 64).

The NPCS0/NSS pin can function as a peripheral chip select output or slave select input. Refer to Table 12 for a description of the SPI pins.

Figure 46. SPI Block Diagram

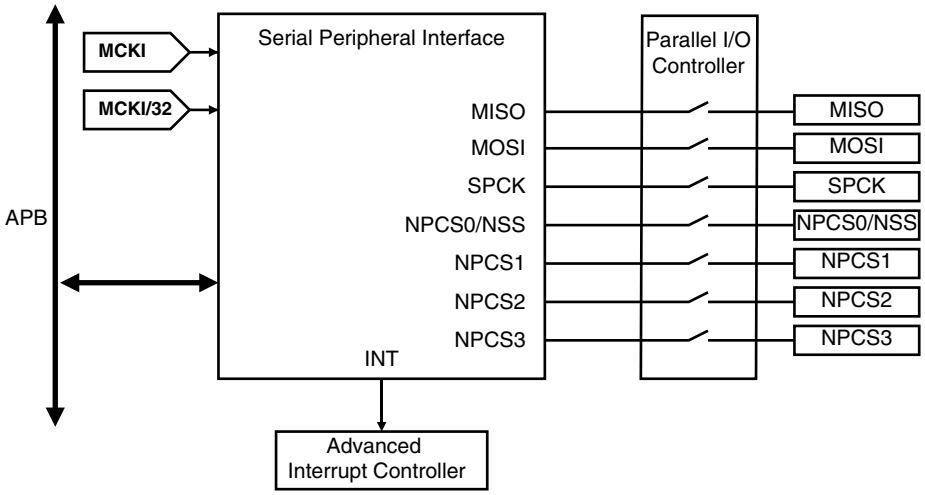


Table 12. SPI Pins

Pin Name	Mnemonic	Mode	Function
Master In Slave Out	MISO	Master Slave	Serial data input to SPI Serial data output from SPI
Master Out Slave In	MOSI	Master Slave	Serial data output from SPI Serial data input to SPI
Serial Clock	SPCK	Master Slave	Clock output from SPI Clock input to SPI
Peripheral Chip Selects	NPCS[3:1]	Master	Select peripherals
Peripheral Chip Select/ Slave Select	NPCS0/ NSS	Master Master Slave	Output: selects peripheral Input: low causes mode fault Input: chip select for SPI

Note: *After a hardware reset, the SPI clock is disabled by default (see "PMC: Power Management Controller" on page 139). The user must configure the Power Management Controller before any access to the user interface of the SPI.*

Note: *After a hardware reset, the SPI pins are deselected by default (see "PIO: Parallel I/O Controller" on page 55). The user must configure the PIO Controller to enable the corresponding pins for their SPI function. NPCS0/NSS must be configured as open-drain in the Parallel I/O Controller for multi-master operation.*

Master Mode

In master mode, the SPI controls data transfers to and from the slave(s) connected to the SPI bus. The SPI drives the chip select(s) to the slave(s) and the serial clock (SPCK). After enabling the SPI, a data transfer begins when the ARM core writes to the SP_TDR (Transmit Data Register). See Table 13.

Transmit and receive buffers maintain the data flow at a constant rate with a reduced requirement for high-priority interrupt servicing. When new data is available in the SP_TDR (Transmit Data Register) the SPI continues to transfer data. If the SP_RDR (Receive Data Register) has not been read before new data is received, the Overrun Error (OVRES) flag is set.

The delay between the activation of the chip select and the start of the data transfer (DLYBS), as well as the delay between each data transfer (DLYBCT), can be programmed for each of the four external chip selects. All data transfer characteristics including the two timing values are programmed in registers SP_CSR0 to SP_CSR3 (Chip Select Registers). See Table 13.

In master mode, the peripheral selection can be defined in two different ways:

- Fixed Peripheral Select: SPI exchanges data with only one peripheral
- Variable Peripheral Select: Data can be exchanged with more than one peripheral

Figures 47 and 48 show the operation of the SPI in master mode. For details concerning the flag and control bits in these diagrams, see the tables in the “Programmer’s Model”, starting on page 99.

Fixed Peripheral Select

This mode is ideal for transferring memory blocks without the extra overhead in the transmit data register to determine the peripheral.

Fixed Peripheral Select is activated by setting bit PS to zero in SP_MR (Mode Register). The peripheral is defined by the PCS field, also in SP_MR.

This option is only available when the SPI is programmed in master mode.

Variable Peripheral Select

Variable Peripheral Select is activated by setting bit PS to one. The PCS field in SP_TDR (Transmit Data Register) is used to select the destination peripheral. The data transfer characteristics are changed when the selected peripheral changes, according to the associated chip select register.

The PCS field in the SP_MR has no effect.

This option is only available when the SPI is programmed in master mode.

Chip Selects

The chip select lines are driven by the SPI only if it is programmed in master mode. These lines are used to select the destination peripheral. The PCSDEC field in SP_MR (Mode Register) selects 1 to 4 peripherals (PCSDEC = 0) or up to 15 peripherals (PCSDEC = 1).

If Variable Peripheral Select is active, the chip select signals are defined for each transfer in the PCS field in SP_TDR. Chip select signals can thus be defined independently for each transfer.

If Fixed Peripheral Select is active, chip select signals are defined for all transfers by the field PCS in SP_MR. If a transfer with a new peripheral is necessary, the software must wait until the current transfer is completed, then change the value of PCS in SP_MR before writing new data in SP_TDR.

The value on the NPCS pins at the end of each transfer can be read in the SP_RDR (Receive Data Register).

By default, all NPCS signals are high (equal to one) before and after each transfer.

Mode Fault Detection

A mode fault is detected when the SPI is programmed in master mode and a low level is driven by an external master on the NPCS0/NSS signal.

When a mode fault is detected, the MODF bit in the SP_SR is set until the SP_SR is read and the SPI is disabled until re-enabled by bit SPIEN in the SP_CR (Control Register).

Figure 47. Functional Flow Diagram in Master Mode

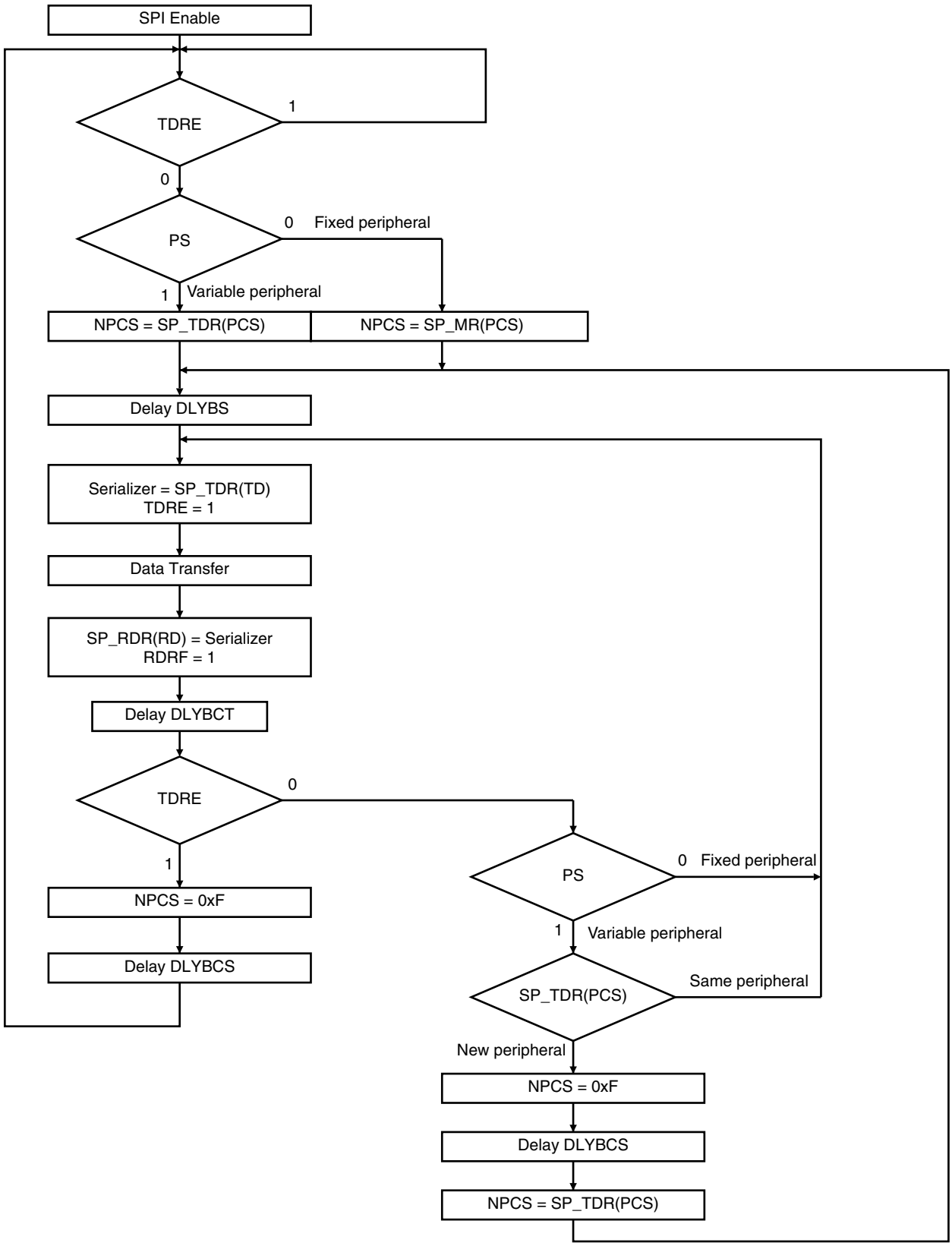
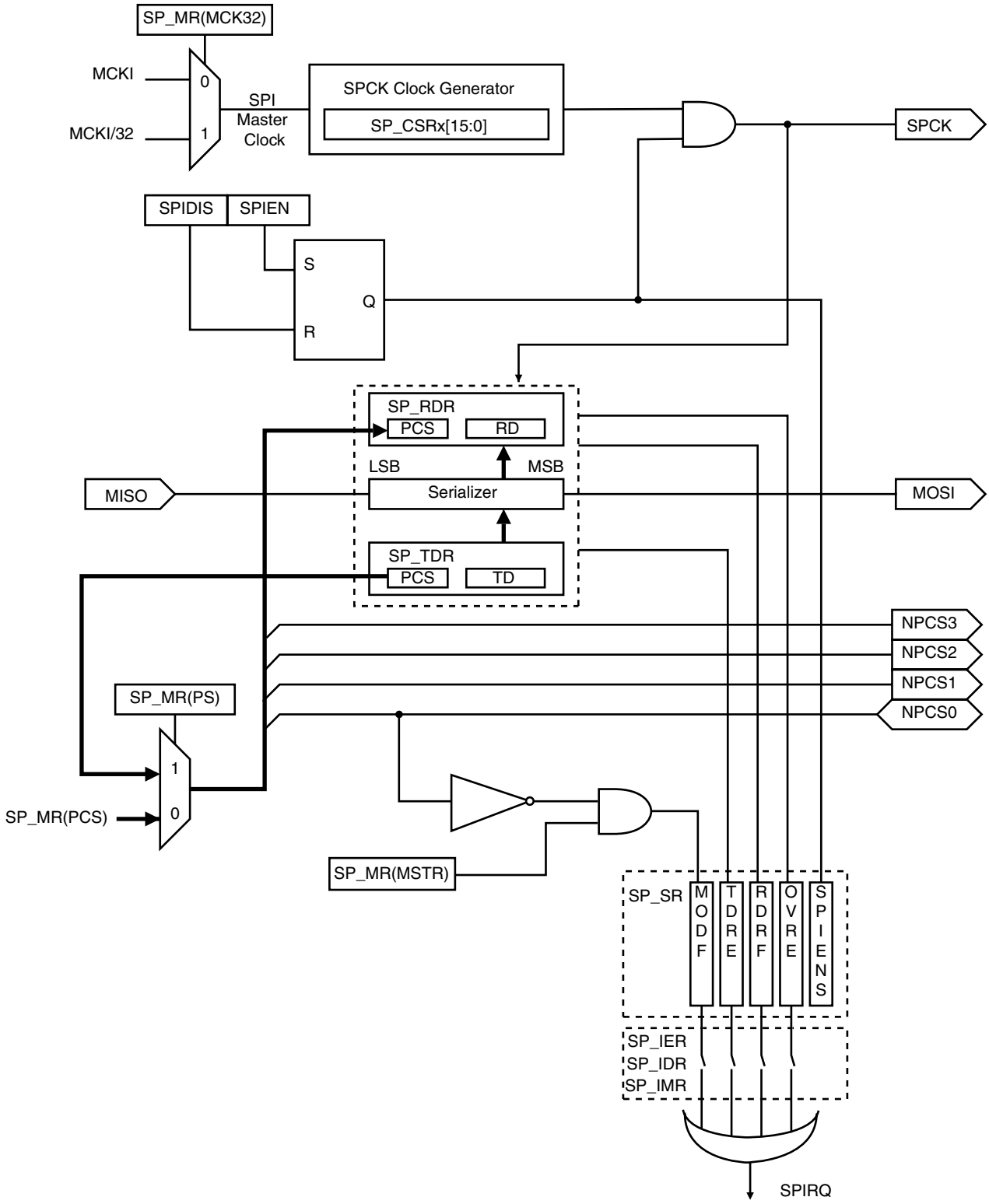


Figure 48. SPI in Master Mode

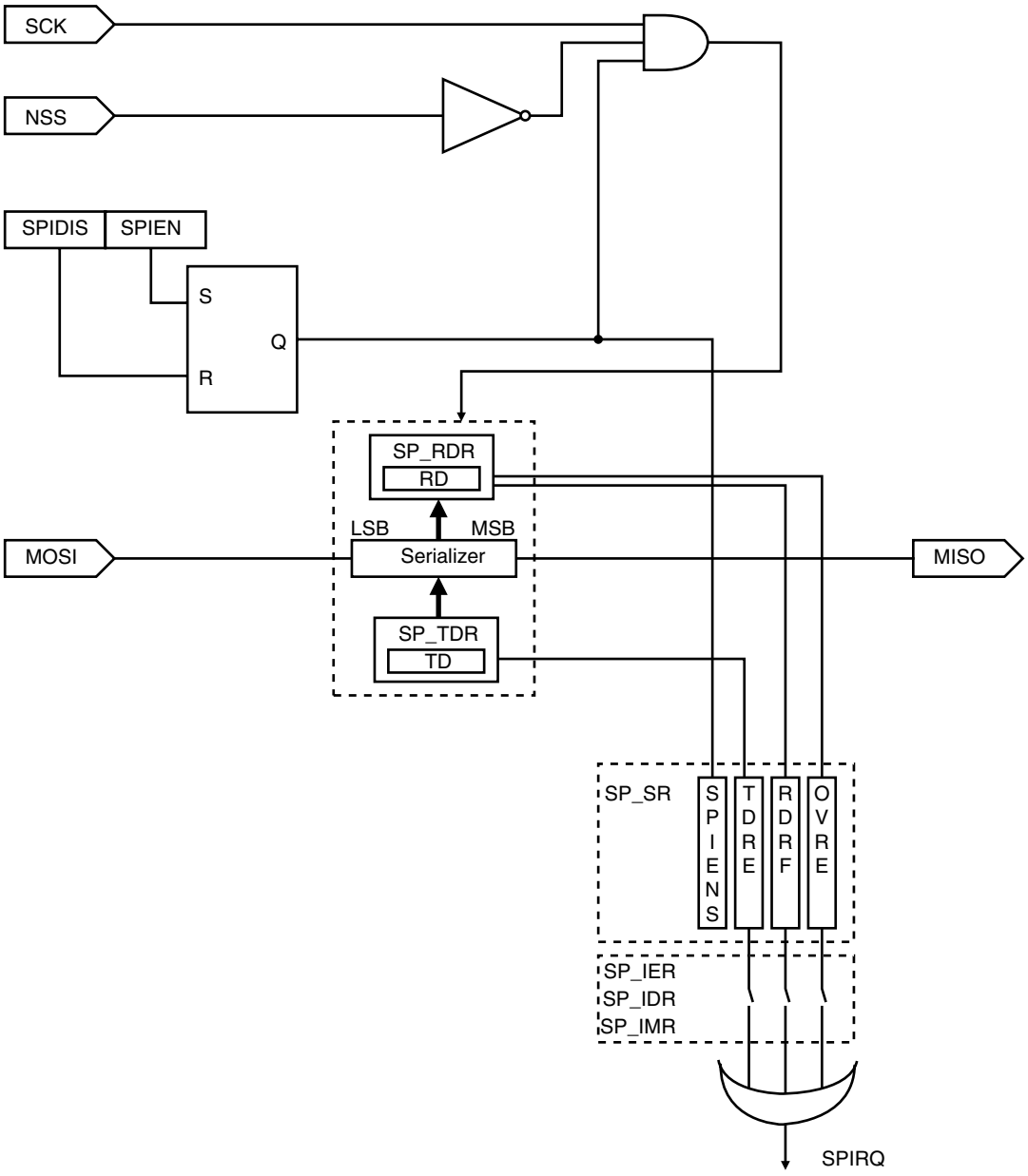


Slave Mode

In slave mode, the SPI waits for NSS to go active low before receiving the serial clock from an external master.

In slave mode, CPOL, NCPHA and BITS fields of SP_CSR0 are used to define the transfer characteristics. The other chip select registers are not used in slave mode.

Figure 49. SPI in Slave Mode



Data Transfer

The following waveforms show examples of data transfers.

Figure 50. SPI Transfer Format (NCPHA Equals One, 8 Bits per Transfer)

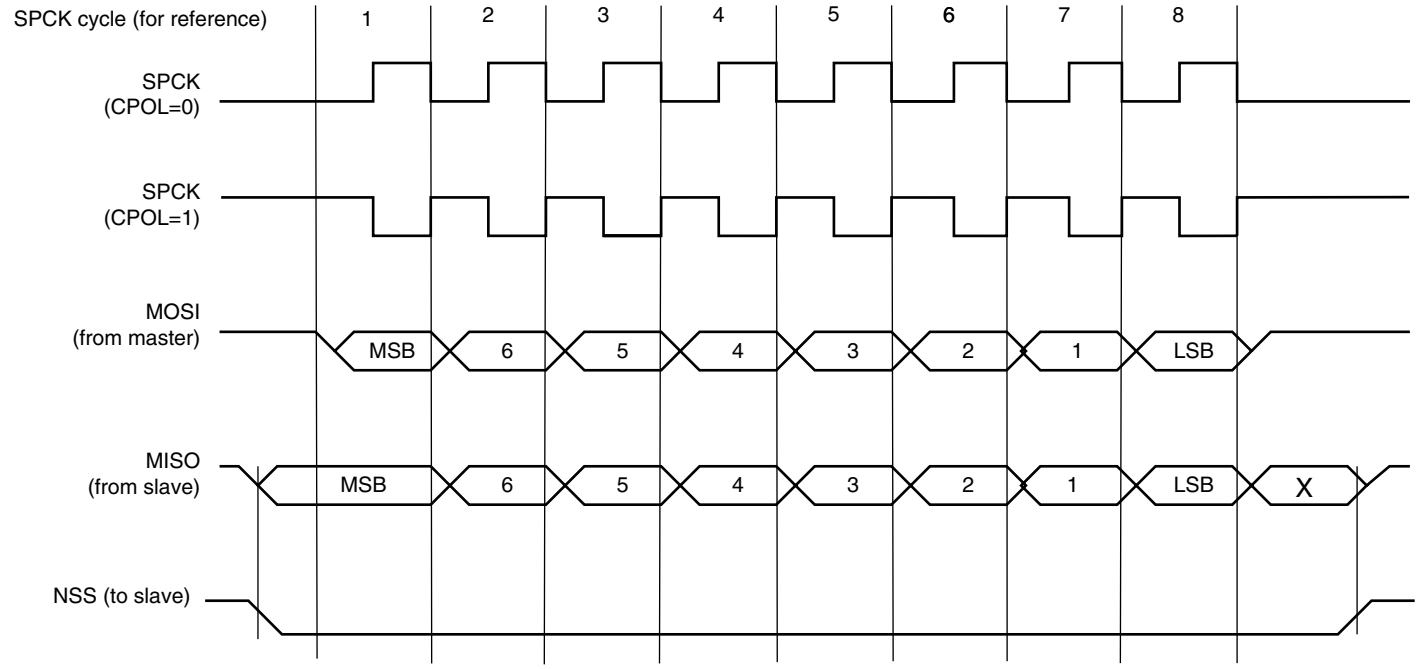


Figure 51. SPI Transfer Format (NCPHA Equals Zero, 8 Bits per Transfer)

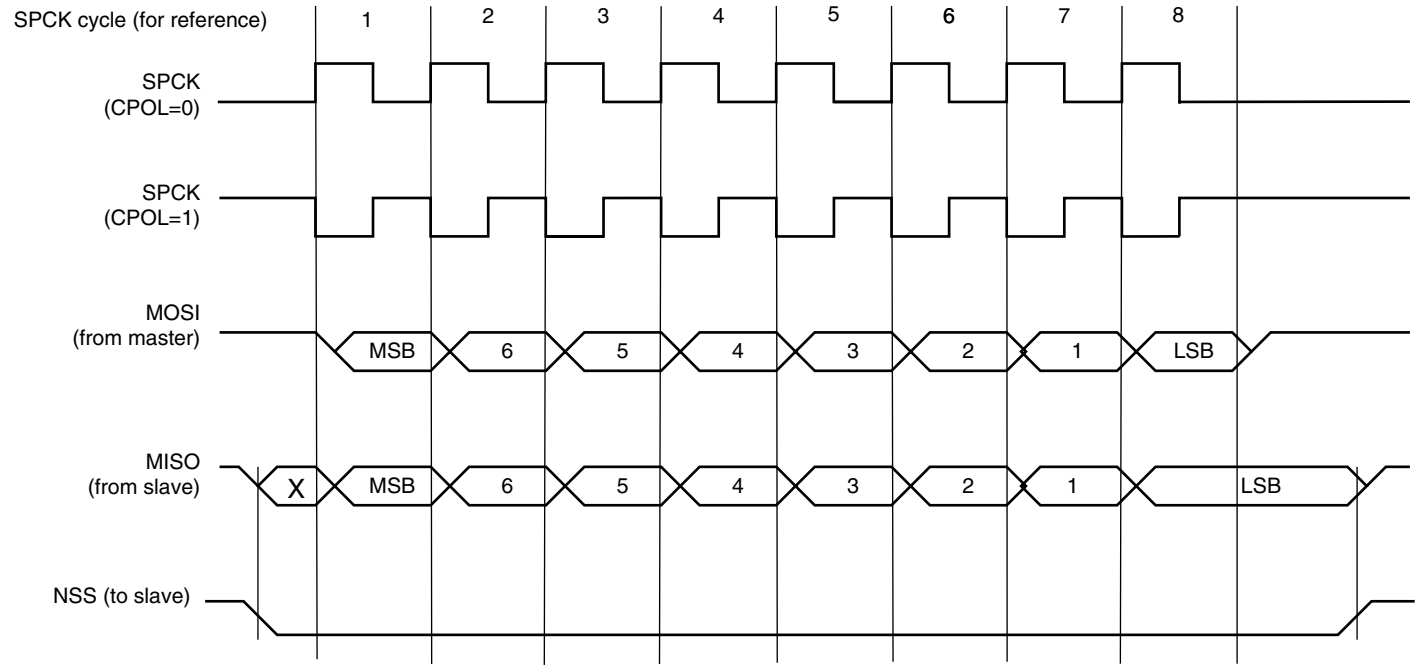
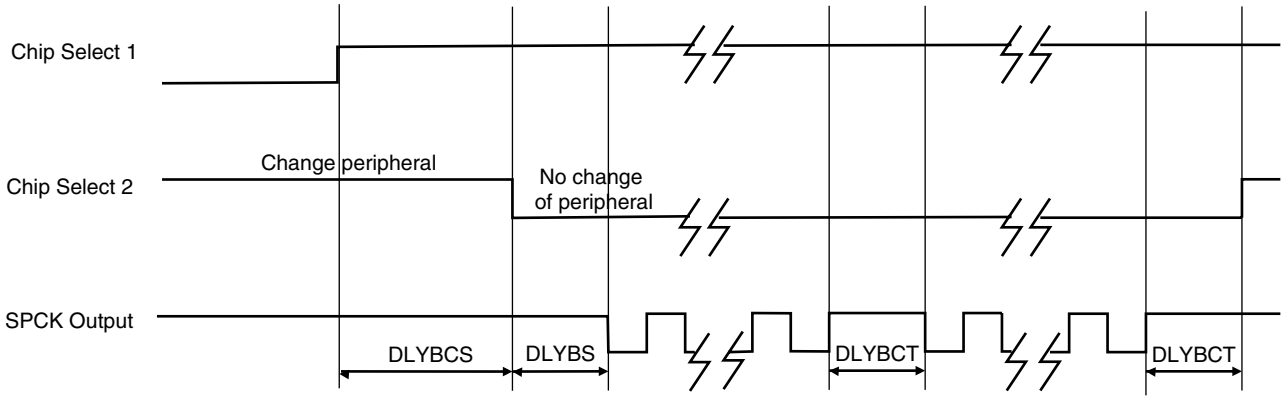


Figure 52. Programmable Delays (DLYBCS, DLYBS and DLYBCT)



Clock Generation

In master mode, the SPI master clock is either MCKI or MCKI/32, as defined by the MCK32 field of SP_MR. The SPI baud rate clock is generated by dividing the SPI master clock by a value between 4 and 510. The divisor is defined in the SCBR field in each chip select register. The transfer speed can thus be defined independently for each chip select signal.

CPOL and NCPHA in the chip select registers define the clock/data relationship between master and slave devices. CPOL defines the inactive value of the SPCK. NCPHA defines which edge causes data to change and which edge causes data to be captured.

In slave mode, the input clock low and high pulse duration must strictly be longer than two system clock (MCKI) periods.

Peripheral Data Controller

The SPI is closely connected to two Peripheral Data Controller channels. One is dedicated to the receiver. The other is dedicated to the transmitter.

The PDC channel is programmed using SP_TPR (Transmit Pointer) and SP_TCR (Transmit Counter) for the transmitter and SP_RPR (Receive Pointer) and SP_RCR (Receive Counter) for the receiver. The status of the PDC is given in SP_SR by the SPENDTX bit for the transmitter and by the SPENDRX bit for the receiver.

The pointer registers (SP_TPR and SP_RPR) are used to store the address of the transmit or receive buffers. The counter registers (SP_TCR and SP_RCR) are used to store the size of these buffers.

The receiver data transfer is triggered by the RDRF bit and the transmitter data transfer is triggered by TDRE. When a transfer is performed, the counter is decremented and the pointer is incremented. When the counter reaches 0, the status bit is set (SPENDRX for the receiver, SPENDTX for the transmitter in SP_SR) and can be programmed to generate an interrupt. While the counter is at zero, the status bit is asserted and transfers are disabled.

SPI Programmer's Model

SPI Base Address: 0xFFFBC000

Table 13. SPI Memory Map

Offset	Register	Name	Access	Reset State
0x00	Control Register	SP_CR	Write only	–
0x04	Mode Register	SP_MR	Read/Write	0
0x08	Receive Data Register	SP_RDR	Read only	0
0x0C	Transmit Data Register	SP_TDR	Write only	–
0x10	Status Register	SP_SR	Read only	0
0x14	Interrupt Enable Register	SP_IER	Write only	–
0x18	Interrupt Disable Register	SP_IDR	Write only	–
0x1C	Interrupt Mask Register	SP_IMR	Read only	0
0x20	Receive Pointer Register	SP_RPR	Read/Write	0
0x24	Receive Counter Register	SP_RCR	Read/Write	0
0x28	Transmit Pointer Register	SP_TPR	Read/Write	0
0x2C	Transmit Counter Register	SP_TCR	Read/Write	0
0x30	Chip Select Register 0	SP_CSR0	Read/Write	0
0x34	Chip Select Register 1	SP_CSR1	Read/Write	0
0x38	Chip Select Register 2	SP_CSR2	Read/Write	0
0x3C	Chip Select Register 3	SP_CSR3	Read/Write	0

SPI Control Register

Register Name: SP_CR
Access Type: Write only

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

- **SPIEN: SPI Enable**
  0 = No effect.
  1 = Enables the SPI to transfer and receive data.
- **SPIDIS: SPI Disable**
  0 = No effect.
  1 = Disables the SPI.
  All pins are set in input mode and no data is received or transmitted.
  If a transfer is in progress, the transfer is finished before the SPI is disabled.
  If both SPIEN and SPIDIS are equal to one when the control register is written, the SPI is disabled.
- **SWRST: SPI Software reset**
  0 = No effect.
  1 = Resets the SPI.
  A software-triggered hardware reset of the SPI interface is performed.

SPI Mode Register

Register Name: SP_MR
Access Type: Read/Write

31	30	29	28	27	26	25	24
DLYBCS							
23	22	21	20	19	18	17	16
–	–	–	–	PCS			
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
LLB	–	–	–	MCK32	PCSDEC	PS	MSTR

- **MSTR: Master/Slave Mode**
  0 = SPI is in Slave mode.
  1 = SPI is in Master mode.
  MSTR configures the SPI interface for either master or slave mode operation.
- **PS: Peripheral Select**
  0 = Fixed Peripheral Select.
  1 = Variable Peripheral Select.
- **PCSDEC: Chip Select Decode**
  0 = The chip selects are directly connected to a peripheral device.
  1 = The four chip select lines are connected to a 4- to 16-bit decoder.
  When PCSDEC equals one, up to 16 chip select signals can be generated with the four lines using an external 4- to 16-bit decoder.
  The chip select registers define the characteristics of the 16 chip selects according to the following rules:
  SP_CSR0 defines peripheral chip select signals 0 to 3.
  SP_CSR1 defines peripheral chip select signals 4 to 7.
  SP_CSR2 defines peripheral chip select signals 8 to 11.
  SP_CSR3 defines peripheral chip select signals 12 to 15*.
  ***Note:** The 16th state corresponds to a state in which all chip selects are inactive. This allows a different clock configuration to be defined by each chip select register.*
- **MCK32: Clock Selection**
  0 = SPI master clock equals MCKI.
  1 = SPI master clock equals MCKI/32.
- **LLB: Local Loopback Enable**
  0 = Local loopback path disabled.
  1 = Local loopback path enabled.
  LLB controls the local loopback on the data serializer for testing in master mode only.
- **PCS: Peripheral Chip Select**
  This field is only used if Fixed Peripheral Select is active (PS=0).
  If PCSDEC=0:
  PCS = xxx0 NPCS[3:0] = 1110
  PCS = xx01 NPCS[3:0] = 1101
  PCS = x011 NPCS[3:0] = 1011
  PCS = 0111 NPCS[3:0] = 0111
  PCS = 1111 forbidden (no peripheral is selected)
  (x = don't care)
  If PCSDEC=1:
  NPCS[3:0] output signals = PCS
- **DLYBCS: Delay Between Chip Selects**
  This field defines the delay from NPCS inactive to the activation of another NPCS. The DLYBCS time guarantees non-overlapping chip selects and solves bus contentions in case of peripherals having long data float times.
  If DLYBCS is less than or equal to six, six SPI master clock periods will be inserted by default.
  Otherwise, the following equation determines the delay:
  Delay_ Between_Chip_Selects = DLYBCS * SPI_Master_Clock_period

SPI Receive Data Register

Register Name: SP_RDR
Access Type: Read only

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

- **RD: Receive Data**
  Data received by the SPI interface is stored in this register right-justified. Unused bits read zero.
- **PCS: Peripheral Chip Select Status**
  In master mode only, these bits indicate the value on the NPCS pins at the end of a transfer. Otherwise, these bits read as zero.

SPI Transmit Data Register

Register Name: SP_TDR
Access Type: Write only

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

- **TD: Transmit Data**
  Data which is to be transmitted by the SPI interface is stored in this register. Information to be transmitted must be written to the transmit data register in a right-justified format.
- **PCS: Peripheral Chip Select**
  This field is only used if Variable Peripheral Select is active (PS = 1) and if the SPI is in master mode.
  If PCSDEC = 0:
  PCS = xxx0 NPCS[3:0] = 1110
  PCS = xx01 NPCS[3:0] = 1101
  PCS = x011 NPCS[3:0] = 1011
  PCS = 0111 NPCS[3:0] = 0111
  PCS = 1111 forbidden (no peripheral is selected)
  (x = don't care)
  If PCSDEC = 1:
  NPCS[3:0] output signals = PCS

SPI Status Register

Register Name: SP_SR
Access Type: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	SPIENS
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	SPENDTX	SPENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full**
  0 = No data has been received since the last read of SP_RDR.
  1 = Data has been received and the received data has been transferred from the serializer to SP_RDR since the last read of SP_RDR.
- **TDRE: Transmit Data Register Empty**
  0 = Data has been written to SP_TDR and not yet transferred to the serializer.
  1 = The last data written in the Transmit Data Register has been transferred in the serializer.
  TDRE equals zero when the SPI is disabled or at reset. The SPI enable command sets this bit to one.
- **MODF: Mode Fault Error**
  0 = No mode fault has been detected since the last read of SP_SR.
  1 = A mode fault occurred since the last read of the SP_SR.
- **OVRES: Overrun Error Status**
  0 = No overrun has been detected since the last read of SP_SR.
  1 = An overrun has occurred since the last read of SP_SR.
  An overrun occurs when SP_RDR is loaded at least twice from the serializer since the last read of the SP_RDR.
- **SPENDRX: End of Receiver Transfer**
  0 = The End of Transfer signal from the Peripheral Data Controller channel dedicated to the receiver is inactive.
  1 = The End of Transfer signal from the Peripheral Data Controller channel dedicated to the receiver is active.
- **SPENDTX: End of Transmitter Transfer**
  0 = The End of Transfer signal from the Peripheral Data Controller channel dedicated to the transmitter is inactive.
  1 = The End of Transfer signal from the Peripheral Data Controller channel dedicated to the transmitter is active.
- **SPIENS: SPI Enable Status**
  0 = SPI is disabled.
  1 = SPI is enabled.

SPI Interrupt Enable Register

Register Name: SP_IER
Access Type: Write only

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

- **RDRF: Receive Data Register Full Interrupt Enable**
  0 = No effect.
  1 = Enables the Receiver Data Register Full Interrupt.
- **TDRE: SPI Transmit Data Register Empty Interrupt Enable**
  0 = No effect.
  1 = Enables the Transmit Data Register Empty Interrupt.
- **MODF: Mode Fault Error Interrupt Enable**
  0 = No effect.
  1 = Enables the Mode Fault Interrupt.
- **OVRES: Overrun Error Interrupt Enable**
  0 = No effect.
  1 = Enables the Overrun Error Interrupt.
- **SPENDRX: End of Receiver Transfer Interrupt Enable**
  0 = No effect.
  1 = Enables the End of Receiver Transfer Interrupt.
- **SPENDTX: End of Transmitter Transfer Interrupt Enable**
  0 = No effect.
  1 = Enables the End of Transmitter Transfer Interrupt.

SPI Interrupt Disable Register

Register Name: SP_IDR
Access Type: Write only

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

- **RDRF: Receive Data Register Full Interrupt Disable**
  0 = No effect.
  1 = Disables the Receiver Data Register Full Interrupt.
- **TDRE: Transmit Data Register Empty Interrupt Disable**
  0 = No effect.
  1 = Disables the Transmit Data Register Empty Interrupt.
- **MODF: Mode Fault Interrupt Disable**
  0 = No effect.
  1 = Disables the Mode Fault Interrupt.
- **OVRES: Overrun Error Interrupt Disable**
  0 = No effect.
  1 = Disables the Overrun Error Interrupt.
- **SPENDRX: End of Receiver Transfer Interrupt Disable**
  0 = No effect.
  1 = Disables the End of Receiver Transfer Interrupt.
- **SPENDTX: End of Transmitter Transfer Interrupt Disable**
  0 = No effect.
  1 = Disables the End of Transmitter Transfer Interrupt.

SPI Interrupt Mask Register

Register Name: SP_IMR
Access Type: Read only

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

- **RDRF: Receive Data Register Full Interrupt Mask**
  0 = Receive Data Register Full Interrupt is disabled.
  1 = Receive Data Register Full Interrupt is enabled.
- **TDRE: Transmit Data Register Empty Interrupt Mask**
  0 = Transmit Data Register Empty Interrupt is disabled.
  1 = Transmit Data Register Empty Interrupt is enabled.
- **MODF: Mode Fault Interrupt Mask**
  0 = Mode Fault Interrupt is disabled.
  1 = Mode Fault Interrupt is enabled.
- **OVRES: Overrun Error Interrupt Mask**
  0 = Overrun Error Interrupt is disabled.
  1 = Overrun Error Interrupt is enabled.
- **SPENDRX: End of Receiver Transfer Interrupt Mask**
  0 = End of Receiver Transfer Interrupt is disabled.
  1 = End of Receiver Transfer Interrupt is enabled.
- **SPENDTX: End of Transmitter Transfer Interrupt Mask**
  0 = End of Transmitter Transfer Interrupt is disabled.
  1 = End of Transmitter Transfer Interrupt is enabled.

SPI Receive Pointer Register

Name: SP_RPR
Access Type: Read/Write

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

- **RXPTR: Receive Pointer**
  RXPTR must be loaded with the address of the receive buffer.

SPI Receive Counter Register

Name: SP_RCR
Access Type: Read/Write

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

- **RXCTR: Receive Counter**
  RXCTR must be loaded with the size of the receive buffer.
  0: Stop peripheral data transfer dedicated to the receiver.
  1-65535: Start peripheral data transfer if RDRF is active.

SPI Transmit Pointer Register

Name: SP_TPR
Access Type: Read/Write

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

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

SPI Transmit Counter Register

Name: SP_TCR
Access Type: Read/Write

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

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

SPI Chip Select Register

Register Name: SP_CSR0...SP_CSR3
Access Type: Read/Write

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

- **CPOL: Clock Polarity**
  0 = The inactive state value of SPCK is logic level zero.
  1 = The inactive state value of SPCK is logic level one.
  CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with NCPHA to produce a desired clock/data relationship between master and slave devices.
- **NCPHA: Clock Phase**
  0 = Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.
  1 = Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.
  NCPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. NCPHA is used with CPOL to produce a desired clock/data relationship between master and slave devices.
- **BITS: Bits Per Transfer**
  The BITS field determines the number of data bits transferred. Reserved values should not be used.

BITS[3:0]	Bits per Transfer
0000	8
0001	9
0010	10
0011	11
0100	12
0101	13
0110	14
0111	15
1000	16
1001	Reserved
1010	Reserved
1011	Reserved
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

- **SCBR: Serial Clock Baud Rate**
  In master mode, the SPI interface uses a modulus counter to derive the SPCK baud rate from the SPI master clock (selected between MCKI and MCKI/32). The baud rate is selected by writing a value from 2 to 255 in the field SCBR. The following equation determines the SPCK baud rate:

$$SPCK\_Baud\_Rate = \frac{SPI\_Master\_Clock\_frequency}{2 \times SCBR}$$

  Giving SCBR a value of zero or one disables the baud rate generator. SPCK is disabled and assumes its inactive state value. No serial transfers may occur. At reset, baud rate is disabled.
- **DLYBS: Delay Before SPCK**
  This field defines the delay from NPCS valid to the first valid SPCK transition.
  When DLYBS equals zero, the NPCS valid to SPCK transition is 1/2 the SPCK clock period.
  Otherwise, the following equation determines the delay:
  NPCS_to_SPCK_Delay = DLYBS * SPI_Master_Clock_period
- **DLYBCT: Delay Between Consecutive Transfers**
  This field defines the delay between two consecutive transfers with the same peripheral without removing the chip select. The delay is always inserted after each transfer and before removing the chip select if needed.
  When DLYBCT equals zero, a delay of four SPI master clock periods are inserted.
  Otherwise, the following equation determines the delay:
  Delay_After_Transfer = 32 * DLYBCT * SPI_Master_Clock_period

TC: Timer/Counter

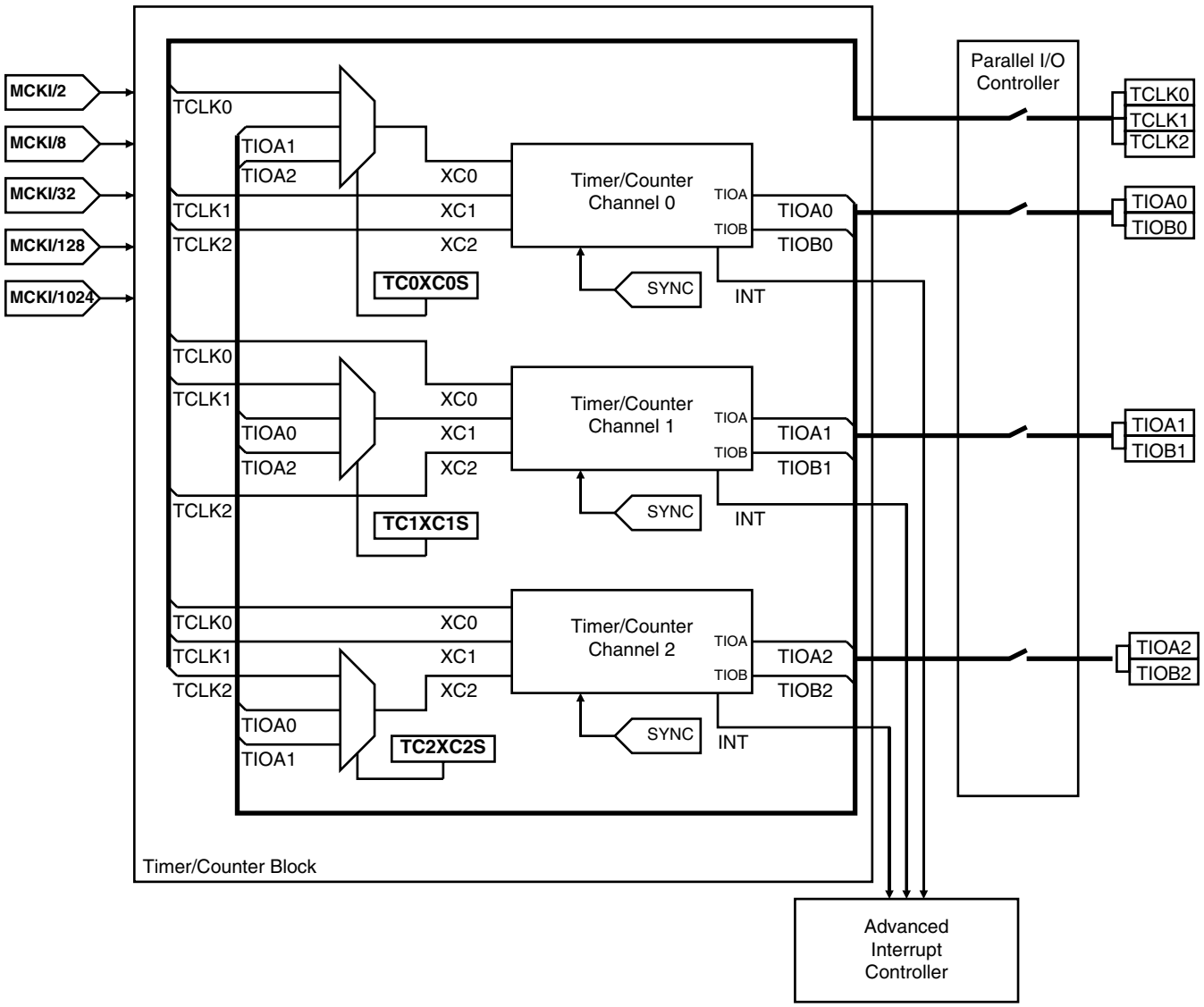
The AT91M63200 features two Timer/Counter blocks, each containing three identical 16-bit Timer/Counter channels. Each channel can be independently programmed to perform a wide range of functions including frequency measurement, event counting, interval measurement, pulse generation, delay timing and pulse width modulation.

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

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

Each Timer/Counter block operates independently and has a complete set of block and channel registers. Since they are identical in operation, only one block is described below (see "Timer/Counter Description" on page 113). The internal configuration of a single Timer/Counter block is shown in Figure 53.

Figure 53. TC Block Diagram



Signal Name Description

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

Note: *After a hardware reset, the TC clock is disabled by default (see "PMC: Power Management Controller" on page 139). The user must configure the Power Management Controller before any access to the user interface of the TC.*

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

Each Timer/CTimer/Counterounter channel is identical in operation. The registers for channel programming are listed in Table 15.

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 input clock. When the counter reaches the value 0xFFFF and passes to 0x0000, an overflow occurs and the bit COVFS in TC_SR (Status Register) is set.

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

Clock Selection

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

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

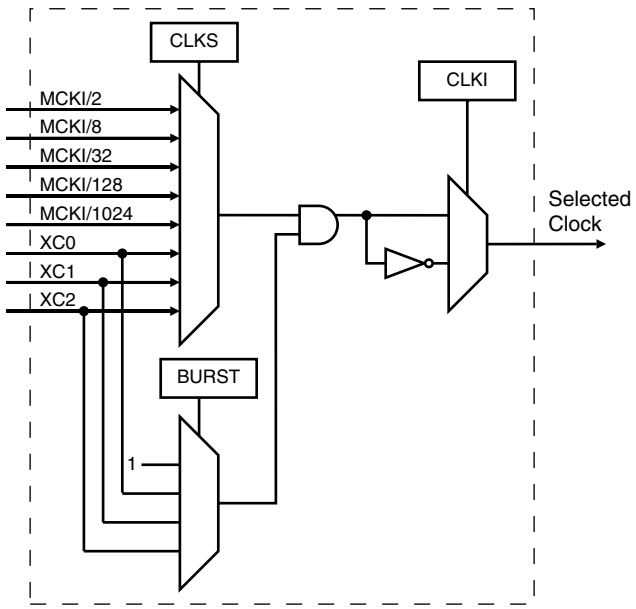
- Internal clock signals: MCKI/2, MCKI/8, MCKI/32, MCKI/128, MCKI/1024
- External clock signals: XC0, XC1 or XC2

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 (MCKI) period. The external clock frequency must be at least 2.5 times lower than the system clock.

Figure 54. Clock Selection

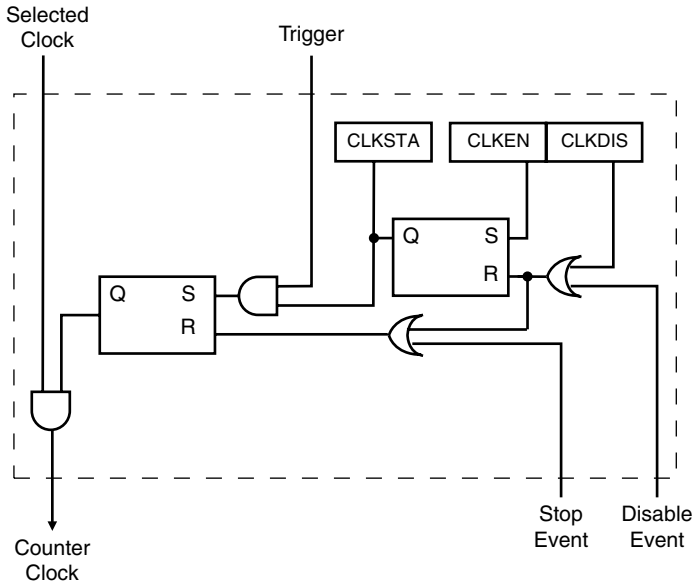


Clock Control

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 an RC compare event in waveform mode (CPCSTOP = 1 in TC_CMR). The start and the stop commands have an effect only if the clock is enabled.

Figure 55. Clock Control



Timer/Counter Operating Modes

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

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

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

Trigger

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

The following triggers are common to both modes:

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

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

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

Capture Operating Mode

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

Figure 56 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

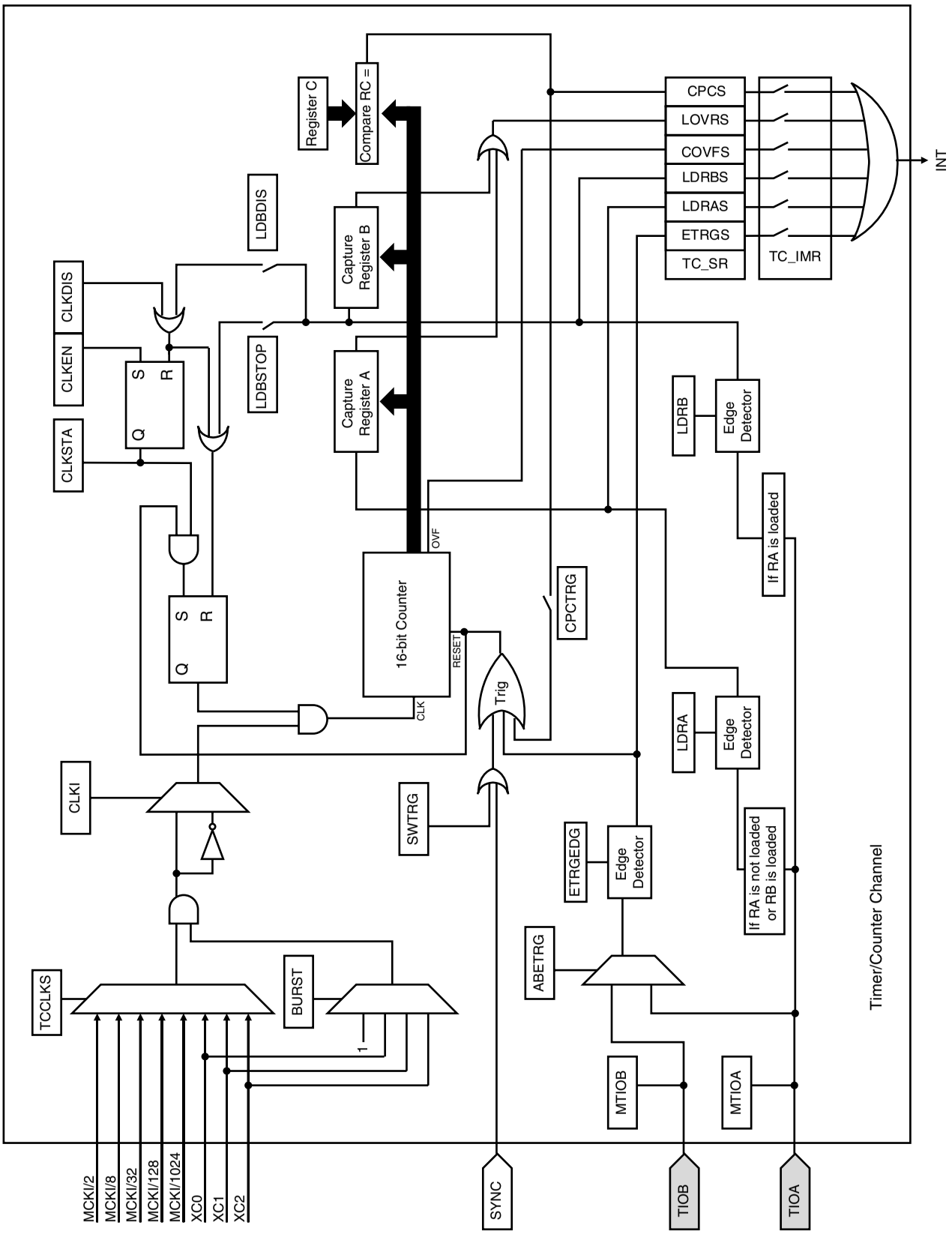
  RA or RB has been loaded at least twice without any read of the corresponding register, since the last read of the status.
- LDRAS: Load RA Status

  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.

Figure 56. 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 57 shows the configuration of the TC channel when programmed in waveform operating mode.

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

In waveform operating mode, RA, RB and RC are all used as compare registers.

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

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

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

External Event/Trigger Conditions

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

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

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

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

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

Output Controller

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

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

The tables below show which parameter in TC_CMR 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 an RA Compare match at least once since the last read of the status.
- CPBS: RB Compare Status

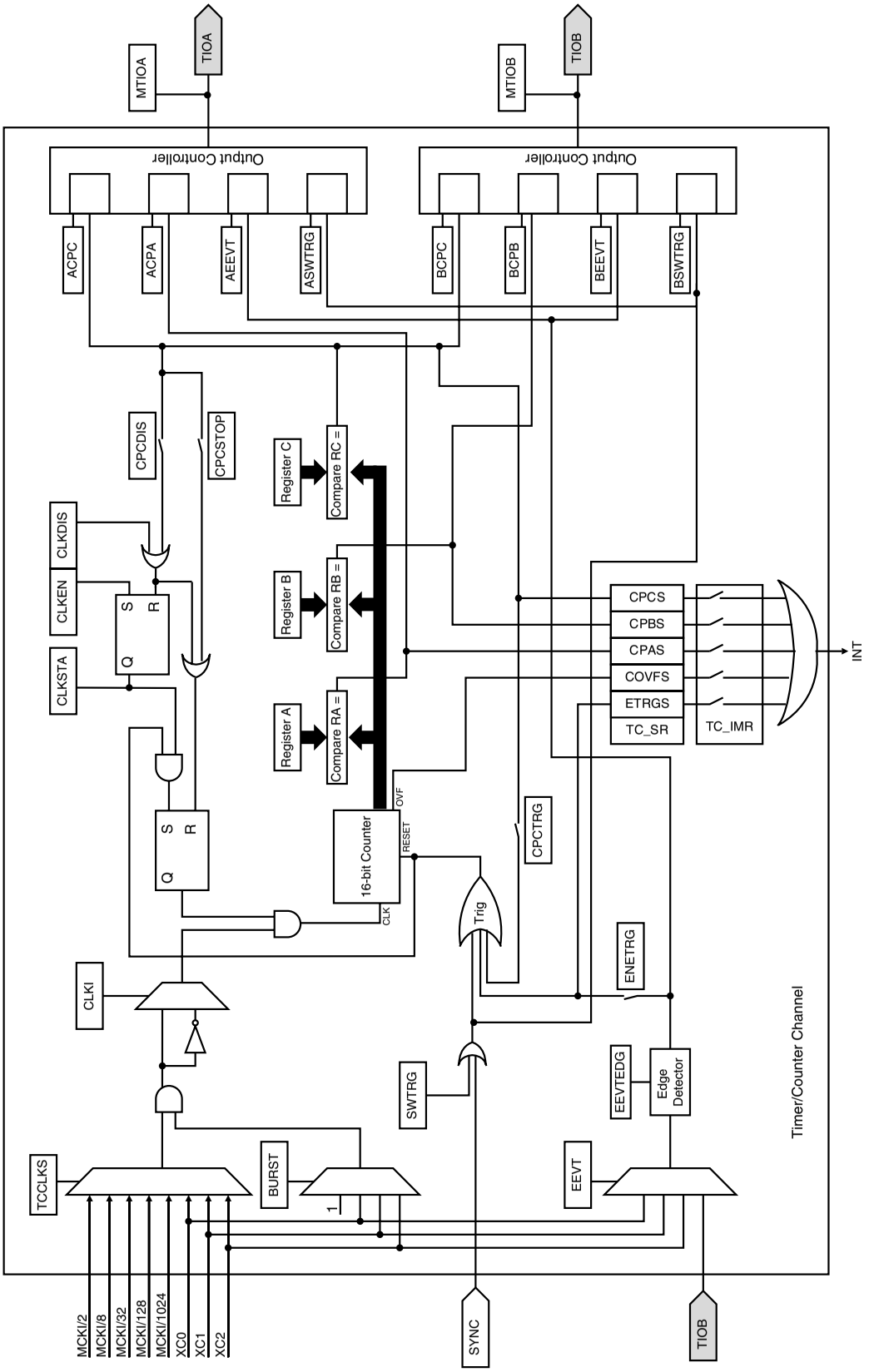
  There has been an RB Compare match at least once since the last read of the status.
- CPCS: RC Compare Status

  There has been an RC Compare match at least once since the last read of the status.
- COVFS: Counter Overflow Status

  Counter has attempted to count past $FFFF since the last read of the status.
- ETRGS: External Trigger Status

  External trigger has been detected since the last read of the status.

Figure 57. Waveform Mode



TC User Interface

TC Block 0 Base Address: 0xFFFD0000
TC Block 1 Base Address: 0xFFFD4000

Table 14. TC Global Memory Map

Offset	Channel/Register	Name	Access	Reset State
0x00	TC Channel 0	See Table 15		
0x40	TC Channel 1	See Table 15		
0x80	TC Channel 2	See Table 15		
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 15. The offset of each of the channel registers in Table 15 is in relation to the offset of the corresponding channel as mentioned in Table 14.

Table 15. TC Channel Memory Map

Offset	Register	Name	Access	Reset State
0x00	Channel Control Register	TC_CCR	Write only	–
0x04	Channel Mode Register	TC_CMR	Read/Write	0
0x08	Reserved			–
0x0C	Reserved			–
0x10	Counter Value	TC_CV	Read/Write	0
0x14	Register A	TC_RA	Read/Write[1]	0
0x18	Register B	TC_RB	Read/Write[1]	0
0x1C	Register C	TC_RC	Read/Write	0
0x20	Status Register	TC_SR	Read only	–
0x24	Interrupt Enable Register	TC_IER	Write only	–
0x28	Interrupt Disable Register	TC_IDR	Write only	–
0x2C	Interrupt Mask Register	TC_IMR	Read only	0

Note: 1. Read only if WAVE = 0

TC Block Control Register

Register Name: TC_BCR
Access Type: Write only

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

- **SYNC: Synchro Command**
  0 = No effect.
  1 = Asserts the SYNC signal, which generates a software trigger simultaneously for each of the channels.

TC Block Mode Register

Register Name: TC_BMR
Access Type: Read/Write

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

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

- **CLKEN: Counter Clock Enable Command**
  0 = No effect.
  1 = Enables the clock if CLKDIS is not 1.
- **CLKDIS: Counter Clock Disable Command**
  0 = No effect.
  1 = Disables the clock.
- **SWTRG: Software Trigger Command**
  0 = No effect.
  1 = A software trigger is performed: the counter is reset and clock is started.

TC Channel Mode Register: Capture Mode

Register Name: TC_CMR
Access Type: Read/Write

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

- **TCCLKS: Clock Selection**

TCCLKS			Clock Selected
0	0	0	MCKI/2
0	0	1	MCKI/8
0	1	0	MCKI/32
0	1	1	MCKI/128
1	0	0	MCKI/1024
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI: Clock Invert**
  0 = Counter is incremented on rising edge of the clock.
  1 = Counter is incremented on falling edge of the clock.
- **BURST: Burst Signal Selection**

BURST		
0	0	The clock is not gated by an external signal.
0	1	XC0 is ANDed with the selected clock.
1	0	XC1 is ANDed with the selected clock.
1	1	XC2 is ANDed with the selected clock.

- **LDBSTOP: Counter Clock Stopped with RB Loading**
  0 = Counter clock is not stopped when RB loading occurs.
  1 = Counter clock is stopped when RB loading occurs.
- **LDBDIS: Counter Clock Disable with RB Loading**
  0 = Counter clock is not disabled when RB loading occurs.
  1 = Counter clock is disabled when RB loading occurs.

- **ETRGEDG: External Trigger Edge Selection**

ETRGEDG		Edge
0	0	none
0	1	rising edge
1	0	falling edge
1	1	each edge

- **ABETRG: TIOA or TIOB External Trigger Selection**
  0 = TIOB is used as an external trigger.
  1 = TIOA is used as an external trigger.
- **CPCTRG: RC Compare Trigger Enable**
  0 = RC Compare has no effect on the counter and its clock.
  1 = RC Compare resets the counter and starts the counter clock.
- **WAVE = 0**
  0 = Capture Mode is enabled.
  1 = Capture Mode is disabled (waveform mode is enabled).
- **LDRA: RA Loading Selection**

LDRA		Edge
0	0	none
0	1	rising edge of TIOA
1	0	falling edge of TIOA
1	1	each edge of TIOA

- **LDRB: RB Loading Selection**

LDRB		Edge
0	0	none
0	1	rising edge of TIOA
1	0	falling edge of TIOA
1	1	each edge of TIOA

TC Channel Mode Register: Waveform Mode

Register Name: TC_CMR
Access Type: Read/Write

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

- **TCCLKS: Clock Selection**

TCCLKS			Clock Selected
0	0	0	MCKI/2
0	0	1	MCKI/8
0	1	0	MCKI/32
0	1	1	MCKI/128
1	0	0	MCKI/1024
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI: Clock Invert**
  0 = Counter is incremented on rising edge of the clock.
  1 = Counter is incremented on falling edge of the clock.
- **BURST: Burst Signal Selection**

BURST		
0	0	The clock is not gated by an external signal.
0	1	XC0 is ANDed with the selected clock.
1	0	XC1 is ANDed with the selected clock.
1	1	XC2 is ANDed with the selected clock.

- **CPCSTOP: Counter Clock Stopped with RC Compare**
  0 = Counter clock is not stopped when counter reaches RC.
  1 = Counter clock is stopped when counter reaches RC.
- **CPCDIS: Counter Clock Disable with RC Compare**
  0 = Counter clock is not disabled when counter reaches RC.
  1 = Counter clock is disabled when counter reaches RC.

- **EEVTEDG: External Event Edge Selection**

EEVTEDG		Edge
0	0	none
0	1	rising edge
1	0	falling edge
1	1	each edge

- **EEVT: External Event Selection**

EEVT		Signal selected as external event	TIOB Direction
0	0	TIOB	input[1]
0	1	XC0	output
1	0	XC1	output
1	1	XC2	output

Note: 1. *If TIOB is chosen as the external event signal, it is configured as an input and no longer generates waveforms.*

- **ENETRG: External Event Trigger Enable**
  0 = The external event has no effect on the counter and its clock. In this case, the selected external event only controls the TIOA output.
  1 = The external event resets the counter and starts the counter clock.
- **CPCTRG: RC Compare Trigger Enable**
  0 = RC Compare has no effect on the counter and its clock.
  1 = RC Compare resets the counter and starts the counter clock.
- **WAVE = 1**
  0 = Waveform Mode is disabled (capture mode is enabled).
  1 = Waveform Mode is enabled.
- **ACPA: RA Compare Effect on TIOA**

ACPA		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **ACPC: RC Compare Effect on TIOA**

ACPC		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

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

AEEVT		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

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

ASWTRG		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **BCPB: RB Compare Effect on TIOB**

BCPB		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

- **BCPC: RC Compare Effect on TIOB**

BCPC		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

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

BEEVT		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

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

BSWTRG		Effect
0	0	none
0	1	set
1	0	clear
1	1	toggle

TC Counter Value Register

Register Name: TC_CVR
Access Type: Read only

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

- **CV: Counter Value**
  CV contains the counter value in real time.

TC Register A

Register Name: TC_RA
Access Type: Read only if WAVE = 0, Read/Write if WAVE = 1

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

- **RA: Register A**
  RA contains the Register A value in real time.

TC Register B

Register Name: TC_RB
Access Type: Read only if WAVE = 0, Read/Write if WAVE = 1

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

- **RB: Register B**
  RB contains the Register B value in real time.

TC Register C

Register Name: TC_RC
Access Type: Read/Write

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

- **RC: Register C**
  RC contains the Register C value in real time.

TC Status Register

Register Name: TC_SR
Access Type: Read/Write

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

- **COVFS: Counter Overflow Status**
  0 = No counter overflow has occurred since the last read of the Status Register.
  1 = A counter overflow has occurred since the last read of the Status Register.
- **LOVRS: Load Overrun Status**
  0 = Load overrun has not occurred since the last read of the Status Register or WAVE = 1.
  1 = RA or RB have been loaded at least twice without any read of the corresponding register since the last read of the Status Register, if WAVE = 0.
- **CPAS: RA Compare Status**
  0 = RA Compare has not occurred since the last read of the Status Register or WAVE = 0.
  1 = RA Compare has occurred since the last read of the Status Register, if WAVE = 1.
- **CPBS: RB Compare Status**
  0 = RB Compare has not occurred since the last read of the Status Register or WAVE = 0.
  1 = RB Compare has occurred since the last read of the Status Register, if WAVE = 1.
- **CPCS: RC Compare Status**
  0 = RC Compare has not occurred since the last read of the Status Register.
  1 = RC Compare has occurred since the last read of the Status Register.
- **LDRAS: RA Loading Status**
  0 = RA Load has not occurred since the last read of the Status Register or WAVE = 1.
  1 = RA Load has occurred since the last read of the Status Register, if WAVE = 0.
- **LDRBS: RB Loading Status**
  0 = RB Load has not occurred since the last read of the Status Register or WAVE = 1.
  1 = RB Load has occurred since the last read of the Status Register, if WAVE = 0.
- **ETRGS: External Trigger Status**
  0 = External trigger has not occurred since the last read of the Status Register.
  1 = External trigger has occurred since the last read of the Status Register.
- **CLKSTA: Clock Enabling Status**
  0 = Clock is disabled.
  1 = Clock is enabled.
- **MTIOA: TIOA Mirror**
  0 = TIOA is low. If WAVE = 0, this means that TIOA pin is low. If WAVE = 1, this means that TIOA is driven low.
  1 = TIOA is high. If WAVE = 0, this means that TIOA pin is high. If WAVE = 1, this means that TIOA is driven high.
- **MTIOB: TIOB Mirror**
  0 = TIOB is low. If WAVE = 0, this means that TIOB pin is low. If WAVE = 1, this means that TIOB is driven low.
  1 = TIOB is high. If WAVE = 0, this means that TIOB pin is high. If WAVE = 1, this means that TIOB is driven high.

TC Interrupt Enable Register

Register Name: TC_IER
Access Type: Write only

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

- **COVFS: Counter Overflow**
  0 = No effect.
  1 = Enables the Counter Overflow Interrupt.
- **LOVRS: Load Overrun**
  0 = No effect.
  1: Enables the Load Overrun Interrupt.
- **CPAS: RA Compare**
  0 = No effect.
  1 = Enables the RA Compare Interrupt.
- **CPBS: RB Compare**
  0 = No effect.
  1 = Enables the RB Compare Interrupt.
- **CPCS: RC Compare**
  0 = No effect.
  1 = Enables the RC Compare Interrupt.
- **LDRAS: RA Loading**
  0 = No effect.
  1 = Enables the RA Load Interrupt.
- **LDRBS: RB Loading**
  0 = No effect.
  1 = Enables the RB Load Interrupt.
- **ETRGS: External Trigger**
  0 = No effect.
  1 = Enables the External Trigger Interrupt.

TC Interrupt Disable Register

Register Name: TC_IDR
Access Type: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: Counter Overflow**
  0 = No effect.
  1 = Disables the Counter Overflow Interrupt.
- **LOVRS: Load Overrun**
  0 = No effect.
  1 = Disables the Load Overrun Interrupt (if WAVE = 0).
- **CPAS: RA Compare**
  0 = No effect.
  1 = Disables the RA Compare Interrupt (if WAVE = 1).
- **CPBS: RB Compare**
  0 = No effect.
  1 = Disables the RB Compare Interrupt (if WAVE = 1).
- **CPCS: RC Compare**
  0 = No effect.
  1 = Disables the RC Compare Interrupt.
- **LDRAS: RA Loading**
  0 = No effect.
  1 = Disables the RA Load Interrupt (if WAVE = 0).
- **LDRBS: RB Loading**
  0 = No effect.
  1 = Disables the RB Load Interrupt (if WAVE = 0).
- **ETRGS: External Trigger**
  0 = No effect.
  1 = Disables the External Trigger Interrupt.

TC Interrupt Mask Register

Register Name: TC_IMR
Access Type: Read only

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

- **COVFS: Counter Overflow**
  0 = The Counter Overflow Interrupt is disabled.
  1 = The Counter Overflow Interrupt is enabled.
- **LOVRS: Load Overrun**
  0 = The Load Overrun Interrupt is disabled.
  1 = The Load Overrun Interrupt is enabled.
- **CPAS: RA Compare**
  0 = The RA Compare Interrupt is disabled.
  1 = The RA Compare Interrupt is enabled.
- **CPBS: RB Compare**
  0 = The RB Compare Interrupt is disabled.
  1 = The RB Compare Interrupt is enabled.
- **CPCS: RC Compare**
  0 = The RC Compare Interrupt is disabled.
  1 = The RC Compare Interrupt is enabled.
- **LDRAS: RA Loading**
  0 = The Load RA Interrupt is disabled.
  1 = The Load RA Interrupt is enabled.
- **LDRBS: RB Loading**
  0 = The Load RB Interrupt is disabled.
  1 = The Load RB Interrupt is enabled.
- **ETRGS: External Trigger**
  0 = The External Trigger Interrupt is disabled.
  1 = The External Trigger Interrupt is enabled.

WD: Watchdog Timer

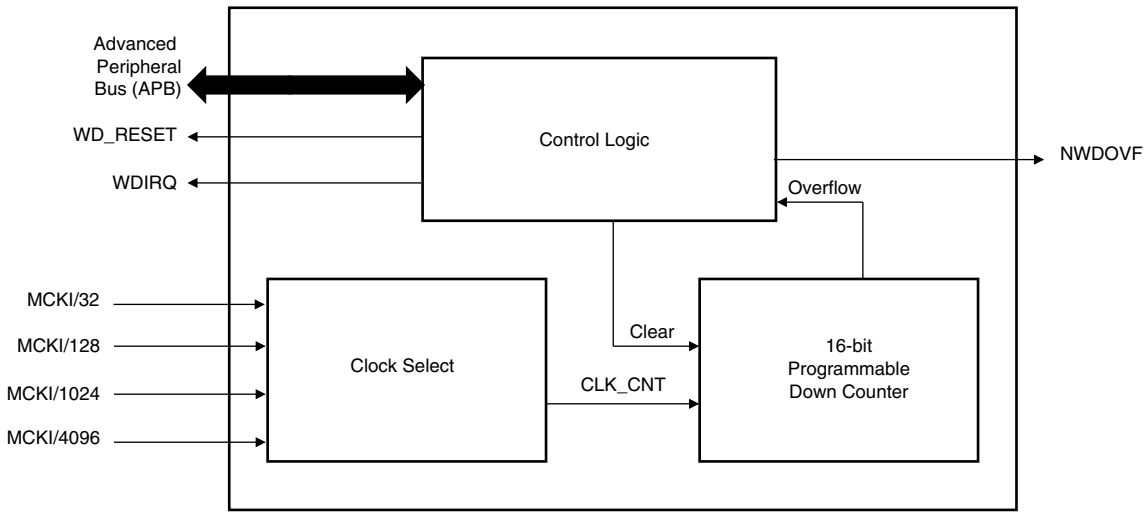
The AT91 series microcontrollers have an internal Watchdog timer which can be used to prevent system lock-up if the software becomes trapped in a deadlock. In normal operation the user reloads the Watchdog at regular intervals before the timer overflow occurs. If an overflow does occur, the Watchdog 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 58). See also "Watchdog Reset" on page 9.
- 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 MCKI 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: MCKI/32, MCKI/128, MCKI/1024 or MCKI/4096. The selection is made using the WDCLKS parameter in WD_CMR. This provides a programmable time-out period of 4ms to 8s 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 58. Watchdog Timer Block Diagram



WD User Interface

WD Base Address: 0xFFFF8000

Table 16. 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

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: Watchdog Enable**
  0 = Watchdog is disabled and does not generate any signals.
  1 = Watchdog is enabled and generates enabled signals.
- **RSTEN: Reset Enable**
  0 = Generation of an internal reset by the Watchdog is disabled.
  1 = When overflow occurs, the Watchdog generates an internal reset.
- **IRQEN: Interrupt Enable**
  0 = Generation of an interrupt by the Watchdog is disabled.
  1 = When overflow occurs, the Watchdog generates an interrupt.
- **EXTEN: External Signal Enable**
  0 = Generation of a pulse on the pin NWDOVF by the Watchdog is disabled.
  1 = When an overflow occurs, a pulse on the pin NWDOVF is generated.
- **OKEY: Overflow Access Key**
  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

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
0	0	MCKI/32
0	1	MCKI/128
1	0	MCKI/1024
1	1	MCKI/4096

- **HPCV: High Preload Counter Value**
  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**
  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

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**
  0xC071 = Watchdog counter is restarted.
  Other value = No effect.

WD Status Register

Name: WD_SR
Access: Read only

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

- **WDOVF: Watchdog Overflow**
  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_CMR
   (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).

PMC: Power Management Controller

The AT91M63200 Power Management Controller allows optimization of power consumption. The PMC controls the system clocks and the peripheral clocks. Two sets of registers are mapped in the user interface in order to enable and to disable these clocks.

System Clock

The AT91M63200 has only one system clock: the ARM core clock. It can be enabled and disabled by writing the System Clock Enable (PMC_SCER) and System Clock Disable Registers (PMC_SCDR). The status of this clock (at least for debug purpose) can be read in the System Clock Status Register (PMC_SCSR).

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

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

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

Peripheral Clocks

The clock of each peripheral integrated in the AT91M63200 can be individually enabled and disabled by writing in the Peripheral Clock Enable (PMC_PCER) and Peripheral Clock Disable Registers (PMC_PCDR). The status of the peripheral clocks can be read in the Peripheral Clock Status Register (PMC_PCSR).

When a peripheral clock is disabled, the clock is immediately stopped. When the clock is re-enabled, the peripheral resumes action where it left off.

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

The peripheral clocks are automatically disabled after a reset.

The bits defined to control the clocks of the peripherals correspond to the bits controlling the interrupt sources in the AIC.

PMC User Interface

Base Address: 0xFFFF4000

Table 17. PMC Memory Map

Offset	Register	Name	Access	Reset State
0x00	System Clock Enable Register	PMC_SCER	Write only	–
0x04	System Clock Disable Register	PMC_SCDR	Write only	–
0x08	System Clock Status Register	PMC_SCSR	Read only	0x1
0x0C	Reserved			
0x10	Peripheral Clock Enable Register	PMC_PCER	Write only	–
0x14	Peripheral Clock Disable Register	PMC_PCDR	Write only	–
0x18	Peripheral Clock Status Register	PMC_PCSR	Read only	0x0

PMC System Clock Enable Register

Register Name: PMC_SCER
Access Type: Write only

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

- **CPU: CPU Clock Enable**
  0 = No effect.
  1 = Enables the CPU clock.

PMC System Clock Disable Register

Register Name: PMC_SCDR
Access Type: Write only

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

- **CPU: CPU Clock Disable**
  0 = No effect.
  1 = Disables the CPU (ARM core) clock.

PMC System Clock Status Register

Register Name: PMC_SCSR
Access Type: Read only

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

- **CPU: CPU Clock Status**
  0 = CPU clock is enabled.
  1 = CPU clock is disabled.

PMC Peripheral Clock Enable Register

Register Name: PMC_PCER
Access Type: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	PIOB	PIOA	–	TC5	TC4	TC3	TC2
7	6	5	4	3	2	1	0
TC1	TC0	SPI	US2	US1	US0	–	–

- **US0: USART0 Clock. Enable**
  0 = No effect.
  1 = Enables the USART0 clock.
- **US1: USART1 Clock. Enable**
  0 = No effect.
  1 = Enables the USART1 clock.
- **US2: USART2 Clock. Enable**
  0 = No effect.
  1 = Enables the USART2 clock.
- **SPI: SPI Clock. Enable**
  0 = No effect.
  1 = Enables the SPI clock.
- **TC0: TC0 Clock. Enable**
  0 = No effect.
  1 = Enables the TC0 clock.
- **TC1: TC1 Clock. Enable**
  0 = No effect.
  1 = Enables the TC1 clock.
- **TC2: TC2 Clock. Enable**
  0 = No effect.
  1 = Enables the TC2 clock.
- **TC3: TC3 Clock. Enable**
  0 = No effect.
  1 = Enables the TC3 clock.
- **TC4: TC4 Clock. Enable**
  0 = No effect.
  1 = Enables the TC4 clock.
- **TC5: TC5 Clock. Enable**
  0 = No effect.
  1 = Enables the TC5 clock.
- **PIOA: PIOA Clock. Enable**
  0 = No effect.
  1 = Enables the PIOA clock.
- **PIOB: PIOB Clock. Enable**
  0 = No effect.
  1 = Enables the PIOB clock.

PMC Peripheral Clock Disable Register

Register Name: PMC_PCDR
Access Type: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	PIOB	PIOA	–	TC5	TC4	TC3	TC2
7	6	5	4	3	2	1	0
TC1	TC0	SPI	US2	US1	US0	–	–

- **US0: USART0 Clock. Disable**
  0 = No effect.
  1 = Disables the USART0 clock.
- **US1: USART1 Clock. Disable**
  0 = No effect.
  1 = Disables the USART1 clock.
- **US2: USART2 Clock. Disable**
  0 = No effect.
  1 = Disables the USART2 clock.
- **SPI: SPI Clock. Disable**
  0 = No effect.
  1 = Disables the SPI clock.
- **TC0: TC0 Clock. Disable**
  0 = No effect.
  1 = Disables the TC0 clock.
- **TC1: TC1 Clock. Disable**
  0 = No effect.
  1 = Disables the TC1 clock.
- **TC2: TC2 Clock. Disable**
  0 = No effect.
  1 = Disables the TC2 clock.
- **TC3: TC3 Clock. Disable**
  0 = No effect.
  1 = Disables the TC3 clock.
- **TC4: TC4 Clock. Disable**
  0 = No effect.
  1 = Disables the TC4 clock.
- **TC5: TC5 Clock. Disable**
  0 = No effect.
  1 = Disables the TC5 clock.
- **PIOA: PIOA Clock. Disable**
  0 = No effect.
  1 = Disables the Parallel IO A clock.
- **PIOB: PIOB Clock. Disable**
  0 = No effect.
  1 = Disables the Parallel IO B clock.

PMC Peripheral Clock Status Register

Register Name: PMC_PCSR
Access Type: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	PIOB	PIOA	–	TC5	TC4	TC3	TC2
7	6	5	4	3	2	1	0
TC1	TC0	SPI	US2	US1	US0	–	–

- **US0: USART0 Clock Status**
  0 = USART0 clock is disabled.
  1 = USART0 clock is enabled.
- **US1: USART1 Clock Status**
  0 = USART1 clock is disabled.
  1 = USART1 clock is enabled.
- **US2: USART2 Clock Status**
  0 = USART2 clock is disabled.
  1 = USART2 clock is enabled.
- **SPI: SPI Clock Status**
  0 = SPI clock is disabled.
  1 = SPI clock is enabled.
- **TC0: TC0 Clock Status**
  0 = TC0 clock is disabled.
  1 = TC0 clock is enabled.
- **TC1: TC1 Clock Status**
  0 = TC1 clock is disabled.
  1 = TC1 clock is enabled.
- **TC2: TC2 Clock Status**
  0 = TC2 clock is disabled.
  1 = TC2 clock is enabled.
- **TC3: TC3 Clock Status**
  0 = TC3 clock is disabled.
  1 = TC3 clock is enabled.
- **TC4: TC4 Clock Status**
  0 = TC4 clock is disabled.
  1 = TC4 clock is enabled.
- **TC5: TC5 Clock Status**
  0 = TC5 clock is disabled.
  1 = TC5 clock is enabled.
- **PIOA: PIOA Clock Status**
  0 = PIOA clock is disabled.
  1 = PIOA clock is enabled.
- **PIOB: PIOB Clock Status**
  0 = PIOB clock is disabled.
  1 = PIOB clock is enabled.

SF: Special Function Registers

The M63X00 provides registers which implement the following special functions:

- Chip identification: a chip identifier module which enables software to recognize certain characteristics of the chip and the version number
- RESET status
- Protect mode (see “Protect Mode” on page 42)

SF User Interface

Chip ID Base Address: 0xFFF00000

Table 18. SF Memory Map

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

Chip ID Register

Register Name: SF_CIDR
Access Type: Read only

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

- **NVPSIZ: Nonvolatile Program Memory Size**

NVPSIZ				Size
0	0	0	0	None
0	0	1	1	32K bytes
0	1	0	1	64K bytes
0	1	1	1	128K bytes
1	0	0	1	256K bytes
Others				Reserved

- **NVDSIZ: Nonvolatile Data Memory Size**

NVDSIZ				Size
0	0	0	0	None
Others				Reserved

- **VDSIZ: Volatile Data Memory Size**

VDSIZ				Size
0	0	0	0	None
0	0	0	1	1K bytes
0	0	1	0	2K bytes
0	1	0	0	4K bytes
1	0	0	0	8K bytes
Others				Reserved

- **ARCH: Chip Architecture**
  Code of Architecture: Two BCD digits

0110 0011	AT91x63yyy
0100 0000	AT91x40yyy

- **NVPTYP: Nonvolatile Program Memory Type**

NVPTYP			Type
0	0	0	Reserved
0	0	1	“M” Series (Mask ROM or ROM less)
0	1	0	“C” Series (Programmable Flash through Parallel Port)
0	1	1	“S” Series (Programmable Flash through Serial Port)
1	x	x	Reserved

- **EXT: Extension Flag**

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

Reset Status Register

Register Name: SF_RSR
Access Type: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
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
0x6C	External Pin
0x53	Internal Watchdog

SF Protect Mode Register

Register Name: SF_PMR
Access Type: Read/Write
Reset Value: 0

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**
  Used only when writing SF_PMR. PMRKEY is read as 0.
  0x27A8: Write access in SF_PMR is allowed.
  Other value: Write access in SF_PMR is prohibited.
- **AIC: AIC Protect Mode Enable**
  0 = The Advanced Interrupt Controller runs in normal mode.
  1 = The Advanced Interrupt Controller runs in protect mode.
  See “Protect Mode” on page 42.

JTAG Boundary Scan Register

The Boundary Scan Register (BSR) contains 303 bits which correspond to active pins and associated control signals.

Each AT91M63200 input pin has a corresponding bit in the Boundary Scan Register for observability.

Each AT91M63200 output pin has a corresponding 2-bit register in the BSR. The OUTPUT bit contains data which can be forced on the pad. The CTRL bit can put the pad into high impedance.

Each AT91M63200 in/out pin corresponds to a 3-bit register in the BSR. The OUTPUT bit contains data which can be forced on the pad. The INPUT bit is for the observability of data applied to the pad. The CTRL bit selects the direction of the pad.

Table 19. JTAG Boundary Scan Register

Bit Number	Pin Name	Pin Type	Associated BSR Cells
303	NWAIT	INPUT	INPUT
302	NRST	INPUT	INPUT
301	PB18/BMS	INOUT	OUTPUT
300			INPUT
299			CTRL
298	MCKI	INPUT	INPUT
297	NWDOVF	OUTPUT	OUTPUT
296			CTRL
295	PB17/MCKO	INOUT	OUTPUT
294			INPUT
293			CTRL
292	PB16	INOUT	OUTPUT
291			INPUT
290			CTRL
289	PB15	INOUT	OUTPUT
288			INPUT
287			CTRL
286	PB14	INOUT	OUTPUT
285			INPUT
284			CTRL
283	PB13	INOUT	OUTPUT
282			INPUT
281			CTRL

Table 19. JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
280	PB12	INOUT	OUTPUT
279			INPUT
278			CTRL
277	PB11	INOUT	OUTPUT
276			INPUT
275			CTRL
274	PB10	INOUT	OUTPUT
273			INPUT
272			CTRL
271	PB9	INOUT	OUTPUT
270			INPUT
269			CTRL
268	PB8	INOUT	OUTPUT
267			INPUT
266			CTRL
265	PB7	INOUT	OUTPUT
264			INPUT
263			CTRL
262	PB6	INOUT	OUTPUT
261			INPUT
260			CTRL
259	PB5	INOUT	OUTPUT
258			INPUT
257			CTRL
256	PB4	INOUT	OUTPUT
255			INPUT
254			CTRL
253	PB3	INOUT	OUTPUT
252			INPUT
251			CTRL
250	PB2/MPI_NUB	INOUT	OUTPUT
249			INPUT
248			CTRL

Table 19. JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
247	PB1/MPI_NLB	INOUT	OUTPUT
246			INPUT
245			CTRL
244	PB0/MPI_NOE	INOUT	OUTPUT
243			INPUT
242			CTRL
241	MPI_D15	INOUT	OUTPUT
240			INPUT
239	MPI_D14	INOUT	OUTPUT
238			INPUT
237	MPI_D13	INOUT	OUTPUT
236			INPUT
235	MPI_D12	INOUT	OUTPUT
234			INPUT
233	MPI_D11	INOUT	OUTPUT
232			INPUT
231	MPI_D10	INOUT	OUTPUT
230			INPUT
229	MPI_D9	INOUT	OUTPUT
228			INPUT
227	MPI_D8	INOUT	OUTPUT
226			INPUT
225	MPI_D[15:8]	INOUT	CTRL
224	MPI_D7	INOUT	OUTPUT
223			INPUT
222	MPI_D6	INOUT	OUTPUT
221			INPUT
220	MPI_D5	INOUT	OUTPUT
219			INPUT
218	MPI_D4	INOUT	OUTPUT
217			INPUT
216	MPI_D3	INOUT	OUTPUT
215			INPUT
214	MPI_D2	INOUT	OUTPUT
213			INPUT

Table 19. JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
212	MPI_D1	INOUT	OUTPUT
211			INPUT
210	MPI_D0	INOUT	OUTPUT
209			INPUT
208	MPI_D[7:0]	INOUT	CTRL
207	MPI_BG	OUTPUT	OUTPUT
206			CTRL
205	MPI_BR	INPUT	INPUT
204	MPI_RNW	INPUT	INPUT
203	MPI_NCS	INPUT	INPUT
202	MPI_A9	INPUT	INPUT
201	MPI_A8	INPUT	INPUT
200	MPI_A7	INPUT	INPUT
199	MPI_A6	INPUT	INPUT
198	MPI_A5	INPUT	INPUT
197	MPI_A4	INPUT	INPUT
196	MPI_A3	INPUT	INPUT
195	MPI_A2	INPUT	INPUT
194	MPI_A1	INPUT	INPUT
193	PA29NPCS3	INOUT	OUTPUT
192			INPUT
191			CTRL
190	PA28NPCS2	INOUT	OUTPUT
189			INPUT
188			CTRL
187	PA27NPCS1	INOUT	OUTPUT
186			INPUT
185			CTRL
184	PA26NPCS0	INOUT	OUTPUT
183			INPUT
182			CTRL
181	PA25MOSI	INOUT	OUTPUT
180			INPUT
179			CTRL

Table 19. JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
178	PA24MISO	INOUT	OUTPUT
177			INPUT
176			CTRL
175	PA23SPCK	INOUT	OUTPUT
174			INPUT
173			CTRL
172	PA22RXD2	INOUT	OUTPUT
171			INPUT
170			CTRL
169	PA21TXD2	INOUT	OUTPUT
168			INPUT
167			CTRL
166	PA20SCK2	INOUT	OUTPUT
165			INPUT
164			CTRL
163	PA19RXD1	INOUT	OUTPUT
162			INPUT
161			CTRL
160	PA18/TXD1/NTRI	INOUT	OUTPUT
159			INPUT
158			CTRL
157	PA17/SCK1	INOUT	OUTPUT
156			INPUT
155			CTRL
154	PA16/RXD0	INOUT	OUTPUT
153			INPUT
152			CTRL
151	PA15/TXD0	INOUT	OUTPUT
150			INPUT
149			CTRL
148	PA14/SCK0	INOUT	OUTPUT
147			INPUT
146			CTRL

Table 19. JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
145	PA13/FIQ	INOUT	OUTPUT
144			INPUT
143			CTRL
142	PA12/IRQ3	INOUT	OUTPUT
141			INPUT
140			CTRL
139	PA11/IRQ2	INOUT	OUTPUT
138			INPUT
137			CTRL
136	PA10/IRQ1	INOUT	OUTPUT
135			INPUT
134			CTRL
133	PA9/IRQ0	INOUT	OUTPUT
132			INPUT
131			CTRL
130	PA8/TIOB5	INOUT	OUTPUT
129			INPUT
128			CTRL
127	PA7/TIOA5	INOUT	OUTPUT
126			INPUT
125			CTRL
124	PA6/TCLK5	INOUT	OUTPUT
123			INPUT
122			CTRL
121	PA5/TIOB4	INOUT	OUTPUT
120			INPUT
119			CTRL
118	PA4/TIOA4	INOUT	OUTPUT
117			INPUT
116			CTRL
115	PA3/TCLK4	INOUT	OUTPUT
114			INPUT
113			CTRL

Table 19. JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
112	PA2/TIOB3	INOUT	OUTPUT
111			INPUT
110			CTRL
109	PA1/TIOA3	INOUT	OUTPUT
108			INPUT
107			CTRL
106	PA0/TCLK3	INOUT	OUTPUT
105			INPUT
104			CTRL
103	PB27/TIOB2	INOUT	OUTPUT
102			INPUT
101			CTRL
100	PB26/TIOA2	INOUT	OUTPUT
99			INPUT
98			CTRL
97	PB25/TCLK2	INOUT	OUTPUT
96			INPUT
95			CTRL
94	PB24/TIOB1	INOUT	OUTPUT
93			INPUT
92			CTRL
91	PB23/TIOA1	INOUT	OUTPUT
90			INPUT
89			CTRL
88	PB22/TCLK1	INOUT	OUTPUT
87			INPUT
86			CTRL
85	PB21/TIOB0	INOUT	OUTPUT
84			INPUT
83			CTRL
82	PB20/TIOA0	INOUT	OUTPUT
81			INPUT
80			CTRL

Table 19. JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
79	PB19/TCLK0	INOUT	OUTPUT
78			INPUT
77			CTRL
76	D15	INOUT	INPUT
75			OUTPUT
74	D14	INOUT	INPUT
73			OUTPUT
72	D13	INOUT	INPUT
71			OUTPUT
70	D12	INOUT	INPUT
69			OUTPUT
68	D11	INOUT	INPUT
67			OUTPUT
66	D10	INOUT	INPUT
65			OUTPUT
64	D9	INOUT	INPUT
63			OUTPUT
62	D8	INOUT	INPUT
61			OUTPUT
60	D[15:8]	INOUT	CTRL
59	D7	INOUT	INPUT
58			OUTPUT
57	D6	INOUT	INPUT
56			OUTPUT
55	D5	INOUT	INPUT
54			OUTPUT
53	D4	INOUT	INPUT
52			OUTPUT
51	D3	INOUT	INPUT
50			OUTPUT
49	D2	INOUT	INPUT
48			OUTPUT
47	D1	INOUT	INPUT
46			OUTPUT

Table 19. JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
45	D0	INOUT	INPUT
44			OUTPUT
43	D[7:0]	INOUT	CTRL
42	A23/CS4	OUTPUT	OUTPUT
41			CTRL
40	A22/CS5	OUTPUT	OUTPUT
39			CTRL
38	A21/CS6	OUTPUT	OUTPUT
37			CTRL
36	A20/CS7	OUTPUT	OUTPUT
35			CTRL
34	A19	OUTPUT	OUTPUT
33	A18	OUTPUT	OUTPUT
32	A17	OUTPUT	OUTPUT
31	A16	OUTPUT	OUTPUT
30	A[19:16]	OUTPUT	CTRL
29	A15	OUTPUT	OUTPUT
28	A14	OUTPUT	OUTPUT
27	A13	OUTPUT	OUTPUT
26	A12	OUTPUT	OUTPUT
25	A11	OUTPUT	OUTPUT
24	A10	OUTPUT	OUTPUT
23	A9	OUTPUT	OUTPUT
22	A8	OUTPUT	OUTPUT
21	A[15:8]	OUTPUT	CTRL
20	A7	OUTPUT	OUTPUT
19	A6	OUTPUT	OUTPUT
18	A5	OUTPUT	OUTPUT
17	A4	OUTPUT	OUTPUT
16	A3	OUTPUT	OUTPUT
15	A2	OUTPUT	OUTPUT
14	A1	OUTPUT	OUTPUT
13	NLB/A0	OUTPUT	OUTPUT
12	A[7:0]	OUTPUT	CTRL
11	NCS3	OUTPUT	OUTPUT

Table 19. JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
10	NCS2	OUTPUT	OUTPUT
9	NCS1	OUTPUT	OUTPUT
8	NCS0	OUTPUT	OUTPUT
7	NUB/NWR1	INOUT	OUTPUT
6			INPUT
5	NWE/NWR0	INOUT	OUTPUT
4			INPUT
3	NOE/NRD	INOUT	OUTPUT
2			INPUT
1	NCS[3:0] NUB/NWR1 NWE/NWR0 NOE/NRD	INOUT	CTRL

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 436-4228
(408) 487-2610
FAX (408) 487-2600

Europe
Atmel U.K., Ltd.
Coliseum Business Centre
Riverside Way
Camberley, Surrey GU15 3YL
England
TEL (44) 1276-686677
FAX (44) 1276-686697

Asia
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road
Tsimshatsui East
Kowloon, Hong Kong
TEL (852) 27219778
FAX (852) 27221369

Japan
Atmel Japan K.K.
Tonetsu Shinkawa Bldg., 9F
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Atmel Colorado Springs
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Rousset
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4 42 53 60 00
FAX (33) 4 42 53 60 01

Fax-on-Demand
North America:
1-(800) 292-8635
International:
1-(408) 441-0732

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

BBS
1-(408) 436-4309

1028A–11/99/0M