Customizable Microcontroller

AT91CAP7E

Preliminary

Features

- Incorporates the ARM7TDMI® ARM® Thumb® Processor
  - 72 MIPS at 80MHz
  - EmbeddedICE™ In-circuit Emulation, Debug Communication Channel Support
- Additional Embedded Memories
  - One 256 Kbyte Internal ROM, Single-cycle Access at Maximum Matrix Speed
  - 160 Kbytes of Internal SRAM, Single-cycle Access at Maximum Processor or Matrix Speed (Configured in blocks of 96 KB and 64 KB with separate AHB slaves)
- External Bus Interface (EBI)
  - Supports SDRAM, Static Memory, NAND Flash/SmartMedia® and CompactFlash®
- USB 2.0 Full Speed (12 Mbits per second) Device Port
  - On-chip Transceiver, 2,432-byte Configurable Integrated DPRAM
- FPGA Interface
  - High Connectivity for up to 2 AHB Masters and 4 dedicated/16 muxed Slaves
- 10-bit Analog to Digital Converter (ADC)
  - Up to 8 multiplexed channels
  - 440 kSample / s
- Bus Matrix
  - Four-layer, 32-bit Matrix
- Fully-featured System Controller, including
  - Reset Controller, Shut Down Controller
  - Twenty 32-bit Battery Backup Registers for a Total of 80 Bytes
  - Clock Generator
  - Advanced Power Management Controller (APMC)
  - Advanced Interrupt Controller and Debug Unit
  - Periodic Interval Timer, Watchdog Timer and Real-Time Timer
- Boot Mode Select Option and Remap Command
- Reset Controller
  - Based on Two Power-on Reset Cells, Reset Source Identification and Reset Output Control
- Shut Down Controller
  - Programmable Shutdown Pin Control and Wake-up Circuitry
- Clock Generator (CKGR)
  - 32768Hz Low-power Oscillator on Battery Backup Power Supply, Providing a Permanent Slow Clock
  - Internal 32kHz RC oscillator for fast start-up
  - 8 to 16 MHz On-chip Oscillator, 50 to 100 MHz PLL, and 80 to 240 MHz PLL
- Advanced Power Management Controller (APMC)
  - Very Slow Clock Operating Mode, Software Programmable Power Optimization Capabilities
  - Four Programmable External Clock Output Signals
- Advanced Interrupt Controller (AIC)
  - Individually Maskable, Eight-level Priority, Vectored Interrupt Sources
  - Two External Interrupt Sources and one Fast Interrupt Source, Spurious interrupt protected
- Debug Unit (DBGU)
  - 2-wire UART and Support for Debug Communication Channel, Programmable ICE Access Prevention

- **Periodic Interval Timer (PIT)**
  - **20-bit interval Timer plus 12-bit interval Counter**
- **Watchdog Timer (WDT)**
  - **Key-protected, Programmable Only Once, Windowed 16-bit Counter Running at Slow Clock**
- **Real-Time Timer (RTT)**
  - **32-bit Free-running Backup Counter Running at Slow Clock with 16-bit Prescaler**
- **Two 32-bit Parallel Input/Output Controllers (PIOA and PIOB)**
  - **32 Programmable I/O Lines Multiplexed with up to Two Peripheral I/Os each**
  - **Input Change Interrupt Capability on Each I/O Line**
  - **Individually Programmable Open-drain, Pull-up Resistor, Bus Holder and Synchronous Output**
- **22 Peripheral DMA Controller Channels (PDC)**
- **Two Universal Synchronous/Asynchronous Receiver Transmitters (USART)**
  - **Individual Baud Rate Generator, IrDA® Infrared Modulation/Demodulation, Manchester Encoding/Decoding**
- **Master/Slave Serial Peripheral Interface (SPI)**
  - **8- to 16-bit Programmable Data Length, External Peripheral Chip Select**
  - **Synchronous Communications at up to 80Mbits/sec**
- **One Three-channel 16-bit Timer/Counters (TC)**
  - **Three External Clock Inputs, Two multi-purpose I/O Pins per Channel**
  - **Double PWM Generation, Capture/Waveform Mode, Up/Down Capability**
- **IEEE 1149.1 JTAG Boundary Scan on All Digital Pins**
- **Required Power Supplies:**
- **1.08V to 1.32V for VDDCORE and VDDBU**
- **1.08V to 1.32V for VDDOSC, VDDOSC32, and VDDPLLB**
- **3.0V to 3.6V for VDDPLLA and VDDIO**
- **3.0V to 3.6V for AVDD (ADC)**
- **Package Options: 144 LQFP, 176 LQFP, 208 PQFP, 144 LFBGA, 176TFBGA, 208 TFBGA, 225 LFBGA**

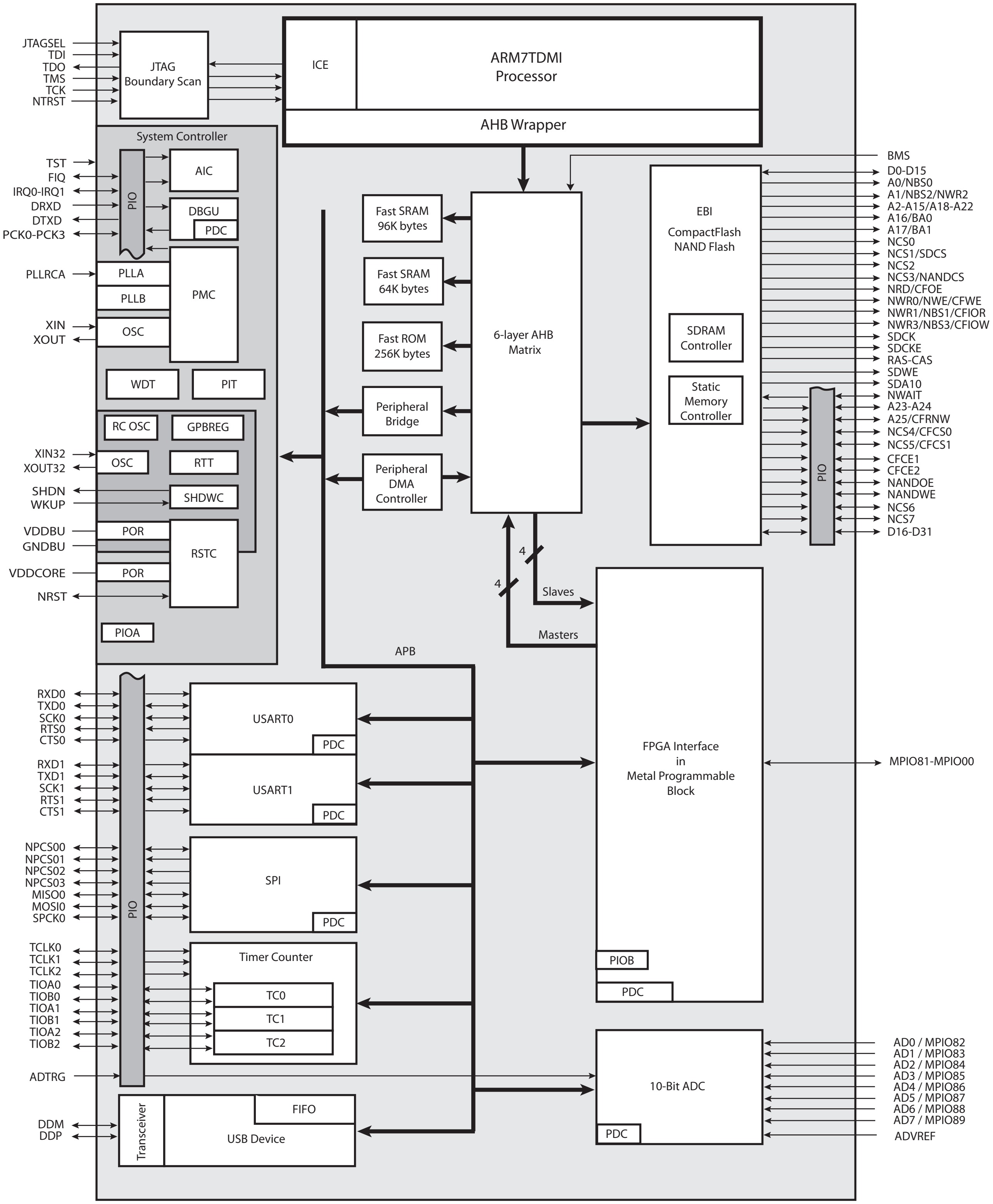
1. Description

The AT91CAP7E semi-custom System on a Chip (SoC) is a microcontroller with a special interface that allows logic in an external FPGA to be mapped directly onto its internal Amba High-speed Bus (AHB). This FPGA interface includes multiple master and slave channels providing much greater bus bandwidth for data passing between the microcontroller and an FPGA than traditional interface methods using general purpose I/O or external memory interfaces. The AT91CAP7E includes an ARM7TDMI core with the AHB, on-chip ROM, SRAM, a full-featured system controller, and various general-purpose peripherals accessible via the Amba Peripheral Bus (APB). It is implemented in a 130 nm CMOS 1.2V process and supports 3.3V I/O.

The AT91CAP7E is built upon Atmel's AT91CAP7S customizablemicrocontroller with up to 450 Kgates of metal programmable (MP) logic. The FPGA Interface is implemented inthe MP block and makes use of MP I/O's available on the AT91CAP7S giving customers not only an efficient, powerful FPGA interface on a standard microcontroller, but also an excellent platform for emulating their own AT91CAP7S-based designs.

2. Block Diagram

Figure 2-1. AT91CAP7E Block Diagram


JTAGSEL
TDI
TDO
TMS
TCK
NTRST
JTAG Boundary Scan
ICE
ARM7TDMI Processor
AHB Wrapper
System Controller
TST
FIQ
IRQ0-IRQ1
DRXD
DTXD
PCK0-PCK3
PIO
AIC
DBGU
PDC
PLLRCA
PLLA
PLLB
PMC
XIN
XOUT
OSC
WDT
PIT
RC OSC
GPBREG
XIN32
XOUT32
OSC
RTT
SHDN
WKUP
SHDWC
VDDBU
GNDBU
POR
RSTC
VDDCORE
POR
NRST
PIOA
Fast SRAM 96K bytes
Fast SRAM 64K bytes
Fast ROM 256K bytes
Peripheral Bridge
Peripheral DMA Controller
6-layer AHB Matrix
EBI
CompactFlash
NAND Flash
SDRAM Controller
Static Memory Controller
PIO
BMS
D0-D15
A0/NBS0
A1/NBS2/NWR2
A2-A15/A18-A22
A16/BA0
A17/BA1
NCS0
NCS1/SDCS
NCS2
NCS3/NANDCS
NRD/CFOE
NWR0/NWE/CFWE
NWR1/NBS1/CFIOR
NWR3/NBS3/CFIOW
SDCK
SDCKE
RAS-CAS
SDWE
SDA10
NWAIT
A23-A24
A25/CFRNW
NCS4/CFCS0
NCS5/CFCS1
CFCE1
CFCE2
NANDOE
NANDWE
NCS6
NCS7
D16-D31
4
4
Slaves
Masters
APB
RXD0
TXD0
SCK0
RTS0
CTS0
RXD1
TXD1
SCK1
RTS1
CTS1
NPCS00
NPCS01
NPCS02
NPCS03
MISO0
MOSI0
SPCK0
TCLK0
TCLK1
TCLK2
TIOA0
TIOB0
TIOA1
TIOB1
TIOA2
TIOB2
ADTRG
DDM
DDP
PIO
USART0
PDC
USART1
PDC
SPI
PDC
Timer Counter
TC0
TC1
TC2
Transceiver
FIFO
USB Device
FPGA Interface in Metal Programmable Block
PIOB
PDC
MPIO81-MPIO00
10-Bit ADC
PDC
AD0 / MPIO82
AD1 / MPIO83
AD2 / MPIO84
AD3 / MPIO85
AD4 / MPIO86
AD5 / MPIO87
AD6 / MPIO88
AD7 / MPIO89
ADVREF

3. Signal Description

Table 3-1. Signal Description by Peripheral

Signal Name	Function	Type	Active Level	Comments
Power Supplies				
VDDCORE	Core Chip Power Supply	Power		1.08V to 1.32V
VDDBU	Backup I/O Lines Power Supply	Power		1.08V to 1.32V
VDDIO	I/O Lines Power Supply	Power		3.0V to 3.6V
VDDPLLA	PLL A Power Supply	Power		3.0V to 3.6V
VDDPLLB	PLL B Power Supply	Power		1.08V to 1.32V
VDDOSC	Oscillator Power Supply	Power		1.08V to 1.32V
VDDOSC32	Oscillator Power Supply	Power		1.08V to 1.32V
AVDD	ADC Analog Power Supply	Power		3.0V to 3.6V
GND	Ground	Ground		
GNDPLLA	PLL Ground A	Ground		
GNDPLLB	PLL Ground B	Ground		
GNDOSC	Main Oscillator Ground	Ground		
GNDOSC32	32 kHz Oscillator Ground	Ground		
GNDBU	Backup Ground	Ground		
AGND	ADC Analog Ground	Ground		
Clocks, Oscillators and PLLs				
XIN	Main Oscillator Input	Input	Analog	Connect to an external crystal or drive with a 1.2V nominal square wave clock input
XOUT	Main Oscillator Output	Output	Analog	Connect to external crystal or leave unconnected
XIN32	Slow Clock Oscillator Input	Input	Analog	Must connect to a 32768Hz crystal or drive with a 1.2V, 32kHz nominal square wave input
XOUT32	Slow Clock Oscillator Output	Output	Analog	Connect to a 32768Hz crystal or leave unconnected
PLLRCA	PLL A Filter	Input	Analog	Must connect to an appropriate RC network for proper PLL operation
PCK0 - PCK3	Programmable Clock Output	Output	Clock	
Analog to Digital Converter				
AD0	ADC Input 0	An. Input	Analog	access via MPIO82 pin
AD1	ADC Input 1	An. Input	Analog	access via MPIO83 pin
AD2	ADC Input 2	An. Input	Analog	access via MPIO84 pin
AD3	ADC Input 3	An. Input	Analog	access via MPIO85 pin

Table 3-1. Signal Description by Peripheral (Continued)

Signal Name	Function	Type	Active Level	Comments
AD4	ADC Input 4	An. Input	Analog	access via MPIO86 pin
AD5	ADC Input 5	An. Input	Analog	access via MPIO87 pin
AD6	ADC Input 6	An. Input	Analog	access via MPIO88 pin
AD7	ADC Input 7	An. Input	Analog	access via MPIO89 pin
ADVREF	ADC Voltage Reference Input	An. Input	Analog	Do not leave floating - Connect to AVDD externally or another analog voltage reference up to 3.3V
ADTRG	ADC External Trigger	Dig. Input	High	Tie to AGND externally if enabled and not used - access via PIOA
Shutdown, Wake-up Logic				
SHDN	Shut-Down Control	Output	High	Driven at 0V only. Do not tie over VDDBU
WKUP	Wake-Up Input	Input		Accept between 0V and VDDBU.
ICE and JTAG				
TCK	Test Clock	Input		
TDI	Test Data In	Input		Pull-up resistor
TDO	Test Data Out	Output		
TMS	Test Mode Select	Input		Pull-up resistor
NTRST	Test Reset Signal	Input	Low	Pull-up resistor
JTAGSEL	JTAG Selection	Input	High	Pull-down resistor
Reset/Test				
NRST	Microcontroller Reset	I/O	Low	Pull-up resistor
TST	Chip Test Enable	Input	High	Pull-down resistor
BMS	Boot Mode Select	Input		Pull-up resistor 1=embedded ROM 0=EBI CS0
Debug Unit - DBGU				
DRXD	Debug Receive Data	Input		access via PIOA
DTXD	Debug Transmit Data	Output		access via PIOA
Advanced Interrupt Controller - AIC				
IRQ0 - IRQ1	External Interrupt Requests	Input	High	access via PIOA
FIQ	Fast Interrupt Request	Input	High	access via PIOA
PIO Controller - PIOA and PIOB				
PA0 - PA31	Parallel IO Controller A	I/O		Pulled-up input at reset

Table 3-1. Signal Description by Peripheral (Continued)

Signal Name	Function	Type	Active Level	Comments
PB0 - PB31	Parallel IO Controller B	I/O		access via MPIO0 - MPIO31
External Bus Interface - EBI				
D0 - D31	Data Bus	I/O		Pulled-up input at reset; access D16 - D31 via PIOA
A0 - A25	Address Bus	Output		0 at reset; access A23-A25 via PIOA
NWAIT	External Wait Signal	Input	Low	access via PIOA
Static Memory Controller - SMC				
NCS0 - NCS7	Chip Select Lines	Output	Low	access NCS4 - NCS7 via PIOA
NWR0 -NWR3	Write Signal	Output	Low	
NRD	Read Signal	Output	Low	
NWE	Write Enable	Output	Low	
NBS0 - NBS3	Byte Mask Signal	Output	Low	
CompactFlash Support				
CFCE1 - CFCE2	CompactFlash Chip Enable	Output	Low	access via PIOA
CFOE	CompactFlash Output Enable	Output	Low	
CFWE	CompactFlash Write Enable	Output	Low	
CFIOR	CompactFlash IO Read	Output	Low	
CFIOW	CompactFlash IO Write	Output	Low	
CFRNW	CompactFlash Read Not Write	Output		access via PIOA
CFCS0 - CFCS1	CompactFlash Chip Select Lines	Output	Low	access via PIOA
NAND Flash Support				
NANDCS	NAND Flash Chip Select	Output	Low	
NANDOE	NAND Flash Output Enable	Output	Low	access via PIOA
NANDWE	NAND Flash Write Enable	Output	Low	access via PIOA
SDRAM Controller				
SDCK	SDRAM Clock	Output		
SDCKE	SDRAM Clock Enable	Output	High	
SDCS	SDRAM Controller Chip Select	Output	Low	
BA0 - BA1	Bank Select	Output		
SDWE	SDRAM Write Enable	Output	Low	
RAS - CAS	Row and Column Signal	Output	Low	
SDA10	SDRAM Address 10 Line	Output		
Universal Synchronous Asynchronous Receiver Transmitter USART				
SCKx	USARTx Serial Clock	I/O		access via PIOA

Table 3-1. Signal Description by Peripheral (Continued)

Signal Name	Function	Type	Active Level	Comments
TXDx	USARTx Transmit Data	I/O		access via PIOA
RXDx	USARTx Receive Data	Input		access via PIOA
RTSx	USARTx Request To Send	Output		access via PIOA
CTSx	USARTx Clear To Send	Input		access via PIOA
Timer/Counter - TC				
TCLKx	TC Channel x External Clock Input	Input		access via PIOA
TIOAx	TC Channel x I/O Line A	I/O		access via PIOA
TIOBx	TC Channel x I/O Line B	I/O		access via PIOA
Serial Peripheral Interface - SPI				
SPIx_MISO	Master In Slave Out	I/O		access via PIOA
SPIx_MOSI	Master Out Slave In	I/O		access via PIOA
SPIx_SPCK	SPI Serial Clock	I/O		access via PIOA
SPIx_NPCS0	SPI Peripheral Chip Select 0	I/O	Low	access via PIOA
SPIx_NPCS1 - SPIx_NPCS3	SPI Peripheral Chip Select	Output	Low	access via PIOA
USB Device Port				
DDM	USB Device Port Data -	Analog		
DDP	USB Device Port Data +	Analog		
FPGA Interface- FPIF				
FPP_IRQ_ENC0 - FPP_IRQ_ENC3	FPGA Peripheral encoded interrupt requests for FPP0 thru FPP5 and FPP8 thru FPP13	I/O	High	access via PIOB/ mapped to MPIO00 thru MPIO03
FPP6_IRQ	FPP6 Interrupt Request	I/O	High	access via PIOB/ mapped to MPIO04
FPP7_IRQ	FPP7 Interrupt Request	I/O	High	access via PIOB/ mapped to MPIO05
FPP6_TX_BFFR_EMPTY	FPP6 Transmit Buffer Empty flag	I/O	High	access via PIOB/ mapped to MPIO06
FPP6_RX_BFFR_FULL	FPP6 Receive Buffer Full flag	I/O	High	access via PIOB/ mapped to MPIO07
FPP6_CHNL_TX_END	FPP6 Channel Transmit End	I/O	High	access via PIOB/ mapped to MPIO08
FPP6_CHNL_RX_END	FPP6 Channel Receive End	I/O	High	access via PIOB/ mapped to MPIO09
FPP6_TX_RDY	FPP6 Transmit Ready	I/O	High	access via PIOB/ mapped to MPIO10
FPP6_RX_RDY	FPP6 Receive Ready	I/O	High	access via PIOB/ mapped to MPIO11
FPP6_TX_SIZE0 - FPP6_TX_SIZE1	FPP6 Transfer Size	I/O		access via PIOB/ mapped to MPIO12 thru MPIO13

Table 3-1. Signal Description by Peripheral (Continued)

Signal Name	Function	Type	Active Level	Comments
FPP6_RX_SIZE0 - FPP6_RX_SIZE1	FPP6 Receive Size	I/O		access via PIOB/ mapped to MPIO14 thru MPIO15
FPP7_TX_BFFR_EMPTY	FPP7 Transmit Buffer Empty flag	I/O	High	access via PIOB/ mapped to MPIO16
FPP7_RX_BFFR_FULL	FPP7 Receive Buffer Full flag	I/O	High	access via PIOB/ mapped to MPIO17
FPP7_CHNL_TX_END	FPP7 Channel Transmit End	I/O	High	access via PIOB/ mapped to MPIO18
FPP7_CHNL_RX_END	FPP7 Channel Receive End	I/O	High	access via PIOB/ mapped to MPIO19
FPP7_TX_RDY	FPP7 Transmit Ready	I/O	High	access 20via PIOB/ mapped to MPIO20
FPP7_RX_RDY	FPP7 Receive Ready	I/O	High	access via PIOB/ mapped to MPIO21
FPP7_TX_SIZE0 - FPP7_TX_SIZE1	FPP7 Transfer Size	I/O		access via PIOB/ mapped to MPIO22 thru MPIO23
FPP7_RX_SIZE0 - FPP7_RX_SIZE1	FPP7 Receive Size	I/O		access via PIOB/ mapped to MPIO24 thru MPIO25
APB_C	APB Bridge serial control	I/O	Low	Pull-up resistor; access via PIOB/ mapped to MPIO26
APB_D0 - APB_D1	APB Bridge serial data lines	I/O	Low	Pull-up resistor; access via PIOB/ mapped to MPIO27 thru MPIO28
APB_A0 - APB_A1	APB Bridge serial address lines	I/O	Low	Pull-up resistor; access via PIOB/ mapped to MPIO29 thru MPIO30
APB_START	APB Bridge serial start	I/O	Low	Pull-up resistor; access via PIOB/ mapped to MPIO29 thru MPIO31
MA_C2 - MA_C1	Master A serial control	I/O	Low	Pull-up resistor; mapped to MPIO
MA_D0 - MA_D3	Master A serial data lines	I/O	Low	Pull-up resistor; mapped to MPIO
MA_A0 - MA_A3	Master A serial address lines	I/O	Low	Pull-up resistor; mapped to MPIO
MA_START	Master A serial start	I/O	Low	Pull-up resistor; mapped to MPIO
MB_C	Master B serial control	I/O	Low	Pull-up resistor; mapped to MPIO
MB_D0 - MB_D1	Master B serial data lines	I/O	Low	Pull-up resistor; mapped to MPIO
MB_A0 - MB_A1	Master B serial address lines	I/O	Low	Pull-up resistor; mapped to MPIO

Table 3-1. Signal Description by Peripheral (Continued)

Signal Name	Function	Type	Active Level	Comments
MB_START	Master B serial start	I/O	Low	Pull-up resistor; mapped to MPIO
SA_C2 - SA_C1	Slave A serial control - single mode	I/O	Low	Pull-up resistor; mapped to MPIO
SA_D0 - SA_D3	Slave A serial data lines	I/O	Low	Pull-up resistor; mapped to MPIO
SA_A0 - SA_A3	Slave A serial address lines	I/O	Low	Pull-up resistor; mapped to MPIO
SA_START	Slave A serial start	I/O	Low	Pull-up resistor; mapped to MPIO
SB_C	Slave B serial control	I/O	Low	Pull-up resistor; mapped to MPIO
SB_D0 - SB_D1	Slave B serial data lines	I/O	Low	Pull-up resistor; mapped to MPIO
SB_A0 - SB_A1	Slave B serial address lines	I/O	Low	Pull-up resistor; mapped to MPIO
SB_START	Slave B serial start	I/O	Low	Pull-up resistor; mapped to MPIO
SC_C2 - SC_C1	Slave C serial control - single mode	I/O	Low	Pull-up resistor; mapped to MPIO
SC_D0 - SC_D3	Slave C serial data lines	I/O	Low	Pull-up resistor; mapped to MPIO
SC_A0 - SC_A3	Slave C serial address lines	I/O	Low	Pull-up resistor; mapped to MPIO
SC_START	Slave C serial start	I/O	Low	Pull-up resistor; mapped to MPIO
SD_C	Slave D serial control	I/O	Low	Pull-up resistor; mapped to MPIO
SD_D0 - SB_D1	Slave D serial data lines	I/O	Low	Pull-up resistor; mapped to MPIO
SD_A0 - SB_A1	Slave D serial address lines	I/O	Low	Pull-up resistor; mapped to MPIO
SD_START	Slave D serial start	I/O	Low	Pull-up resistor; mapped to MPIO
SZBT_C2 - SZBT_C1	Slave ZBT RAM serial control	I/O	Low	Pull-up resistor; mapped to MPIO
SZBT_D0 - SZBT_D3	Slave ZBT RAM serial data lines	I/O	Low	Pull-up resistor; mapped to MPIO
SZBT_A0 - SZBT_A3	Slave ZBT RAM serial address lines	I/O	Low	Pull-up resistor; mapped to MPIO
SZBT_START	Slave ZBT RAM serial start	I/O	Low	Pull-up resistor; mapped to MPIO

Table 3-1. Signal Description by Peripheral (Continued)

Signal Name	Function	Type	Active Level	Comments
FPIF_SCLK	FPIF Serial Clock	Input		mapped to MPIO
FPIF_SCLK_FEEDBK	FPIF Serial Clock Feedback	Output		mapped to MPIO
FPIF_RESETN	FPIF Reset	Input	Low	Pull-up resistor; mapped to MPIO

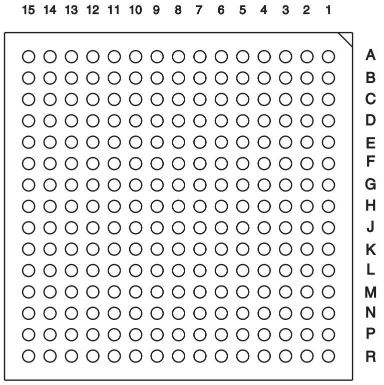
4. Package and Pinout

The AT91CAP7E is available in a RoHS-compliant 225-ball LFBGA 13x13x1.4mm, 0.8 mm ball pitch.

4.1 Mechanical Overview of the 225-ball LFBGA Package

Figure 4-1 shows the orientation of the 225-ball LFBGA Package. A detailed mechanical description is given in the Mechanical Characteristics section of the product datasheet.

Figure 4-1. 225-ball LFBGA Pinout (Bottom View)



4.2 225-ball LFBGA Package Pinout

Warning: This package pinout is preliminary and is subject to change.

Table 4-1. AT91CAP7E Pinout for 225-ball LFBGA Package

Pin	Signal Name
A1	MPIO81
A2	PA9
A3	PA8
A4	MPIO45
A5	MPIO25
A6	PA4
A7	MPIO13
A8	MPIO23
A9	MPIO20
A10	MPIO43
A11	MPIO41
A12	MPIO40
A13	MPIO03
A14	MPIO76
A15	A18
B1	A6
B2	MPIO49
B3	MPIO48
D13	MPIO01
D14	MPIO75
D15	MPIO34
E1	A3
E2	A4
E3	MPIO80
E4	MPIO56
E5	BMS
E6	PA10
E7	NCS2
E8	MPIO09
E9	MPIO08
E10	MPIO05
E11	MPIO39
E12	MPIO00
E13	MPIO35
E14	MPIO32
E15	SDA10
H10	VDDC
H11	D5
H12	PA3
H13	PA2
H14	A9
H15	A10
J1	D7
J2	D6
J3	MPIO31
J4	D8
J5	DDP
J6	D2
J7	GND
J8	GND
J9	GND
J10	A12
J11	MPIO17
J12	PA0
M7	PA22
M8	MPIO89/AD7
M9	PA14
M10	MPIO70
M11	GNDPLLA
M12	TDO
M13	TDI
M14	PA28
M15	NWR0
N1	MPIO61
N2	MPIO64
N3	VDDBU
N4	XOUT32
N5	MPIO85/AD3
N6	AVDD
N7	PA20
N8	PA13
N9	MPIO67

Table 4-1. AT91CAP7E Pinout for 225-ball LFBGA Package (Continued)

Pin	Signal Name
B4	MPIO46
B5	PA5
B6	MPIO24
B7	MPIO15
B8	MPIO11
B9	MPIO22
B10	MPIO44
B11	MPIO06
B12	MPIO04
B13	MPIO37
B14	MPIO74
B15	A20
C1	MPIO52
C2	NCS3
C3	MPIO50
C4	MPIO79
C5	PA7
C6	MPIO27
C7	PA6
C8	MPIO12
C9	MPIO21
C10	MPIO07
C11	MPIO38
C12	MPIO78
C13	A22
C14	A21
C15	A17
D1	MPIO54
D2	A5
D3	A7
D4	NCS0
D5	MPIO51
D6	MPIO47
D7	NWR3
D8	MPIO14
D9	MPIO10
F1	SDWE
F2	A2
F3	MPIO55
F4	SDRAMCKE
F5	MPIO53
F6	A0
F7	VDDIO
F8	MPIO26
F9	VDDIO
F10	A19
F11	MPIO36
F12	MPIO33
F13	A14
F14	A16
F15	A15
G1	MPIO28
G2	SDRAMCLK
G3	A1
G4	D14
G5	D15
G6	VDDC
G7	GND
G8	GND
G9	GND
G10	VDDIO
G11	RAS
G12	N/C
G13	A11
G14	CAS
G15	A13
H1	D10
H2	D9
H3	D13
H4	D11
H5	D12
H6	VDDIO
J13	PA1
J14	MPIO19
J15	A8
K1	MPIO29
K2	MPIO30
K3	MPIO60
K4	MPIO59
K5	MPIO62
K6	WKUP
K7	VDDIO
K8	VDDC
K9	VDDIO
K10	XOUT
K11	PA25
K12	TMS
K13	PA24
K14	MPIO16
K15	MPIO18
L1	MPIO57
L2	MPIO58
L3	D1
L4	MPIO65
L5	VDDOSC32
L6	GNDBU
L7	MPIO86/AD4
L8	NCS1
L9	PA17
L10	GNDPLLB
L11	PA31
L12	NTRST
L13	MPIO73
L14	PA30
L15	PA18
M1	DDM
M2	MPIO63
M3	D0
N10	NRD
N11	PLLRCA
N12	XIN
N13	VDDPLLA
N14	PA29
N15	NRST
P1	D4
P2	D3
P3	SHDN
P4	TST
P5	MPIO82/AD0
P6	MPIO87/AD5
P7	PA21
P8	PA16
P9	PA11
P10	MPIO68
P11	GNDOSC
P12	NWR1
P13	VDDOSC
P14	TCK
P15	PA27
R1	JTAGSEL
R2	ADVREF
R3	MPIO84/AD2
R4	MPIO88/AD6
R5	AGND
R6	PA23
R7	PA19
R8	PA15
R9	PA12
R10	MPIO66
R11	MPIO69
R12	MPIO71
R13	MPIO72
R14	VDDPLLB
R15	PA26

Table 4-1. AT91CAP7E Pinout for 225-ball LFBGA Package (Continued)

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
D10	MPIO42	H7	GND	M4	XIN32		
D11	MPIO77	H8	GND	M5	GNDOSC32		
D12	MPIO02	H9	GND	M6	MPIO83/AD1		

5. Power Considerations

5.1 Power Supplies

The AT91CAP7E has several types of power supply pins:

- VDDCORE pins: Power the core, including the processor, the embedded memories and the peripherals; voltage ranges from 1.08V and 1.32V (1.2V nominal). The associated ground pins for this supply and the VDDIO supply are the GND pins.
- VDDIO pins: Power the I/O lines; voltage ranges between 3.0V and 3.6V (3.3V nominal). The associated ground pins for this supply and the VDDCORE supply are the GND pins.
- VDDBU pin: Powers the Slow Clock oscillator and a part of the System Controller; voltage ranges from 1.08V and 1.32V, 1.2V nominal. The associated ground pin for this supply is the GNDBU pin.
- VDDPLLA pin: Powers the PLLA cell; voltage ranges from 3.0V and 3.6V (3.3V nominal). The associated ground pin for this supply is the GNDPLLA pin.
- VDDPLLB pin: Powers the PLLB cell and related internal loop filter cell; voltage ranges from 1.08V and 1.32V (1.2V nominal). The associated ground pin for this supply is the GNDPLLB pin.
- VDDOSC pins: Powers the Main Oscillator cell; voltage ranges from 1.08V and 1.32V (1.2V nominal). The associated ground pin for this supply is the GNDOSC pin.
- VDDOSC32 pins: Powers the 32 kHz cell; voltage ranges from 1.08V and 1.32V (1.2V nominal). The associated ground pin for this supply is the GNDOSC32 pin.
- AVDD pin: Powers the 10-bit Analog to Digital Converter and associated cells; voltage ranges from 3.0V and 3.6V (3.3V nominal). The associated ground pin for this supply is the AGND pin.

5.2 Power Consumption

Note: The following figures are preliminary figures based on prototype silicon. They are subject to change for the production silicon.

The AT91CAP7E consumes about 600 µA of static current on VDDCORE at typical conditions (1.2V, 25°C).

On VDDBU, the current does not exceed 30 µA at typical conditions.

For dynamic power consumption, the AT91CAP7E consumes about 0.33 mW/MHz of power or 275 µA/MHz of current on VDDCORE at typical conditions (1.2V, 25°C) and with the ARM subsystem running full-performance algorithm with on-chip memories, and no peripherals active.

6. I/O Line Considerations

6.1 JTAG Port Pins

TMS, TDI and TCK are Schmitt trigger inputs and have no pull-up resistors.

TDO and RTCK are outputs, driven at up to VDDIO, and have no pull-up resistor.

The JTAGSEL pin is used to select the JTAG boundary scan when asserted at a high level. It integrates a permanent pull-down resistor of about 15 kΩ to GNDBU, so that it can be left unconnected for normal operations.

The NTRST signal is described in the Reset Pins paragraph. All the JTAG signals are supplied with VDDIO.

6.2 Test Pin

The TST pin is used for manufacturing test purposes when asserted high. It integrates a permanent pull-down resistor of about 15 kΩ to GNDBU, so that it can be left unconnected for normal operations. Driving this line at a high level leads to unpredictable results.

This pin is supplied with VDDBU.

6.3 Reset Pins

NRST is an open-drain output integrating a non-programmable pull-up resistor. It can be driven with voltage at up to VDDIO.

NTRST is an input which allows reset of the JTAG Test Access port. It has no action on the processor.

As the product integrates power-on reset cells, which manages the processor and the JTAG reset, the NRST and NTRST pins can be left unconnected.

The NRST and NTRST pins both integrate a permanent pull-up resistor of 100 kΩ minimum to VDDIO.

The NRST signal is inserted in the Boundary Scan.

6.4 PIO Controllers

All the I/O lines which are managed by a PIO Controller integrate a programmable pull-up resistor of 100 kΩ minimum. Programming of this pull-up resistor is performed independently for each I/O line through the PIO Controllers.

After reset, all the I/O lines default as inputs with pull-up resistors enabled, except those which are multiplexed with the External Bus Interface signals that must be enabled as Peripheral at reset. This is explicitly indicated in the column “Reset State” of the PIO Controller multiplexing tables.

6.5 Shut Down Logic pins

The SHDN pin is an output only, which is driven by the Shut Down Controller only at low level. It can be tied high with an external pull-up resistor at VDDBU only.

7. Processor and Architecture

7.1 ARM7TDMI Processor

- RISC Processor Based on ARMv4T Von Neumann Architecture
  - Runs at up to 80 MHz, providing up to 72 MIPS
- Two instruction sets
  - ARM high-performance 32-bit Instruction Set
  - Thumb high code density 16-bit Instruction Set
- Three-stage pipeline architecture
  - Instruction Fetch (F)
  - Instruction Decode (D)
  - Execute (E)

7.2 Debug and Test Features

- Integrated embedded in-circuit emulator
  - Two watchpoint units
  - Test access port accessible through a JTAG protocol
  - Debug communication channel
- Debug Unit
  - Two-pin UART
  - Debug communication channel interrupt handling
  - Chip ID Register
- IEEE1149.1 JTAG Boundary-scan on all digital pins

7.3 Bus Matrix

- 6 Layers Matrix, handling requests from 6 masters
- Programmable Arbitration strategy
  - Fixed-priority Arbitration
  - Round-Robin Arbitration, either with no default master, last accessed default master or fixed default master
- Burst Management
  - Breaking with Slot Cycle Limit Support
  - Undefined Burst Length Support
- One Address Decoder provided per Master
  - Three different slaves may be assigned to each decoded memory area: one for internal boot, one for external boot, one after remap
- Boot Mode Select
  - Non-volatile Boot Memory can be internal or external
  - Selection is made by BMS pin sampled at reset
- Remap Command
  - Allows Remapping of an Internal SRAM in Place of the Boot Non-Volatile Memory
  - Allows Handling of Dynamic Exception Vectors

7.3.1 Matrix Masters

The Bus Matrix of the AT91CAP7E manages four Masters, which means that each master can perform an access concurrently with others, as long as the slave it accesses is available.

Each Master has its own decoder, which is defined specifically for each master. In order to simplify the addressing, all the masters have the same decoding. There are two independent masters available for an external FPGA.

Table 7-1. List of Bus Matrix Masters

Master 0	ARM7TDMI
Master 1	Peripheral DMA Controller
Master 2	FPGA Master A
Master 3	FPGA Master B

7.3.2 Matrix Slaves

The Bus Matrix of the AT91CAP7E manages nine Slaves. Each Slave has its own arbiter, thus allowing to program a different arbitration per Slave.

There are four independent slaves available for the FPGA Interface.

Table 7-2. List of Bus Matrix Slaves

Slave 0	Internal SRAM 96 Kbytes
Slave 1	Internal SRAM 64 Kbytes
Slave 2	Internal ROM 256 Kbytes
Slave 3	FPGA Slave A
Slave 4	FPGA Slave B
Slave 5	FPGA Slave C
Slave 6	FPGA Slave D
Slave 7	Unavailable
Slave 8	External Bus Interface
Slave 9	Peripheral Bridge

7.4 Peripheral DMA Controller

- Acting as one Matrix Master
- Allows data transfers from/to peripheral to/from any memory space without any intervention of the processor.
- Next Pointer Support, forbids strong real-time constraints on buffer management.
- 9 channels
  - Two for each USART
  - Two for the Debug Unit
  - Two for each Serial Peripheral Interface
  - One for the Analog to Digital Converter (ADC)
  - Two for peripherals implemented through the FPGA Interface

8. Memories

8.1 Embedded Memories

- 256 Kbyte Fast ROM
    - Single Cycle Access at full matrix speed
- 96 Kbyte Fast SRAM
    - Single Cycle Access at full matrix speed
- 64 Kbyte Fast SRAM
    - Single Cycle Access at full matrix speed

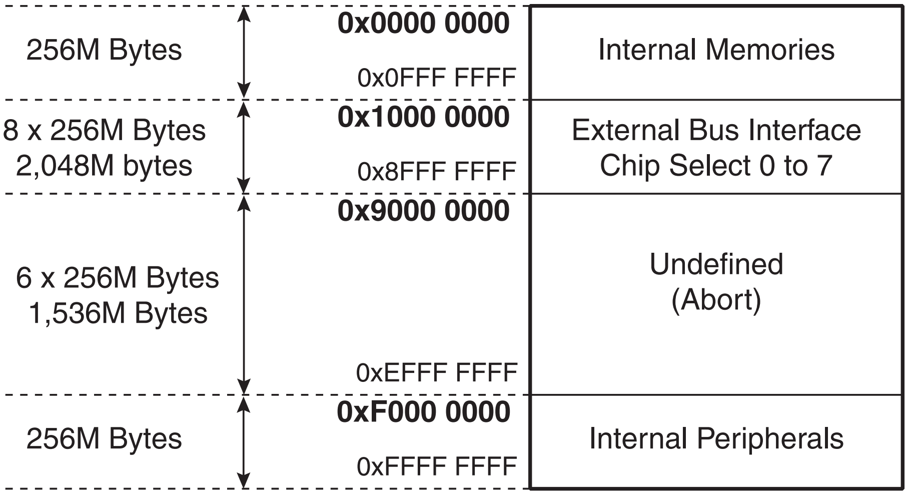
8.2 Memory Mapping

A first level of address decoding is performed by the Bus Matrix, i.e., the implementation of the Advanced High performance Bus (AHB) for its Master and Slave interfaces with additional features.

Decoding breaks up the 4G bytes of address space into 16 banks of 256M bytes. The banks 1 to 9 are directed to the EBI that associates these banks to the external chip selects NCS0 to NCS7. The bank 0 is reserved for the addressing of the internal memories, and a second level of decoding provides 1M byte of internal memory area. The bank 15 is reserved for the peripherals and provides access to the Advanced Peripheral Bus (APB).

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

Figure 8-1. AT91CAP7E Product Memory Mapping


256M Bytes
0x0000 0000
0x0FFF FFFF
Internal Memories
8 x 256M Bytes
2,048M bytes
0x1000 0000
0x8FFF FFFF
External Bus Interface
Chip Select 0 to 7
6 x 256M Bytes
1,536M Bytes
0x9000 0000
0xEFFF FFFF
Undefined
(Abort)
256M Bytes
0xF000 0000
0xFFFF FFFF
Internal Peripherals

Each Master has its own bus and its own decoder, thus allowing a different memory mapping per Master. However, in order to simplify the mappings, all the masters have a similar address decoding.

Regarding Master 0 (ARM7TDMI), two different Slaves are assigned to the memory space decoded at address 0x0: one for internal boot and one for external boot.

8.3 Internal Memory Mapping

8.3.1 Internal 160-kBytes Fast SRAM

The AT91CAP7E embeds 160-Kbytes of high-speed SRAM configured in blocks of 96 KB and 64KB. When accessed from the AHB, each SRAM block is independently single cycle accessible at full matrix speed (MCK).

8.3.2 Boot Memory

The AT91CAP7E Matrix manages a boot memory which depends on the level on the pin BMS at reset. The internal memory area mapped between address 0x0 and 0x000F FFFF is reserved at this effect.

If BMS is detected at logic 0, the boot memory is the memory connected on the Chip Select 0 of the External Bus Interface. The default configuration for the Static Memory Controller, byte select mode, 16-Bit data bus, Read/Write controlled by Chip Select, allows to boot on 16Bit non-volatile memory.

If BMS is detected at logic 1, the boot memory is the embedded ROM.

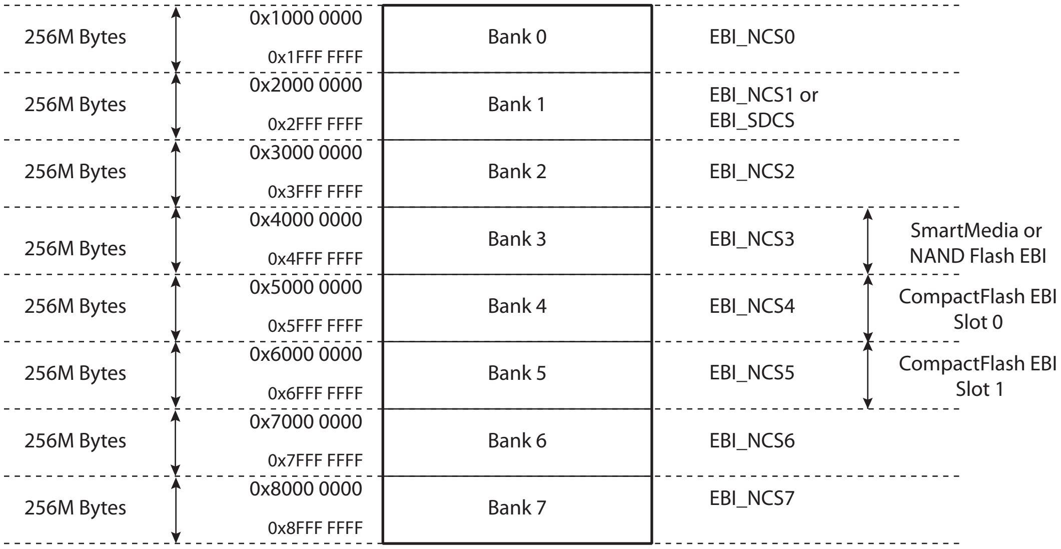
8.4 Boot Program

The internal 256 KB ROM is metal-programmable and each AT91CAP7E customer may develop their own boot program using their own code or a combination of their own code and routines available from Atmel.

8.5 External Memories Mapping

The external memories are accessed through the External Bus Interface. Each Chip Select line has a 256-MByte memory area assigned.

Figure 8-2. AT91CAP7E External Memory Mapping



256M Bytes	0x1000 0000 0x1FFF FFFF	Bank 0	EBI_NCS0	
256M Bytes	0x2000 0000 0x2FFF FFFF	Bank 1	EBI_NCS1 or EBI_SDCS	
256M Bytes	0x3000 0000 0x3FFF FFFF	Bank 2	EBI_NCS2	
256M Bytes	0x4000 0000 0x4FFF FFFF	Bank 3	EBI_NCS3	SmartMedia or NAND Flash EBI
256M Bytes	0x5000 0000 0x5FFF FFFF	Bank 4	EBI_NCS4	CompactFlash EBI Slot 0
256M Bytes	0x6000 0000 0x6FFF FFFF	Bank 5	EBI_NCS5	CompactFlash EBI Slot 1
256M Bytes	0x7000 0000 0x7FFF FFFF	Bank 6	EBI_NCS6	
256M Bytes	0x8000 0000 0x8FFF FFFF	Bank 7	EBI_NCS7	

8.6 External Bus Interface

- Optimized for Application Memory Space support

- Integrates two External Memory Controllers:
  - Static Memory Controller
  - SDRAM Controller
- Additional logic for NANDFlash and CompactFlash™
- Optional Full 32-bit External Data Bus
- Up to 26-bit Address Bus (up to 64MBytes linear per chip select)
- Up to 6 chips selects, Configurable Assignment:
  - Static Memory Controller on NCS0
  - SDRAM Controller or Static Memory Controller on NCS1
  - Static Memory Controller on NCS2
  - Static Memory Controller on NCS3, Optional NAND Flash support
  - Static Memory Controller on NCS4 - NCS5, Optional CompactFlash^{M} support

8.6.1 Static Memory Controller

- 8-, 16- or 32-bit Data Bus
- Multiple Access Modes supported
  - Byte Write or Byte Select Lines
  - Asynchronous read in Page Mode supported (4- up to 32-byte page size)
- Multiple device adaptability
  - Compliant with LCD Module
  - Control signals programmable setup, pulse and hold time for each Memory Bank
- Multiple Wait State Management
  - Programmable Wait State Generation
  - External Wait Request
  - Programmable Data Float Time
- Slow Clock mode supported

8.6.2 SDRAM Controller

- Supported devices:
  - Standard and Low Power SDRAM (Mobile SDRAM)
- Numerous configurations supported
  - 2K, 4K, 8K Row Address Memory Parts
  - SDRAM with two or four Internal Banks
  - SDRAM with 16- or 32-bit Data Path
- Programming facilities
  - Word, half-word, byte access
  - Automatic page break when Memory Boundary has been reached
  - Multi-bank Ping-pong Access
  - Timing parameters specified by software
  - Automatic refresh operation, refresh rate is programmable
- Energy-saving capabilities

  - Self-refresh, power down and deep power down modes supported
- Error detection
  - Refresh Error Interrupt
- SDRAM Power-up Initialization by software
- CAS Latency of 1, 2 and 3 supported
- Auto Precharge Command not used

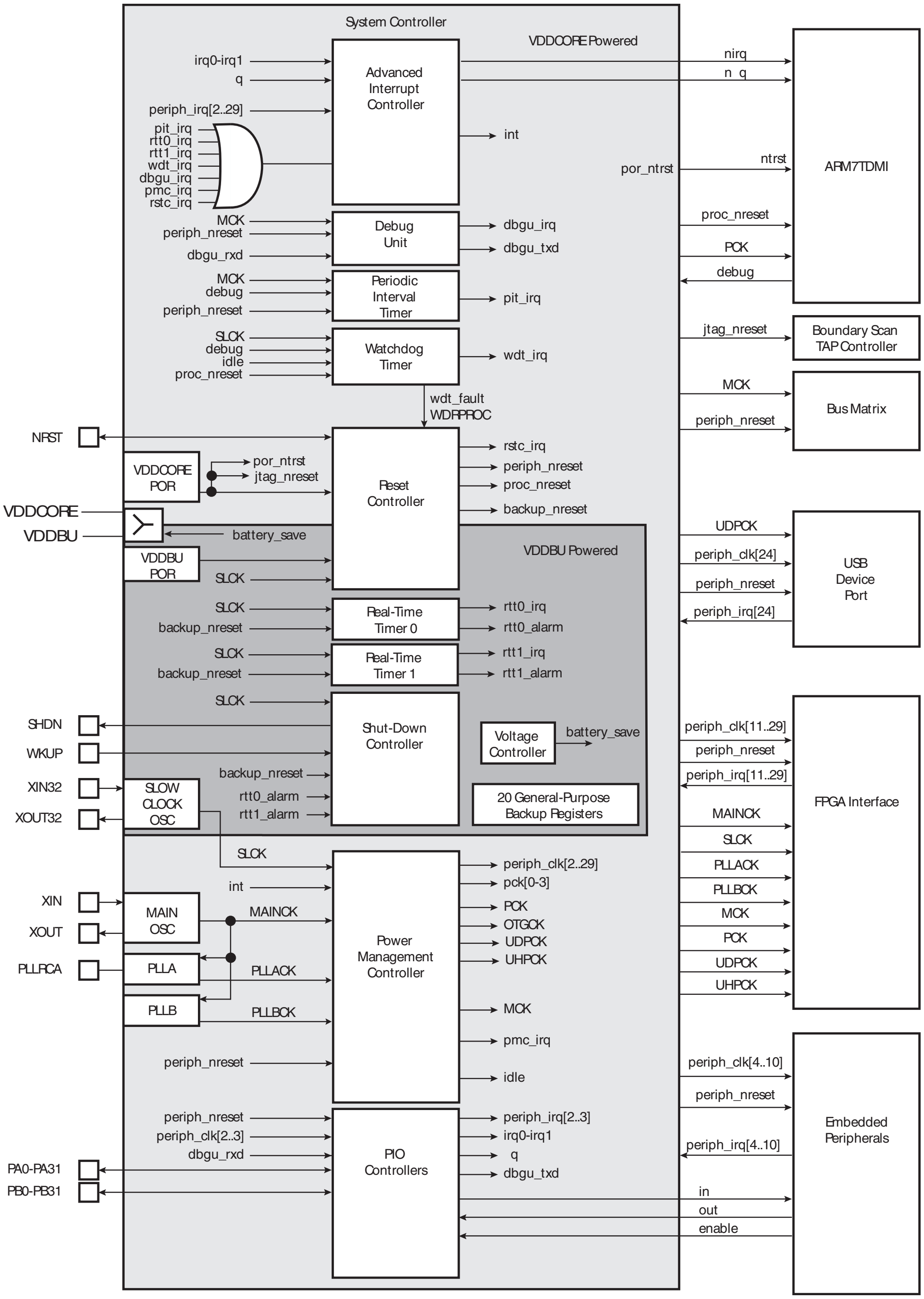
9. System Controller

The System Controller is a set of peripherals, which allow handling of key elements of the system, such as power, resets, clocks, time, interrupts, watchdog, etc.

The System Controller User Interface also includes control registers for configuring the AHB Matrix and the chip configuration. The chip configuration registers allow setting the EBI chip select assignment for external memories.

9.1 System Controller Block Diagram

Figure 9-1. AT91CAP7E System Controller Block Diagram

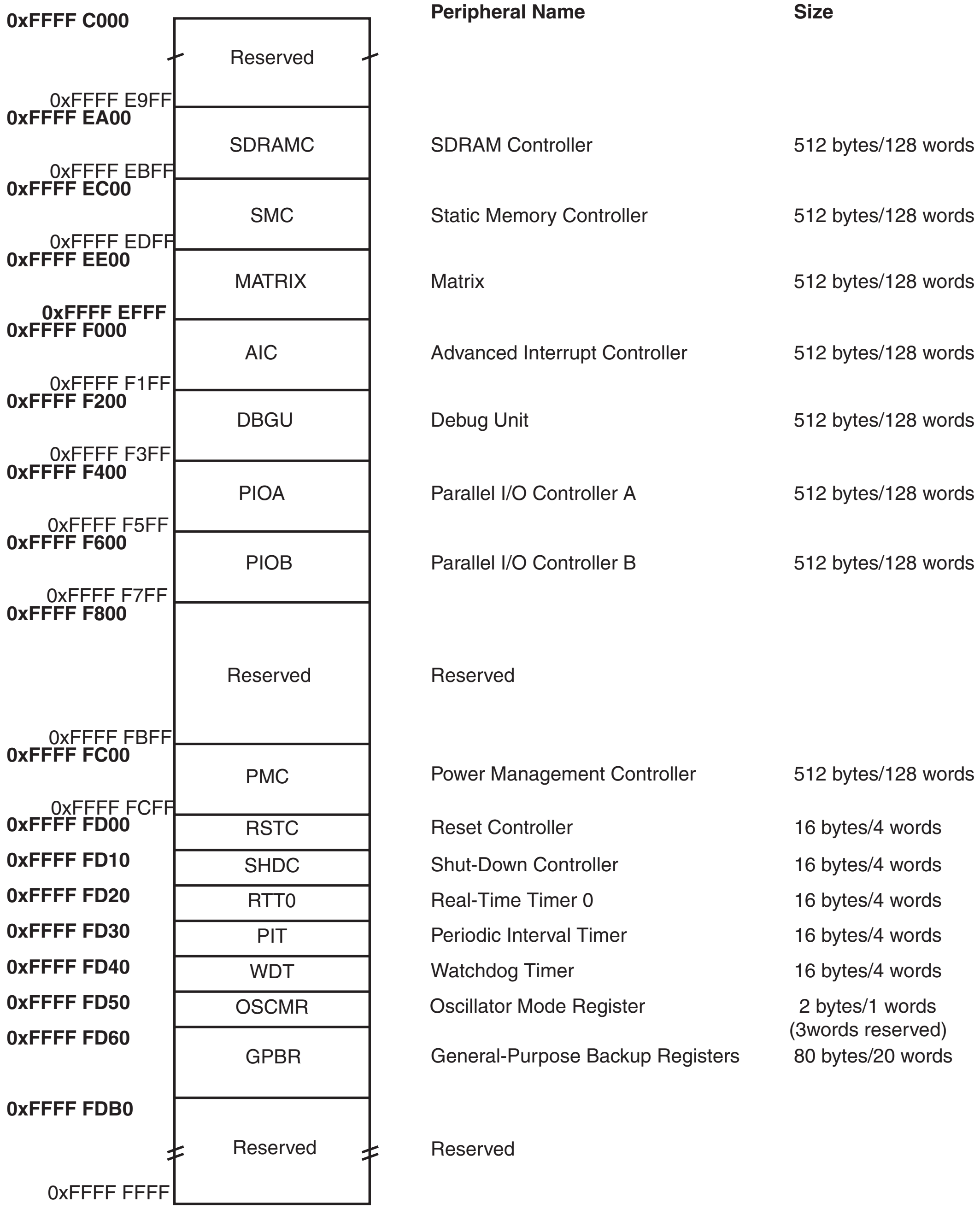

System Controller
VDDCORE Powered
irq0-irq1
fiq
periph_irq[2..29]
pit_irq
rtt0_irq
rtt1_irq
wdt_irq
dbgu_irq
pmc_irq
rstc_irq
Advanced Interrupt Controller
int
nirq
nfiq
por_ntrst
ntrst
ARM7TDMI
MCK
periph_nreset
dbgu_rxd
Debug Unit
dbgu_irq
dbgu_txd
proc_nreset
PCK
debug
MCK
debug
periph_nreset
Periodic Interval Timer
pit_irq
SLCK
debug
idle
proc_nreset
Watchdog Timer
wdt_irq
jtag_nreset
Boundary Scan TAP Controller
MCK
Bus Matrix
periph_nreset
wdt_fault
WDRPROC
NRST
VDDCORE POR
por_ntrst
jtag_nreset
Reset Controller
rstc_irq
periph_nreset
proc_nreset
backup_nreset
VDDCORE
VDDBU
battery_save
VDDBU POR
SLCK
VDDBU Powered
UDPCK
periph_clk[24]
periph_nreset
periph_irq[24]
USB Device Port
SLCK
backup_nreset
Real-Time Timer 0
rtt0_irq
rtt0_alarm
SLCK
backup_nreset
Real-Time Timer 1
rtt1_irq
rtt1_alarm
SLCK
SHDN
WKUP
Shut-Down Controller
Voltage Controller
battery_save
backup_nreset
XIN32
XOUT32
SLOW CLOCK OSC
rtt0_alarm
rtt1_alarm
20 General-Purpose Backup Registers
periph_clk[11..29]
periph_nreset
periph_irq[11..29]
MAINCK
SLCK
PLLACK
PLLBCK
MCK
PCK
UDPCK
UHPCK
FPGA Interface
SLCK
int
XIN
XOUT
MAIN OSC
MAINCK
PLLRCA
PLLA
PLLACK
PLLB
PLLBCK
Power Management Controller
periph_clk[2..29]
pck[0-3]
PCK
OTGCK
UDPCK
UHPCK
MCK
pmc_irq
idle
periph_nreset
periph_clk[4..10]
periph_nreset
periph_irq[4..10]
Embedded Peripherals
periph_nreset
periph_clk[2..3]
dbgu_rxd
PA0-PA31
PB0-PB31
PIO Controllers
periph_irq[2..3]
irq0-irq1
fiq
dbgu_txd
in
out
enable

9.2 System Controller Mapping

The System Controller's peripherals are all mapped within the highest 16K bytes of address space, between addresses 0xFFFF C000 and 0xFFFF FFFF.

However, all the registers of System Controller are mapped on the top of the address space. This allows addressing all the registers of the System Controller from a single pointer by using the standard ARM instruction set since the Load/Store instructions have an indexing mode of +/-4kbytes. Figure 9-2 shows where the User Interfaces for the System Controller peripherals fit into the memory map (relative to bus matrix and EBI (SMC, SDRAMC).

Figure 9-2. System Controller Mapping



			Peripheral Name	Size
0xFFFF C000	0xFFFF E9FF	Reserved		
0xFFFF EA00	0xFFFF EBFF	SDRAMC	SDRAM Controller	512 bytes/128 words
0xFFFF EC00	0xFFFF EDFF	SMC	Static Memory Controller	512 bytes/128 words
0xFFFF EE00	**0xFFFF EFFF**	MATRIX	Matrix	512 bytes/128 words
0xFFFF F000	0xFFFF F1FF	AIC	Advanced Interrupt Controller	512 bytes/128 words
0xFFFF F200	0xFFFF F3FF	DBGU	Debug Unit	512 bytes/128 words
0xFFFF F400	0xFFFF F5FF	PIOA	Parallel I/O Controller A	512 bytes/128 words
0xFFFF F600	0xFFFF F7FF	PIOB	Parallel I/O Controller B	512 bytes/128 words
0xFFFF F800	0xFFFF FBFF	Reserved	Reserved	
0xFFFF FC00	0xFFFF FCFF	PMC	Power Management Controller	512 bytes/128 words
0xFFFF FD00		RSTC	Reset Controller	16 bytes/4 words
0xFFFF FD10		SHDC	Shut-Down Controller	16 bytes/4 words
0xFFFF FD20		RTT0	Real-Time Timer 0	16 bytes/4 words
0xFFFF FD30		PIT	Periodic Interval Timer	16 bytes/4 words
0xFFFF FD40		WDT	Watchdog Timer	16 bytes/4 words
0xFFFF FD50		OSCMR	Oscillator Mode Register	2 bytes/1 words (3words reserved)
0xFFFF FD60		GPBR	General-Purpose Backup Registers	80 bytes/20 words
0xFFFF FDB0	0xFFFF FFFF	Reserved	Reserved	

9.3 Reset Controller

- Based on two Power-on-Reset cells
  - one on VDDBU and one on VDDCORE
- Status of the last reset
  - Either general reset (VDDBU rising), wake-up reset (VDDCORE rising), software reset, user reset or watchdog reset
- Controls the internal resets and the NRST pin output
  - Allows shaping a reset signal for the external devices

9.4 Shut Down Controller

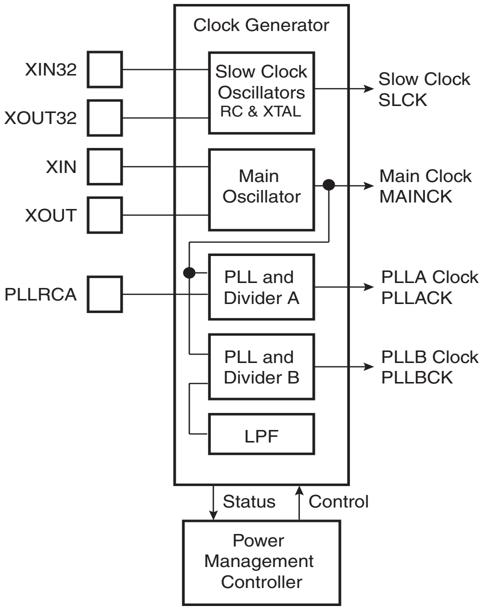
- Shut-Down and Wake-Up logic
  - Software programmable assertion of the SHDN open-drain pin
  - De-assertion Programmable on a WKUP pin level change or on alarm

9.5 Clock Generator

- Embeds the Low Power, fast start-up 32kHz RC Oscillator
  - Provides the default Slow Clock SLCK to the system
  - The SLCK is required for AT91CAP7E to start-up because it is the default clock for the ARM7TDMI at power-up.
- Embeds the Low Power 32768Hz Slow Clock Oscillator
  - Requires an external 32768Hz crystal
  - Optional Slow Clock SLCK source when a real-time timebase is required
- Embeds the Main Oscillator
  - Requires an external crystal. For systems using the USB features, 12MHz is recommended.
  - Oscillator bypass feature
  - Supports 8 to 16MHz crystals. Recommend 12 MHz crystal if using the USB features of AT91CAP7E.
  - Generates input reference clock for the two PLLs.
- Embeds PLLA primarily for generating processor and master clocks. For full-speed operation on the ARM7TDMI processor, this PLL should be programmed to generate a 160 MHz clock that must then be divided in half to generate the 80 MHz PCK and related clocks.
  - PLLA outputs an 80 to 240MHz clock
  - Requires an external RC filter network
  - PLLA has a 1MHz minimum input frequency
  - Integrates an input divider to increase output accuracy
- Embeds PLLB primarily for generating a 96 MHz clock that is divided down to generate the USB related clocks.
  - PLLB uses an internal low-pass filter (LPF) and can output a 50 to 100 MHz clock
  - PLLB and its internal low-pass filter (LPF) are tuned especially for generating a 96 MHz clock with a 12 MHz input frequency
  - 12 MHz minimum input frequency

– Integrates an input divider to increase output accuracy

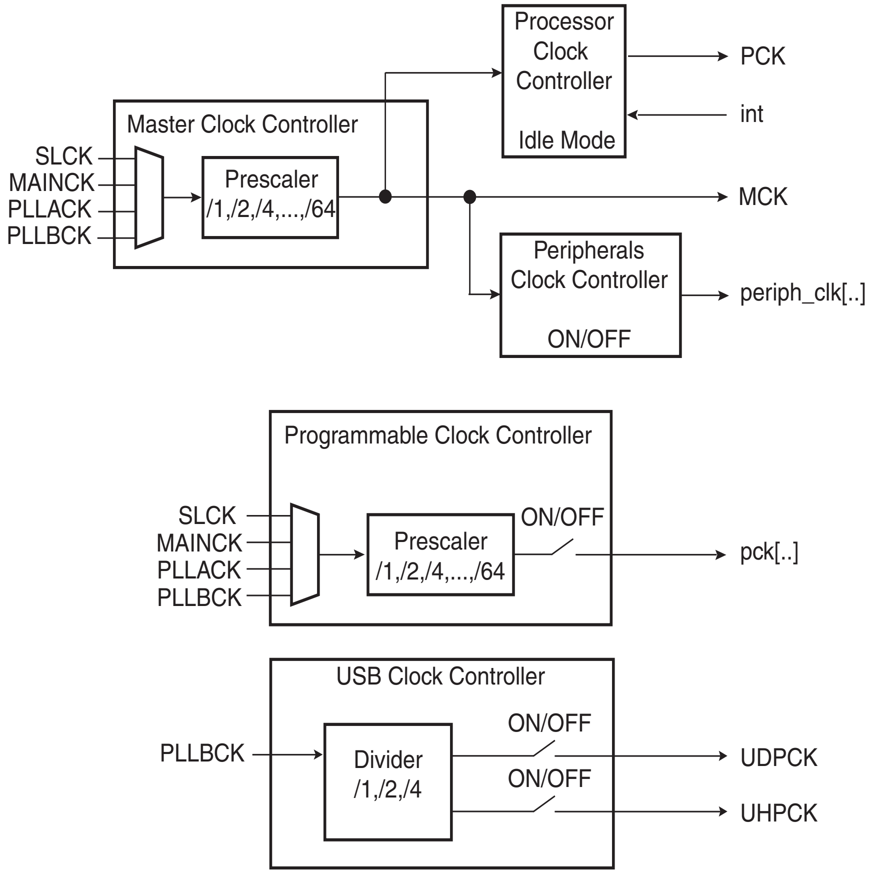
Figure 9-3. Clock Generator Block Diagram


Clock Generator
XIN32
XOUT32
XIN
XOUT
PLLRCA
Slow Clock Oscillators RC & XTAL
Main Oscillator
PLL and Divider A
PLL and Divider B
LPF
Slow Clock SLCK
Main Clock MAINCK
PLLA Clock PLLACK
PLLB Clock PLLBCK
Status
Control
Power Management Controller

9.6 Power Management Controller

- The Power Management Controller provides the following clocks as shown in Figure 7 below:
  - the Processor Clock PCK
  - the Master Clock MCK, in particular to the Matrix and the memory interfaces
  - the USB Device Clock UDPCK
  - independent peripheral clocks (periph_clk), typically at the frequency of MCK
  - four programmable clock outputs: PCK0 to PCK3
- Five flexible operating modes:
  - Normal Mode, processor and peripherals running at a programmable frequency
  - Idle Mode, processor stopped waiting for an interrupt
  - Slow Clock Mode, processor and peripherals running at low frequency
  - Standby Mode, mix of Idle and Backup Mode, peripheral running at low frequency, processor stopped waiting for an interrupt
  - Backup Mode, Main Power Supplies off, VDDBU powered by a battery

Figure 9-4. AT91CAP7E Power Management Controller Block Diagram


Processor Clock Controller
Idle Mode
PCK
int
Master Clock Controller
SLCK
MAINCK
PLLACK
PLLBCK
Prescaler /1,/2,/4,...,/64
MCK
Peripherals Clock Controller
ON/OFF
periph_clk[..]
Programmable Clock Controller
SLCK
MAINCK
PLLACK
PLLBCK
Prescaler /1,/2,/4,...,/64
ON/OFF
pck[..]
USB Clock Controller
PLLBCK
Divider /1,/2,/4
ON/OFF
UDPCK
ON/OFF
UHPCK

9.7 Periodic Interval Timer

- Includes a 20-bit Periodic Counter, with less than 1µs accuracy
- Includes a 12-bit Interval Overlay Counter
- Real Time OS or Linux/WinCE compliant tick generator

9.8 Watchdog Timer

- 16-bit key-protected only-once-Programmable Counter
- Windowed, prevents the processor to be in a dead-lock on the watchdog access

9.9 Real-Time Timer

- One Real-Time Timer, allowing backup of time
  - 32-bit Free-running, back-up Counter
  - Integrates a 16-bit programmable prescaler running on the embedded 32.768Hz oscillator
  - Alarm Register capable to generate a wake-up of the system through the Shut Down Controller

9.10 General-Purpose Backed-up Registers

- Twenty 32-bit backup general-purpose registers

9.11 Backup Power Switch

- Automatic switch of VDDBU to VDDCORE guaranteeing very low power consumption on VDDBU while VDDCORE is present

9.12 Advanced Interrupt Controller

- Controls the interrupt lines (nIRQ and nFIQ) of the ARM Processor
- Thirty-two individually maskable and vectored interrupt sources
  - Source 0 is reserved for the Fast Interrupt Input (FIQ)
  - Source 1 is reserved for system peripherals (PIT, RTT, PMC, DBGU, etc.)
  - Programmable Edge-triggered or Level-sensitive Internal Sources
  - Programmable Positive/Negative Edge-triggered or High/Low Level-sensitive
- Two External Sources plus the Fast Interrupt signal
- 8-level Priority Controller
  - Drives the Normal Interrupt of the processor
  - Handles priority of the interrupt sources 1 to 31
  - Higher priority interrupts can be served during service of lower priority interrupt
- Vectoring
  - Optimizes Interrupt Service Routine Branch and Execution
  - One 32-bit Vector Register per interrupt source
  - Interrupt Vector Register reads the corresponding current Interrupt Vector
- Protect Mode
  - Easy debugging by preventing automatic operations when protect models are enabled
- Fast Forcing
  - Permits redirecting any normal interrupt source on the Fast Interrupt of the processor

9.13 Debug Unit

- Composed of two functions
  - Two-pin UART
  - Debug Communication Channel (DCC) support
- Two-pin UART
  - Implemented features are 100% compatible with the standard Atmel USART
  - Independent receiver and transmitter with a common programmable Baud Rate Generator
  - Even, Odd, Mark or Space Parity Generation
  - Parity, Framing and Overrun Error Detection
  - Automatic Echo, Local Loopback and Remote Loopback Channel Modes
  - Support for two PDC channels with connection to receiver and transmitter

- Debug Communication Channel Support
    - – Offers visibility of and interrupt trigger from COMMRX and COMMTX signals from the ARM Processor's ICE Interface

9.14 Chip Identification

- Chip ID: 83770904 (0x1000 0011 0111 0111 0000 1001 0000 0100). This value is stored in the Chip ID Register (DBGU_CIDR) in the Debug Unit. The last 5 bits of the register are reserved for a chip version number.
- JTAG ID: unique for each CAP7 personalization.

9.15 PIO Controllers

- Two PIO Controllers (PIOA and PIOB) included.
- Each PIO Controller controls up to 32 programmable I/O Lines
    - – PIOA controls 32 I/O Lines (PA0 - PA31)
    - – PIOB can control up to 32 of the MPIO Lines
- Fully programmable through Set/Clear Registers
- For each I/O Line (whether assigned to a peripheral or used as general purpose I/O)
    - – Input change interrupt
    - – Glitch filter
    - – Multi-drive option enables driving in open drain
    - – Programmable pull up on each I/O line
    - – Pin data status register, supplies visibility of the level on the pin at any time
- Synchronous output, provides Set and Clear of several I/O lines in a single write
- PIOA has multiplexing of two peripheral functions per I/O Line (see section 10.4.1 ”PIO Controller A Multiplexing” on page 36)
- PIOB multiplexing is controlled by the FPGA Interface (see section 11.4.2 ”PIO Controller B Multiplexing” on page 47)

9.16 User Interface

9.16.1 Special System Controller Register Mapping

Table 9-1. Special System Controller Registers

Offset	Register	Name	Access	Reset Value
0x50	Oscillator Mode Register	SYSC_OSCMR	Read/Write	0x1
0x60	General Purpose Backup Register 1	SYSC_GPBR1	Read/Write	0x0
---	---	---	---	---
0xAC	General Purpose Backup Register 20	SYSC_GPBR20	Read/Write	0x0

9.16.2 Oscillator Mode Register

Register Name: SYSC_OSCMR

Access Type: Read/Write

Reset Value: 0x00000001

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
–	–	–	–	OSC32K_SEL	–	OSC32K_XT_ EN	OSC32K_RC_ EN

- **OSC32K_RC_EN: Enable internal RC oscillator**

0: No effect.

1: Enables the internal RC oscillator [enabled out of reset indicating system starts off of RC]

- **OSC32K_XT_EN: Enable external crystal oscillator**

0: No effect.

1: Enables the external crystal oscillator

- **OSC32K_SEL: Slow clock source select**

0: Selects internal RC as source of slow clock

1: Selects external crystal and source of slow

NOTE: After setting OSC32K_XT_EN bit, wait till 1.2s of on chip slow clock timing before setting OSC32K_SEL bit.

9.16.3 General Purpose Backup Register

Register Name: SYSC_GPBRx

Access Type: Read/Write

Reset Value: 0x0

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">GPBRx</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">GPBRx</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="8">GPBRx</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="8">GPBRx</td></tr>
</table>

- **GPBRx: General Purpose Backup Register**

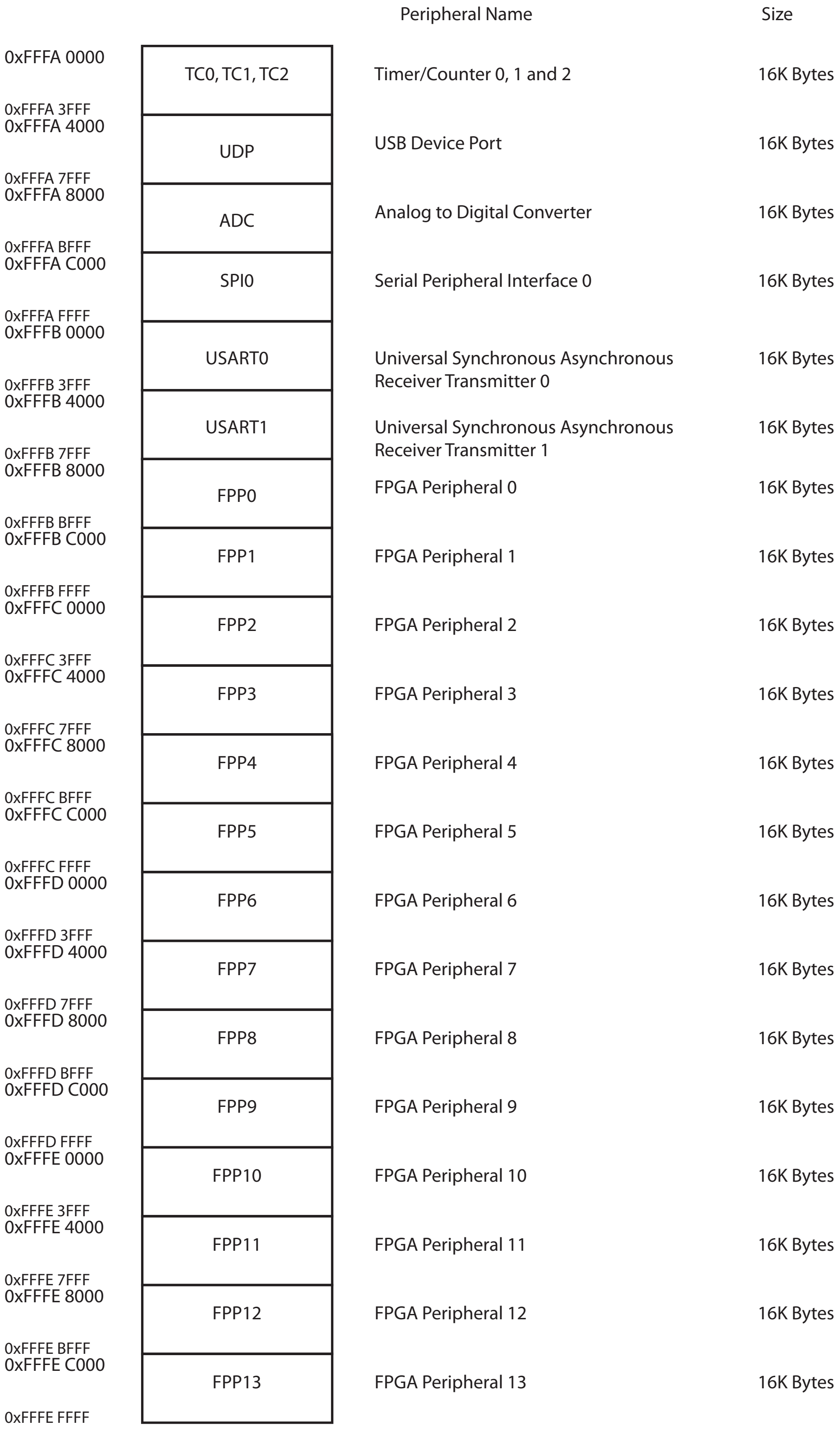
These are user programmable registers that are powered by the backup power supply (VDDBU).

10. Peripherals

10.1 Peripheral Mapping

Both the standard peripherals and any APB peripherals implemented in the MPBlock are mapped in the upper 256M bytes of the address space between the addresses 0xFFFA 0000 and 0xFFFE FFFF. Each User Peripheral is allocated 16K bytes of address space as shown below in Figure 10-1.

Figure 10-1. AT91CAP7E Peripheral Mapping



Start	End		Peripheral Name	Size
0xFFFA 0000	0xFFFA 3FFF	TC0, TC1, TC2	Timer/Counter 0, 1 and 2	16K Bytes
0xFFFA 4000	0xFFFA 7FFF	UDP	USB Device Port	16K Bytes
0xFFFA 8000	0xFFFA BFFF	ADC	Analog to Digital Converter	16K Bytes
0xFFFA C000	0xFFFA FFFF	SPI0	Serial Peripheral Interface 0	16K Bytes
0xFFFB 0000	0xFFFB 3FFF	USART0	Universal Synchronous Asynchronous Receiver Transmitter 0	16K Bytes
0xFFFB 4000	0xFFFB 7FFF	USART1	Universal Synchronous Asynchronous Receiver Transmitter 1	16K Bytes
0xFFFB 8000	0xFFFB BFFF	FPP0	FPGA Peripheral 0	16K Bytes
0xFFFB C000	0xFFFB FFFF	FPP1	FPGA Peripheral 1	16K Bytes
0xFFFC 0000	0xFFFC 3FFF	FPP2	FPGA Peripheral 2	16K Bytes
0xFFFC 4000	0xFFFC 7FFF	FPP3	FPGA Peripheral 3	16K Bytes
0xFFFC 8000	0xFFFC BFFF	FPP4	FPGA Peripheral 4	16K Bytes
0xFFFC C000	0xFFFC FFFF	FPP5	FPGA Peripheral 5	16K Bytes
0xFFFD 0000	0xFFFD 3FFF	FPP6	FPGA Peripheral 6	16K Bytes
0xFFFD 4000	0xFFFD 7FFF	FPP7	FPGA Peripheral 7	16K Bytes
0xFFFD 8000	0xFFFD BFFF	FPP8	FPGA Peripheral 8	16K Bytes
0xFFFD C000	0xFFFD FFFF	FPP9	FPGA Peripheral 9	16K Bytes
0xFFFE 0000	0xFFFE 3FFF	FPP10	FPGA Peripheral 10	16K Bytes
0xFFFE 4000	0xFFFE 7FFF	FPP11	FPGA Peripheral 11	16K Bytes
0xFFFE 8000	0xFFFE BFFF	FPP12	FPGA Peripheral 12	16K Bytes
0xFFFE C000	0xFFFE FFFF	FPP13	FPGA Peripheral 13	16K Bytes

10.2 Peripheral Identifiers

The AT91CAP7E embeds some of the most common peripherals. Additional peripherals can be readily implemented in the external FPGA by the customer, and mapped direcly on the APB. The table below defines the Peripheral Identifiers of the AT91CAP7E. A peripheral identifier is required for the control of the peripheral interrupt with the Advanced Interrupt Controller and for the control of the peripheral clock with the Power Management Controller.

Table 10-1. AT91CAP7E Peripheral Identifiers

Peripheral ID	Peripheral Mnemonic	Peripheral Name	External Interrupt
0	**AIC**	Advanced Interrupt Controller	FIQ
1	**SYSC**	System Controller	
2	**PIOA**	Parallel I/O Controller A	
3	**PIOB**	Parallel I/O Controller B	
4	**US0**	USART 0	
5	**US1**	USART 1	
6	**SPI0**	Serial Peripheral Interface 0	
7	**TC0**	Timer/Counter 0	
8	**TC1**	Timer/Counter 1	
9	**TC2**	Timer/Counter 2	
10	**UDP**	USB Device Port	
11	**ADC**	Analog to Digital Converter	
12	**FPP0**	FPGA Peripheral 0	
13	**FPP1**	FPGA Peripheral 1	
14	**FPP2**	FPGA Peripheral 2	
15	**FPP3**	FPGA Peripheral 3	
16	**FPP4**	FPGA Peripheral 4	
17	**FPP5**	FPGA Peripheral 5	
18	**FPP6**	FPGA Peripheral 6	
19	**FPP7**	FPGA Peripheral 7	
20	**FPP8**	FPGA Peripheral 8	
21	**FPP9**	FPGA Peripheral 9	
22	**FPP10**	FPGA Peripheral 10	
23	**FPP11**	FPGA Peripheral 11	
24	**FPP12**	FPGA Peripheral 12	
25	**FPP13**	FPGA Peripheral 13	
26	**FPMA**	FPGA Master A	
27	**FPMB**	FPGA Master B	
28	**N/A**	Not Available	

Table 10-1. AT91CAP7E Peripheral Identifiers (Continued)

Peripheral ID	Peripheral Mnemonic	Peripheral Name	External Interrupt
29	**N/A**	Not Available	
30	**AIC**	Advanced Interrupt Controller	IRQ0
31	**AIC**	Advanced Interrupt Controller	IRQ1

10.3 Peripheral Interrupts and Clock Control

10.3.1 System Interrupt

The System Interrupt in Source 1 is the wired-OR of the interrupt signals coming from:

- the SDRAM Controller
- the Debug Unit
- the Periodic Interval Timer
- the Real-Time Timer
- the Watchdog Timer
- the Reset Controller
- the Power Management Controller

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

10.3.2 External Interrupts

All external interrupt signals, i.e., the Fast Interrupt signal FIQ or the Interrupt signals IRQ0 to IRQ1, use a dedicated Peripheral ID. However, there is no clock control associated with these peripheral IDs.

10.3.3 Timer Counter Interrupts

The three Timer Counter channels interrupt signals are OR-wired together to provide the interrupt source 7 of the Advanced Interrupt Controller. This forces the programmer to read all Timer Counter status registers before branching the right Interrupt Service Routine.

The Timer Counter channels clocks cannot be deactivated independently. Switching off the clock of the Peripheral 7 disables the clock of the 3 channels.

10.4 Peripherals Signals Multiplexing on I/O Lines

The AT91CAP7E features two PIO controllers, PIOA which multiplexes the I/O lines of the standard peripheral set and PIOB which multiplexes the FPGA Interface through MPIO.

Each PIO Controller controls up to 32 lines. On PIOA, each line can be assigned to one of two peripheral functions, A or B. The multiplexing table in the following paragraph define how the I/O lines of the peripherals A and B are multiplexed on PIOA.

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

If a signal name is listed in the "Reset State" column, the PIO Line is assigned to this function and the corresponding bit in PIO_PSR resets high. This is the case of pins controlling memories, in particular the address lines, which require the pin to be driven as soon as the reset is released. Note that the pull-up resistor is also enabled in this case.

10.4.1 PIO Controller A Multiplexing

Table 10-2. Multiplexing on PIO Controller A

PIO Controller A			
I/O Line	**Peripheral A**	**Peripheral B**	**Reset State**
PA0	FIQ	DBG_DRXD	
PA1	NWAIT	DBG_DTXD	
PA2	NCS4/CFCS0	USART0_SCK0	
PA3	CFCE1	USART0_RTS0	
PA4	A25/CFRNW	USART0_CTS0	
PA5	NANDOE	USART0_TXD0	
PA6	NANDWE	USART0_RXD0	
PA7	NCS6	SPI_MISO	
PA8	NCS7	SPI_MOSI	
PA9	ADCTRIG	SPI_SPCK	
PA10	IRQ0	SPI_NPCS0	
PA11	IRQ1	SPI_NPCS1	
PA12	NCS5/CFCS1	SPI_NPCS2	
PA13	CFCE2	SPI_NPCS3	
PA14	A23	APMC_PCK0	
PA15	A24	APMC_PCK1	
PA16	D16	APMC_PCK2	
PA17	D17	APMC_PCK3	
PA18	D18	USART1_SCK1	
PA19	D19	USART1_RTS1	
PA20	D20	USART1_CTS1	
PA21	D21	USART1_TXD1	
PA22	D22	USART1_RXD1	
PA23	D23	TIMER0_TCLK0	
PA24	D24	TIMER1_TCLK1	
PA25	D25	TIMER2_TCLK2	
PA26	D26	TIMER0_TIOA0	
PA27	D27	TIMER0_TIOB0	
PA28	D28	TIMER1_TIOA1	

Table 10-2. Multiplexing on PIO Controller A

PIO Controller A			
I/O Line	**Peripheral A**	**Peripheral B**	**Reset State**
PA29	D29	TIMER1_TIOB1	
PA30	D30	TIMER2_TIOA2	
PA31	D31	TIMER2_TIOB2	

10.4.2 PIO Controller B Multiplexing

- The PIOB Port is part of the FPGA Interface, and its multiplexing is determined by that interface (see section 11.4.2 "PIO Controller B Multiplexing" on page 47).

10.4.3 Resource Multiplexing

10.4.3.1 EBI

If not required, the NWAIT function (external wait request) can be deactivated by soft-ware allowing this pin to be used as a PIO. Use of the NWAIT function prevents use of the Debug Unit.

10.4.3.2 32-bit Data Bus

Using a 32-bit Data Bus prevents:

- using the three Timer Counter channels' outputs and trigger inputs
- using the USART1
- using two of the clock outputs (APMC_PCK2 and APMC_PCK3)

10.4.3.3 NAND Flash Interface

Using the NAND Flash interface prevents using the NCS3 and USART0.

10.4.3.4 Compact Flash Interface

Using the CompactFlash interface prevents using the USART0.

10.4.3.5 SPI

Using the SPI prevents use of NCS6, NCS7, and the ADC external trigger.

10.4.3.6 USARTs

Using the USART0 prevents use of CompactFlash or NAND Flash.

Using the USART1 prevents using a full 32-bit bus for the EBI.

10.4.3.7 Clock Outputs

Using the clock outputs prevents use of either higher EBI address bits or a full 32-bit data bus (see table 10-2).

10.4.3.8 Interrupt Lines

Using FIQ prevents using the Debug Unit.

Using IRQ0 prevents the use of SPI_NPCS0.

Using IRQ1 prevents the use of SPI_NPCS1.

10.5 Embedded Peripherals Overview

10.5.1 Serial Peripheral Interface

- Supports communication with serial external devices
  - Four chip selects with external decoder support allow communication with up to 15 peripherals
  - Serial memories, such as DataFlash and 3-wire EEPROMs
  - Serial peripherals, such as ADCs, DACs, LCD Controllers, CAN Controllers and Sensors
  - External co-processors
- Master or slave serial peripheral bus interface
  - 8- to 16-bit programmable data length per chip select
  - Programmable phase and polarity per chip select
  - Programmable transfer delays between consecutive transfers and between clock and data per chip select
  - Programmable delay between consecutive transfers
  - Selectable mode fault detection
- Very fast transfers supported
  - Transfers with baud rates up to MCK
  - The chip select line may be left active to speed up transfers on the same device

10.5.2 USART

- Programmable Baud Rate Generator
- 5- to 9-bit full-duplex synchronous or asynchronous serial communications
  - 1, 1.5 or 2 stop bits in Asynchronous Mode or 1 or 2 stop bits in Synchronous Mode
  - Parity generation and error detection
  - Framing error detection, overrun error detection
  - MSB-first or LSB-first
  - Optional break generation and detection
  - By 8 or by-16 over-sampling receiver frequency
  - Hardware handshaking RTS-CTS
  - Receiver time-out and transmitter time-guard
  - Optional Multi-drop Mode with address generation and detection
  - Optional Manchester Encoding
- RS485 with driver control signal
- ISO7816, T = 0 or T = 1 Protocols for interfacing with smart cards
  - NACK handling, error counter with repetition and iteration limit
- IrDA modulation and demodulation
  - Communication at up to 115.2 Kbps
- Test Modes
  - Remote Loopback, Local Loopback, Automatic Echo

10.5.3 Timer Counter

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

10.5.4 USB Device Port

- USB V2.0 full-speed compliant, 12 MBits per second
- Embedded USB V2.0 full-speed transceiver
- Embedded 2,432-byte dual-port RAM for endpoints
- Suspend/Resume logic
- Ping-pong mode (two memory banks) for isochronous and bulk endpoints
- Six general-purpose endpoints
  - Endpoint 0 and 3: 64 bytes, no ping-pong mode
  - Endpoint 1 and 2: 64 bytes, ping-pong mode
  - Endpoint 4 and 5: 512 bytes, ping-pong mode

10.5.5 Analog to Digital Converter

- 10-bit Successive Approximation Register (SAR) ADC based on thermometric-resistive
- Up to 440 kSamples/sec.
- Up to 8 independent analog input channels
- Low active power: < 2 mW
- Low power stand-by mode
- External voltage reference of 2.6V to analog supply for better accuracy
- ± 2LSB Integral Non-Linearity (INL), ± 0.9 LSB Differential Non-Linearity (DNL)
- Individual enable and disable of each channel
- Multiple trigger sources:
  - Hardware or software trigger
  - External trigger pin
- Sleep Mode and conversion sequencer
  - Automatic wakeup on trigger and back to sleep mode after conversions of all enabled channels

11. FPGA Interface (FPIF)

11.1 Description

The FPGA Interface (FPIF) module provides a means to connect an external FPGA directly to the AT91CAP7E internal AHB Bus. This interface is implemented in the metal-programmable logic block (MP Block) that is provided as part of the AT91CAP7S customizable microcontroller platform. Therefore the interface is constrained to access the AHB Bus through the Masters and Slaves already pre-defined for the MP block.

- The FPGA interface uses 82 of the metal-programmable I/O pads (MPIO's) provided on the CAP7 platform, and it provides FPGA access to the following MP block features:
- 2 AHB Masters
- 4 AHB Slaves
- 1 AHB Slave to remap the ROM using an external ZBT RAM through the FPGA (For CAP7 Emulation purposes). Programmable ROM remap feature at startup.
- 14 APB's slaves
- 2 DMA full duplex channels
- Up to 13 priority encoded IRQ's
- 2 unencoded IRQ's for DMA transfers
- 32 bits PIO (Shared I/O)

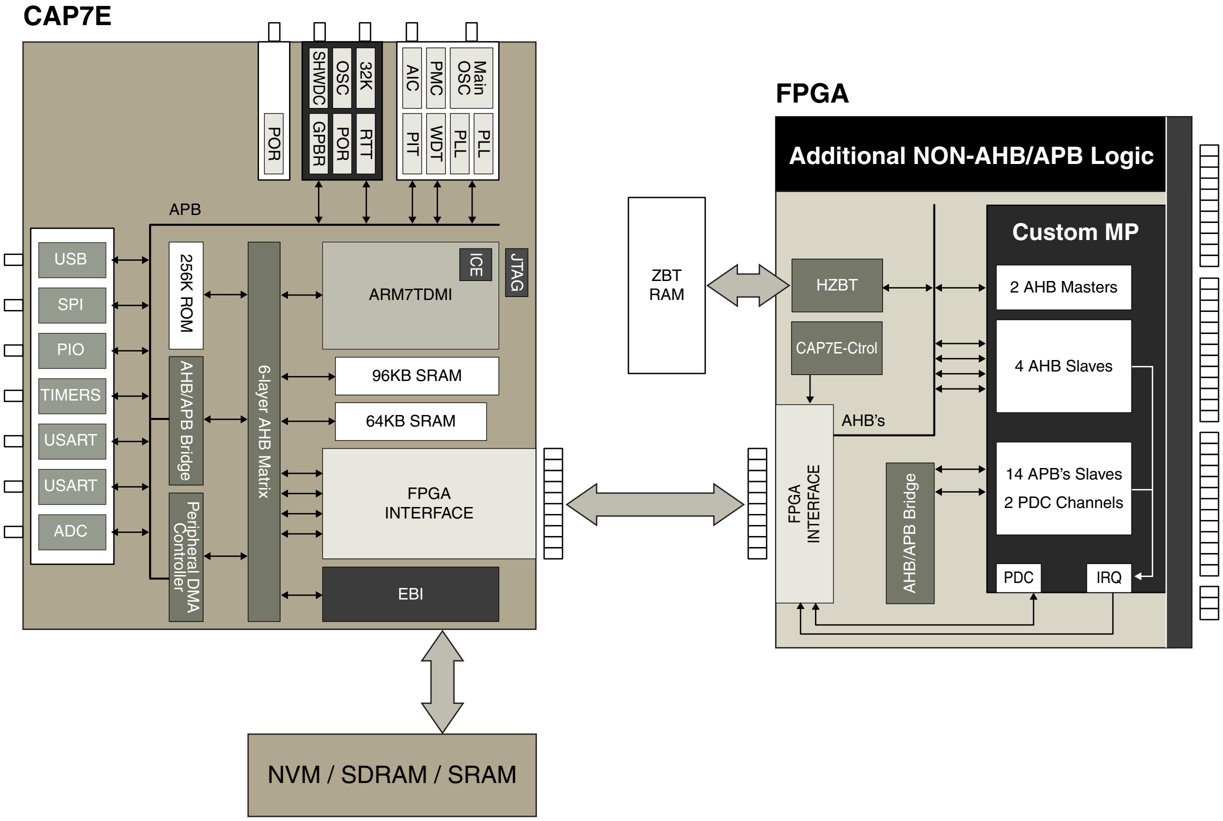
11.2 System Requirements and Integration

The FPGA interface is implemented using several serializers that encode/decode all the traffic between the CAP7E and the FPGA. In order to have proper communication and synchronization between both devices, the following requirements must be met:

1. The FPGA being connected to CAP7E must be capable of handling skew clock balancing and latency cancellation. For example in a Xilinx FPGA, the use of DCM's is mandatory.
2. The FPGA must provide the configuration modes and a reset to the CAP7E.
3. The FPGA must provide the serial communication clock to CAP7E.
4. The frequency for the serializer clock can be as fast as 100Mhz for the commercial temperature/voltage/process range.
5. The ratio between the internal CAP7E AHB Master Clock (MCK) and the FPGA Interface Serial Clock (FPIF_SCLK) should be approximately 0.8 or lower (MCK / FPIF_SCLK).
6. All the logic added to the FPGA must utilize the Atmel-provided encoding/decoding logic to ensure proper communication with CAP7E. Currently only Altera and Xilinx FPGA's are supported, but other FPGA's may be supported in the future.
7. A template is provided to instantiate the AHB Masters and Slaves with the FPGA interface.

ATMEL provides some examples of how to integrate logic in the FPGA using the CAP7E FPGA interface. Figure 11-1 shows a system diagram of the CAP7E and an FPGA.

Figure 11-1. .CAP7E and FPGA System Diagram


CAP7E
POR
SHWDC
GPBR
OSC
POR
32K
RTT
AIC
PIT
PMC
WDT
Main OSC
PLL
PLL
APB
USB
SPI
PIO
TIMERS
USART
USART
ADC
256K ROM
AHB/APB Bridge
Peripheral DMA Controller
6-layer AHB Matrix
ARM7TDMI
ICE
JTAG
96KB SRAM
64KB SRAM
FPGA INTERFACE
EBI
ZBT RAM
FPGA
Additional NON-AHB/APB Logic
HZBT
CAP7E-Ctrol
FPGA INTERFACE
AHB's
AHB/APB Bridge
Custom MP
2 AHB Masters
4 AHB Slaves
14 APB's Slaves
2 PDC Channels
PDC
IRQ
NVM / SDRAM / SRAM

Note: The external ZBT-RAM and NVM/SDRAM/SRAM are optional, based on applications and system requirements

The module called "Custom MP" shown inside the FPGA is logic from an RTL template provided to simplify the integratration of AHB or APB peripherals. Using "Custom MP" will also make a migration from a CAP7E to a fully customized CAP7 solution much easier since modules are connected the same way in the wrapper for the CAP7 MP block.

All the RTL for the interface targeted for the FPGA and additional modules such as a HZBT, AHB/APB bridge, etc. provided by ATMEL contain all the proper constraints for each supported FPGA vendor. Additional customer-specific logic can also be added to the FPGA.

11.3 Functional Description

The FPGA Interface includes logic that encodes or decodes the internal AHB transactions. The encoded/decoded data is transferred through MPIO's using dedicated serializers for each master and slave. Due to the large number of bits to be transferred, a single transfer will take several AHB clock cycles. The specific number of clock cycles depends on the ratio between the CAP7E MCK and FPIF_SCLK and the ratio between the FPGA AHB clock and the FPIF_SCLK. The lower those two ratios are, the fewer AHB clocks it will take for a single transfer.

NOTE The AHB master clock on the CAP7E is independent from the AHB clock on the FPGA. Therefore, the FPGA can run at a different frequency than the CAP7E.

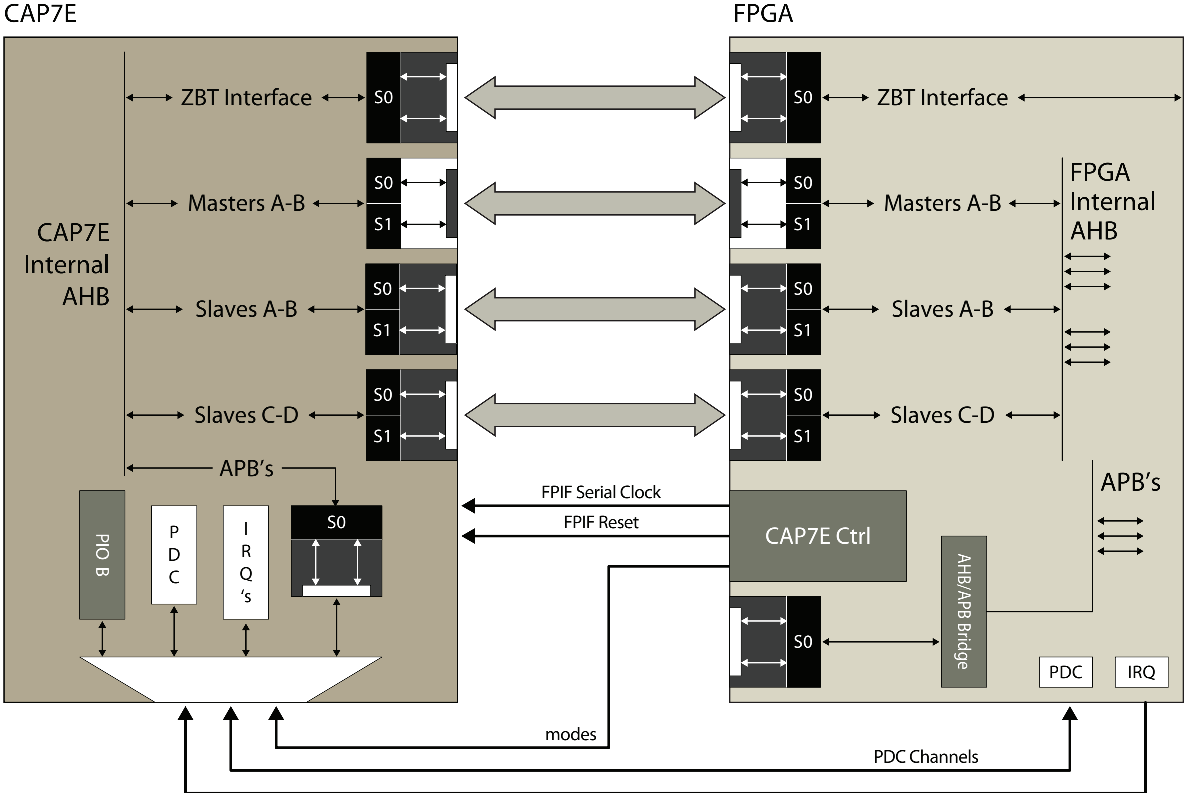
11.3.1 Interface Modules

Each serializer block on CAP7E and FPGA includes a FSM (Finite State Machine) that can communicate with the AHB bus. Thus, the interface can handle simultaneous transfers from either side eliminating the common bottleneck found using other interface types such as EBI or PIO.

By using the dedicated DMA channels (PDC), the overall system performance and bandwidth is greatly improved. The ARM7TDMI need not be burdened with transferring data to or from the FPGA but can be reserved for more intense processing.

Figure 11-2 shows a top level description for both interfaces (CAP7E and FPGA).

Figure 11-2. FPGA Interface architecture


CAP7E
FPGA
CAP7E Internal AHB
ZBT Interface
S0
Masters A-B
S0
S1
Slaves A-B
S0
S1
Slaves C-D
S0
S1
APB's
PIO B
PDC
IRQ's
S0
FPGA Internal AHB
ZBT Interface
Masters A-B
Slaves A-B
Slaves C-D
APB's
FPIF Serial Clock
FPIF Reset
CAP7E Ctrl
AHB/APB Bridge
S0
PDC
IRQ
modes
PDC Channels

11.3.2 Serializer Modules

The Serializer Module handles all the AHB and serial communications. It contains 2 main sub-modules, a finite state machine (FSM) and a shifter.

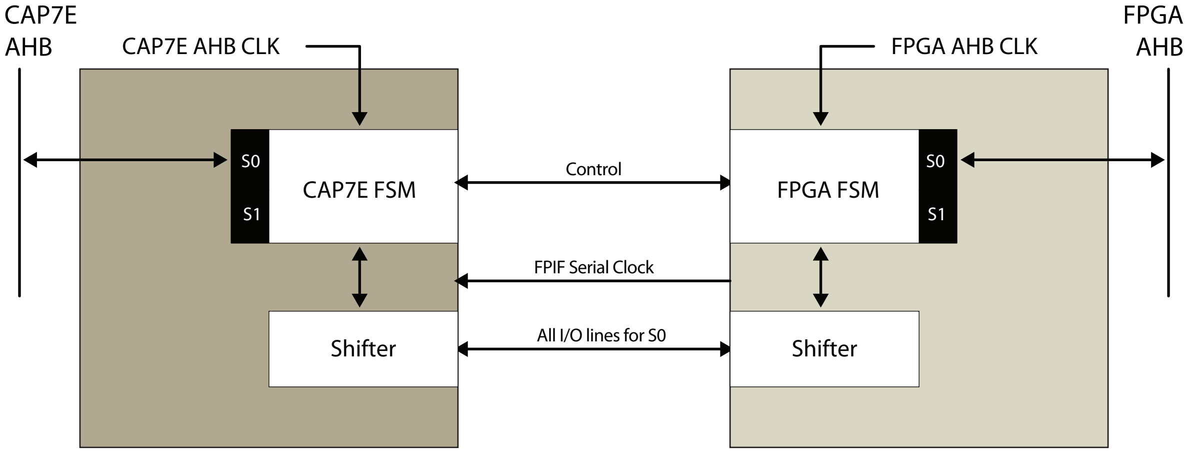
- **FSM:** This block communicates with the AHB bus. When a master initiates a transfer (read/write operation), the FSM inserts any necessary wait states using HREADY to comply with the AHB protocol. The number of wait cycles inserted by the FSM depends upon the two ratios between the CAP7E and FPGA AHB clocks and the FPGA Interface Serial Clock (FPIF_SCLK). Therefore, the smaller those ratios, the less number of wait states are inserted.

- Shifter: This block is controlled by the FSM, and it handles all the data shifting (serializing) between the CAP7E-FPGA and transfers 2 bits per cycle. If the FPIF_SCLK rate is set @100mhz, then the shifters transfer 200Mbps.

11.3.3 Serializer Programmability

In order to maximize the number of I/O supported, modules that handle the Masters-A/B, Slaves-A/B and Slaves C/D, are programmable at reset time through the CAP7E Control module in the FPGA. This programmability allows the user to choose whether or not to use "all" 10 I/O lines for a single serial configuration. In Figure 11-3, the serial module is shown configured to handle only 1 AHB interface. For example, if the user wants to use only AHB master A, then the appropriate serial module will need to be configured by setting the Master mode configuration in the CAP7E Control module to Single Master Mode, which will improve the number of bits transferred between shifters and speed-up the transfers between the CAP7E and FPGA.

Figure 11-3. Single Master Mode

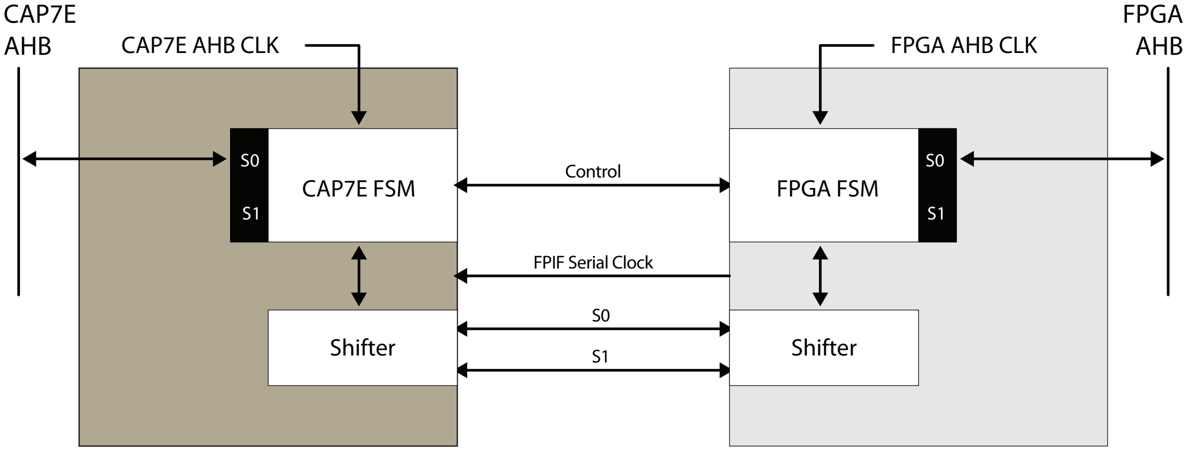


Another option is to configure the serial module to handle 2 AHB interfaces in Dual Master Mode. Here the 10 I/O lines are shared between the 2 AHB (Masters/Slaves).

In this case, the data transfer rate between the CAP7E and the FPGA is reduced, but the data bandwidth increases because now 2 AHB interfaces are enabled.

Figure 11-4 shows how the Dual Master Mode uses half of the dedicated I/O for another AHB interface.

Figure 11-4. Dual Master Mode


CAP7E
AHB
CAP7E AHB CLK
FPGA AHB CLK
FPGA
AHB
S0
S1
CAP7E FSM
Control
FPGA FSM
S0
S1
FPIF Serial Clock
S0
S1
Shifter
Shifter

11.3.4 Transfer Timing

As mentioned previously, the number of clocks per transfer and therefore the effective transfer speed depends upon the two ratios between the CAP7E and FPGA AHB clock frequencies and the FPIF_SCLK. In addition, the Master Mode selection affects the effective transfer speed as follows:

- **Single Master Mode:** Takes 4 FPIF_SCLK cycles to transfer data for 1 AHB interface. See t2 and t3 on Figure 11-5 below.
- **Dual Master Mode:** Takes 8 FPIF_SCLK cycles to transfer all AHB data of 2 AHB interfaces.

Figure 11-5. Read/Write timing for Single Master Mode

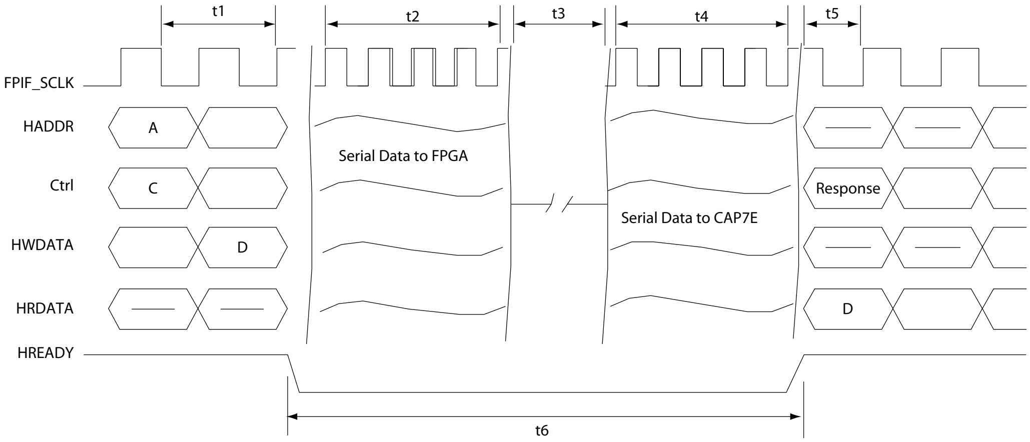

t1
t2
t3
t4
t5
FPIF_SCLK
HADDR
Ctrl
HWDATA
HRDATA
HREADY
A
C
D
Serial Data to FPGA
Serial Data to CAP7E
Response
D
t6

Figure 11-5 shows all the timing for a transfer between the CAP7E and the FPGA.

- **t1:** Time for a standard 2 cycles AHB
- **t2:** Time to transfer the request to FPGA (4 cycles single AHB interface, 8 cycles dual AHB interface).

- **t3:** Time for FPGA-Peripheral response
- **t4:** Time to transfer response back to CAP7E (4 cycles single AHB interface, 8 cycles dual AHB interface)
- **t5:** Time to read back the response/data from FPGA to the internal CAP7E AHB bus
- **t6:** Time for introduced wait cycles

An approximation formula for the access time, from the ARM inside the CAP7E to the peripherals in the FPGA is shown below:

$$T_{access} = t1 + t2 + t3 + t4 + t5$$

11.4 Programmability Options

Inside the FPGA, the module called “CAP7E Control”, produces a reset and provides the different modes under reset conditions for the CAP7E. The RTL provided by ATMEL lets the user configure their FPGA interface. By default, all mode bits are zeroes.

11.4.1 Mode-Bits

The following table shows the description and value for the emulation/modes bits supported by CAP7E.

Mode-Bit	Description	0	1
0	Internal ROM select	Use internal ROM	Use external ZBT
1	Master mode select	Single Master Mode - use only Master A	Dual Master Mode - use Masters A and B
2	Slave mode select 1	SlaveA Mode - use only Slave A	SlaveA-B Mode - use Slaves A and B
3	Slave mode select 2	SlaveC Mode - use only Slave C	SlaveC-D Mode - use Slaves C and D
4	PIOB mode select	Use PIOB	Use FPIF IRQ’s, PDC, and APB bridge
5	CAP7 in ARM MODE used for emulation of CAP7 only	CAP7E mode	CAP7-ARM emulation mode
6	Disable Pullups	Use Pull-Ups	Disable-Pullups
7	ADC / LVDS Select used for emulation of CAP7 only	Use ADC	Use LVDS

Table 11-1. Mode-bits description

11.4.2 PIO Controller B Multiplexing

Table 11-2. Multiplexing on PIO Controller B

PIO Controller B			
I/O Line	**PIO Mode**	**APB Mode**	**Reset State**
MPIO00	PB0	FPP_IRQ_ENC0	
MPIO01	PB1	FPP_IRQ_ENC1	
MPIO02	PB2	FPP_IRQ_ENC2	
MPIO03	PB3	FPP_IRQ_ENC3	
MPIO04	PB4	FPP6_IRQ	
MPIO05	PB5	FPP7_IRQ	
MPIO06	PB6	FPP6_TX_BFFR_ EMPTY	
MPIO07	PB7	FPP6_RX_BFFR_ FULL	
MPIO08	PB8	FPP6_CHNL_TX_ END	
MPIO09	PB9	FPP6_CHNL_RX _END	
MPIO10	PB10	FPP6_TX_RDY	
MPIO11	PB11	FPP6_RX_RDY	
MPIO12	PB12	FPP6_TX_SIZE0	
MPIO13	PB13	FPP6_TX_SIZE1	
MPIO14	PB14	FPP6_RX_SIZE0	
MPIO15	PB15	FPP6_RX_SIZE1	
MPIO16	PB16	FPP7_TX_BFFR_ EMPTY	
MPIO17	PB17	FPP7_RX_BFFR_ FULL	
MPIO18	PB18	FPP7_CHNL_TX_ END	
MPIO19	PB19	FPP7_CHNL_RX _END	
MPIO20	PB20	FPP7_TX_RDY	
MPIO21	PB21	FPP7_RX_RDY	
MPIO22	PB22	FPP7_TX_SIZE0	
MPIO23	PB23	FPP7_TX_SIZE1	
MPIO24	PB24	FPP7_RX_SIZE0	
MPIO25	PB25	FPP7_RX_SIZE1	
MPIO26	PB26	APB_C	
MPIO27	PB27	APB_D0	

Table 11-2. Multiplexing on PIO Controller B

PIO Controller B			
I/O Line	**PIO Mode**	**APB Mode**	**Reset State**
MPIO28	PB28	APB_D1	
MPIO29	PB29	APB_A0	
MPIO30	PB30	APB_A1	
MPIO31	PB31	APB_START	

11.4.3 Other MPIO Signal Assignments/Multiplexing

Table 11-3. MPIO Signal Assignments/Multiplexing

I/O Line	Single Mode	Dual Mode
MPIO32	MA_C2	MB_C
MPIO33	MA_C1	MB_D0
MPIO34	MA_D0	MB_D1
MPIO35	MA_D1	MB_A0
MPIO36	MA_D2	MB_A1
MPIO37	MA_D3	MA_C
MPIO38	MA_A0	MA_D
MPIO39	MA_A1	MA_D1
MPIO40	MA_A2	MA_A0
MPIO41	MA_A3	MA_A1
MPIO42	MA_START	MA_START
MPIO43	MB_START	MB_START
MPIO44	SA_C2	SB_C
MPIO45	SA_C1	SB_D0
MPIO46	SA_D0	SB_D1
MPIO47	SA_D1	SB_A0
MPIO48	SA_D2	SB_A1
MPIO49	SA_D3	SA_C
MPIO50	SA_A0	SA_D0
MPIO51	SA_A1	SA_D1
MPIO52	SA_A2	SA_A0
MPIO53	SA_A3	SA_A1
MPIO54	SA_START	SA_START
MPIO55	SB_START	SB_START
MPIO56	SC_C2	SD_C
MPIO57	SC_C1	SD_D0

Table 11-3. MPIO Signal Assignments/Multiplexing

I/O Line	Single Mode	Dual Mode
MPIO58	SC_D0	SD_D1
MPIO59	SC_D1	SD_A0
MPIO60	SC_D2	SD_A1
MPIO61	SC_D3	SC_C
MPIO62	SC_A0	SC_D0
MPIO63	SC_A1	SC_D1
MPIO64	SC_A2	SC_A0
MPIO65	SC_A3	SC_A1
MPIO66	SC_START	SC_START
MPIO67	SD_START	SD_START
MPIO68	SZBT_C2	
MPIO69	SZBT_C1	
MPIO70	SZBT_D0	
MPIO71	SZBT_D1	
MPIO72	SZBT_D2	
MPIO73	SZBT_D3	
MPIO74	SZBT_A0	
MPIO75	SZBT_A1	
MPIO76	SZBT_A2	
MPIO77	SZBT_A3	
MPIO78	SZBT_START	
MPIO79	FPIF_SCLK	
MPIO80	FPIF_SCLK_FEEDBK	
MPIO81	FPIF_RESETN	

11.5 Interfacing using PIO

An FPGA interace can also be created using PIO's (Programmable Input/Outputs). This approach is relatively simple, and most of the hard work is done by software. However, the ARM processor must move the data to/from the PIO and generate all the necessary signaling on PIO for the FPGA to properly handle the transfers being made.

This kind of interface is easy to implement, however in the FPGA special logic has to be implemented to decode all the traffic generated by the PIO. The traffic from the standard microcontroller to the FPGA is very likely to be completely asynchronous, so the FPGA must be able to oversample the control signals from the micro, otherwise the FPGA will miss the time window and the data will not arrive at its final destination inside the FPGA.

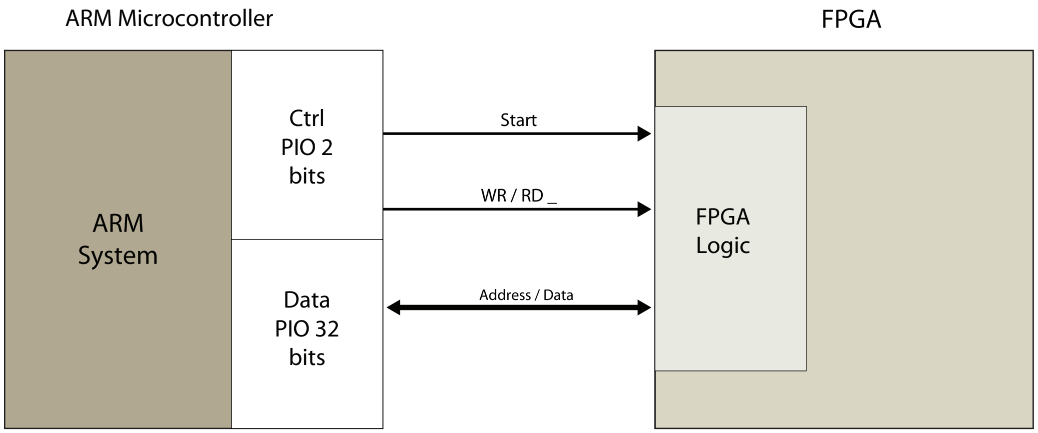
Since the processor must manage the flow of data to keep the PIO busy, there is a significant overhead in processing time. Note that DMA is not possible using this architecture, therefore the bandwidth is limited by the number of cycles the software programmer allocates for the processor to communicate with the PIO. For example, if there is a routine running that demands 100% of the processor cycles and concurrently there is serial data (e.g. SPI, USART, USB, or TWI) to be transferred to/from the FPGA, one of these processes must wait. If the data from the FPGA is not buffered on time, it will probably be overrun by the next byte/word.

11.5.1 PIO-FPGA Connections

To accomplish a proper data transfer to/from the FPGA, we need to transfer 32 bits of address (or possibly less), 32 bits of data, and some control signals. For this approach, one will need to use more that a 32 bit PIO port. At least 2 more PIO bits are necessary for control signals.

The Figure 11-6 shows the 32+2 PIO interface to a FPGA.

Figure 11-6. PIO interface to FPGA



11.5.2 PIO-FPGA Access Routines

Based on the resources shown above, we can define a software algorithm to transfer data from/to FPGA.

- **write_to_fpga:** Algorithm to write 32 bits of data to FPGA, this assumes that, the direction of the bidirectional buffers in the PIO's has been previously set.

  ```
  PIO_DATA = ADDRESS; // Pass the address to write
  PIO_CTRL = START | WR; // Send start of address cycle
  PIO_CTRL = CLEAR; // Clear PIO ctrl, this ends the address cycle
  PIO_DATA = DATA; // Set data to transfer
  PIO_CTRL = START; // Data is ready in PIO
  PIO_CTRL = CLEAR; // This end the data cycle
  ```

- read_from_fpga: Algorithm to read data from the FPGA, this assumes that, the direction of the bidirectional buffers in the PIO's has been previously set.

  ```
  PIO_DATA = ADDRESS; // Set the address to read
  PIO_CTRL = START | RD; // Send start of address cycle
  PIO_CTRL = CLEAR; // Clear PIO ctrl, this ends the address cycle
  ```

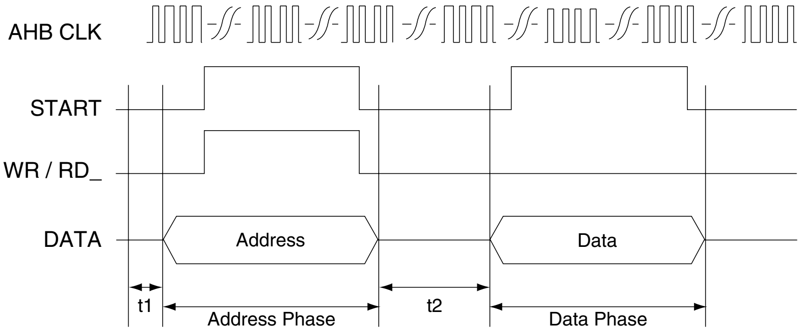
```
PIO_DATA_DIR = INPUT; // Set PIO-Data direction as input to receive the data
DELAY(WAIT_FOR_FPGA); // wait for the FPGA to send the data
DATA_FROM_FPGA = *PIO_DATA; // this is the end of read cycle
```

NOTE These algorithms are for a basic transfer, a more sophisticated algorithm is necessary to establish a proper communication between the ARM microcontroller and the FPGA.

11.5.3 PIO-FPGA Waveforms

Figure 11-7shows the PIO timing when writing to FPGA.

Figure 11-7. Write to FPGA


AHB CLK
START
WR / RD_
DATA
Address
Data
t1
Address Phase
t2
Data Phase

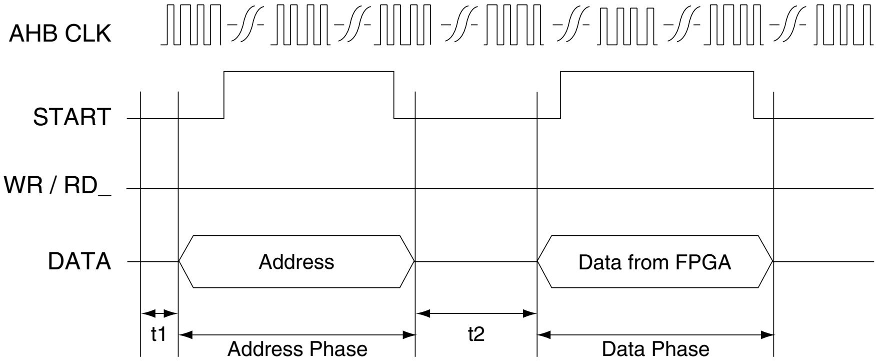
The access time is calculated as the sum of:

$$T_{access\text{-}Pio} = t1 + \text{address phase} + t2 + \text{data phase}$$

Using the GCC compiler with maximum optimizations, the system takes approximately 55 AHB cycles to perform the write operation to the FPGA.

Figure 11-8 shows the PIO timing when reading from the FPGA.

Figure 11-8. Read from FPGA


AHB CLK
START
WR / RD_
DATA
Address
Data from FPGA
t1
Address Phase
t2
Data Phase

Using the GCC compiler with maximum optimization and assuming t2 (wait for FPGA response ready) is also around 25 AHB cycles, and the system takes approximately 85 AHB cycles for a read operation from the FPGA.

11.6 Interfacing using EBI

The External Bus Interface (EBI) module, is designed to transfer data between external devices and the Memory Controllers of an ARM based device. These external Memory Controllers are capable of handling several types of external memory and peripheral devices, such as SDRAM, SRAM, NOR Flash, NAND Flash, and various PROM devices.

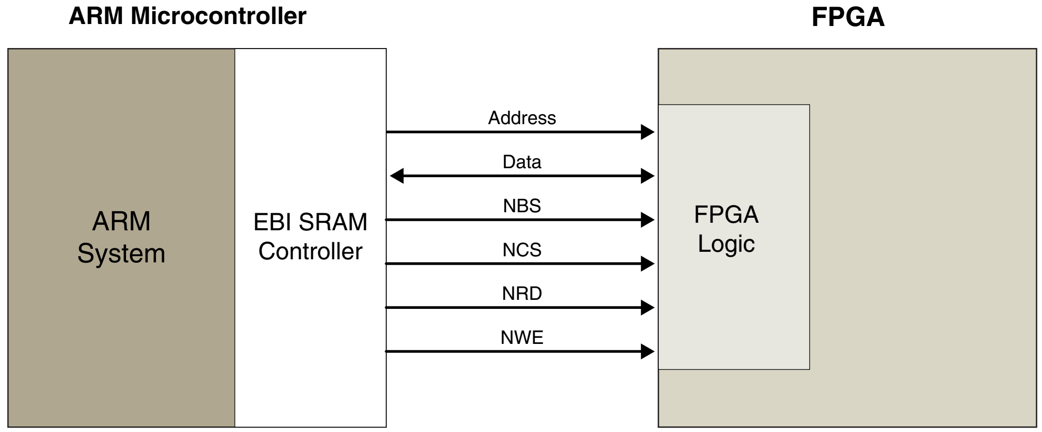
However, the EBI can also provide an interface to an FPGA as long as the FPGA can work with one of the predefined memory interfaces. Due to its simplicity and familiarity, the Static Memory Controller (SMC) which supports an SRAM-type interface is preferred for this purpose. Usually the FPGA will have to include a module that understands the SMC timing and is able to respond to the SMC as expected.

The EBI interface already provides all the necessary parallel, high-drive I/O to allow a user to communicate with an FPGA with reasonable performance. However if the external device is slow or introduces wait cycles, the throughput of the interface could be compromised. Also since the EBI must be driven by the processor or another AHB master, the bandwidth that the EBI can achieve is partly determined by the software that sets the bus and interrupt priorities, etc.

11.6.1 EBI-FPGA Connections

Figure 11-9 shows the ARM Microcontroller driving the FPGA through the EBI. The selected interface is the SMC. A special module need to be designed in the FPGA to interface the EBI-SMC to the CAP7E microcontroller.

Figure 11-9. EBI driving the FPGA



11.6.2 EBI TIming

Figure 11-10 shows the standard read timing for the EBI using the SMC memory interface and Figure 11-11 shows the standard write cycle.

NOTE These timing diagrams are also shown in section TBD. All parameters shown are programmable based on the speed of the external FPGA.

Figure 11-10. Read Cycle

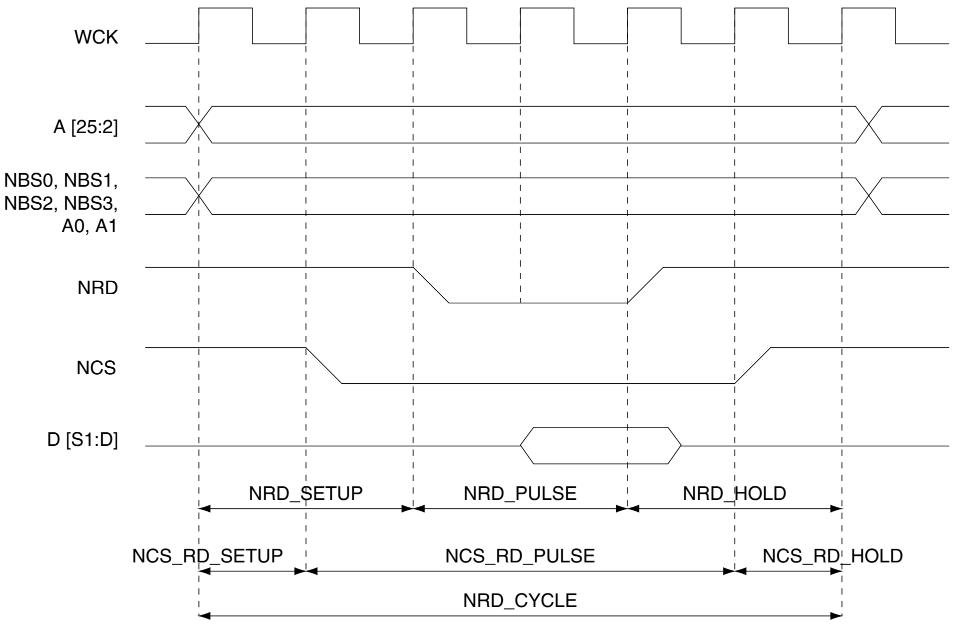
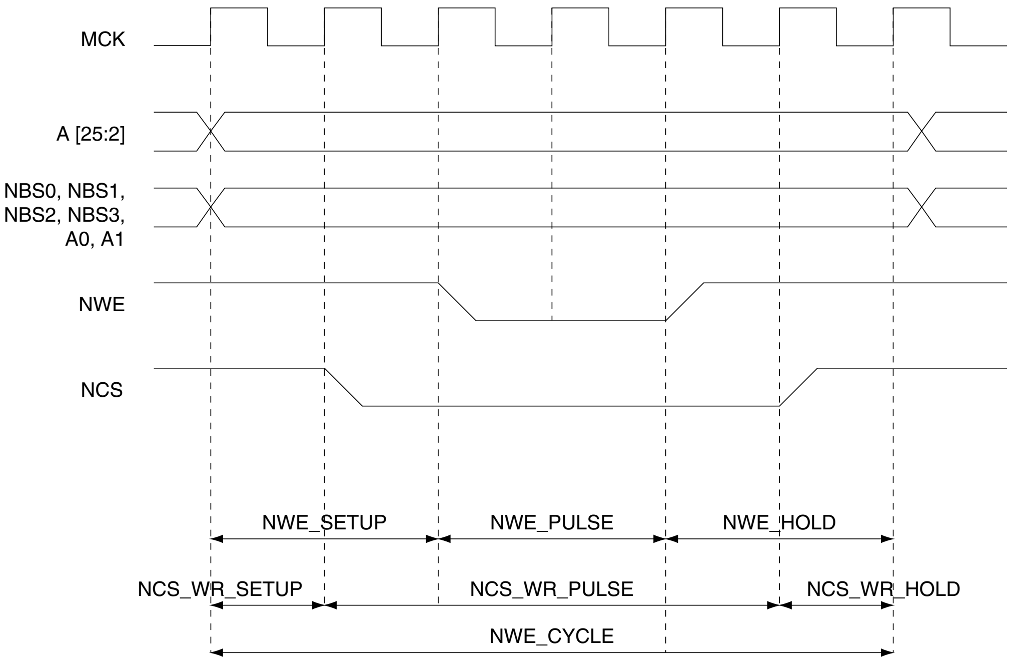

WCK
A [25:2]
NBS0, NBS1,
NBS2, NBS3,
A0, A1
NRD
NCS
D [S1:D]
NRD_SETUP
NRD_PULSE
NRD_HOLD
NCS_RD_SETUP
NCS_RD_PULSE
NCS_RD_HOLD
NRD_CYCLE

Figure 11-11. Write Cycle


MCK
A [25:2]
NBS0, NBS1,
NBS2, NBS3,
A0, A1
NWE
NCS
NWE_SETUP
NWE_PULSE
NWE_HOLD
NCS_WR_SETUP
NCS_WR_PULSE
NCS_WR_HOLD
NWE_CYCLE

12. ARM7TDMI Processor Overview

12.1 Overview

The ARM7TDMI core executes both the 32-bit ARM® and 16-bit Thumb® instruction sets, allowing the user to trade off between high performance and high code density.The ARM7TDMI processor implements Von Neuman architecture, using a three-stage pipeline consisting of Fetch, Decode, and Execute stages.

The main features of the ARM7tDMI processor are:

- ARM7TDMI Based on ARMv4T Architecture
- Two Instruction Sets
  - ARM® High-performance 32-bit Instruction Set
  - Thumb® High Code Density 16-bit Instruction Set
- Three-Stage Pipeline Architecture
  - Instruction Fetch (F)
  - Instruction Decode (D)
  - Execute (E)

12.2 ARM7TDMI Processor

For further details on ARM7TDMI, refer to the following ARM documents:

ARM Architecture Reference Manual (DDI 0100E)

ARM7TDMI Technical Reference Manual (DDI 0210B)

12.2.1 Instruction Type

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

12.2.2 Data Type

ARM7TDMI supports byte (8-bit), half-word (16-bit) and word (32-bit) data types. Words must be aligned to four-byte boundaries and half words to two-byte boundaries.

Unaligned data access behavior depends on which instruction is used where.

12.2.3 ARM7TDMI Operating Mode

The ARM7TDMI, based on ARM architecture v4T, supports seven processor modes:

User: The normal ARM program execution state

FIQ: Designed to support high-speed data transfer or channel process

IRQ: Used for general-purpose interrupt handling

Supervisor: Protected mode for the operating system

Abort mode: Implements virtual memory and/or memory protection

System: A privileged user mode for the operating system

Undefined: Supports software emulation of hardware coprocessors

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

12.2.4 ARM7TDMI Registers

The ARM7TDMI processor has a total of 37registers:

- 31 general-purpose 32-bit registers
- 6 status registers

These registers are not accessible at the same time. The processor state and operating mode determine which registers are available to the programmer.

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

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

R14 holds the return address after a subroutine call.

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

Table 12-1. ARM7TDMI ARM Modes and Registers Layout

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

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_SVC	SPSR_ABORT	SPSR_UNDEF	SPSR_IRQ	SPSR_FIQ

▨ Mode-specific banked registers

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

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

12.2.4.1 Modes and Exception Handling

All exceptions have banked registers for R14 and R13.

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

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

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

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

Exception vectors are located starting at address 0x0000 0000.

12.2.4.2 Status Registers

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

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

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

12.2.4.3 Exception Types

The ARM7TDMI supports five types of exception and a privileged processing mode for each type. The types of exceptions are:

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

Exceptions are generated by internal and external sources.

More than one exception can occur in the same time.

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

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

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

12.2.5 ARM Instruction Set Overview

The ARM instruction set is divided into:

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

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

Table 12-2 gives the ARM instruction mnemonic list.

Table 12-2. ARM Instruction Mnemonic List

Mnemonic	Operation	Mnemonic	Operation
MOV	Move	CDP	Coprocessor Data Processing
ADD	Add	MVN	Move Not
SUB	Subtract	ADC	Add with Carry
RSB	Reverse Subtract	SBC	Subtract with Carry
CMP	Compare	RSC	Reverse Subtract with Carry
TST	Test	CMN	Compare Negated
AND	Logical AND	TEQ	Test Equivalence
EOR	Logical Exclusive OR	BIC	Bit Clear
MUL	Multiply	ORR	Logical (inclusive) OR
SMULL	Sign Long Multiply	MLA	Multiply Accumulate
SMLAL	Signed Long Multiply Accumulate	UMULL	Unsigned Long Multiply
MSR	Move to Status Register	UMLAL	Unsigned Long Multiply Accumulate
B	Branch	MRS	Move From Status Register
BX	Branch and Exchange	BL	Branch and Link
LDR	Load Word	SWI	Software Interrupt
LDRSH	Load Signed Halfword	STR	Store Word
LDRSB	Load Signed Byte	STRH	Store Half Word
LDRH	Load Half Word	STRB	Store Byte
LDRB	Load Byte	STRBT	Store Register Byte with Translation
LDRBT	Load Register Byte with Translation	STRT	Store Register with Translation
LDRT	Load Register with Translation	STM	Store Multiple

Table 12-2. ARM Instruction Mnemonic List

Mnemonic	Operation	Mnemonic	Operation
LDM	Load Multiple	SWPB	Swap Byte
SWP	Swap Word	MRC	Move From Coprocessor
MCR	Move To Coprocessor	STC	Store From Coprocessor
LDC	Load To Coprocessor		

12.2.6 Thumb Instruction Set Overview

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

The Thumb instruction set is divided into:

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

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

Table 12-3 gives the Thumb instruction mnemonic list.

Table 12-3. Thumb Instruction Mnemonic List

Mnemonic	Operation	Mnemonic	Operation
MOV	Move	MVN	Move Not
ADD	Add	ADC	Add with Carry
SUB	Subtract	SBC	Subtract with Carry
CMP	Compare	CMN	Compare Negated
TST	Test	NEG	Negate
AND	Logical AND	BIC	Bit Clear
EOR	Logical Exclusive OR	ORR	Logical (inclusive) OR
LSL	Logical Shift Left	LSR	Logical Shift Right
ASR	Arithmetic Shift Right	ROR	Rotate Right
MUL	Multiply		
B	Branch	BL	Branch and Link
BX	Branch and Exchange	SWI	Software Interrupt
LDR	Load Word	STR	Store Word
LDRH	Load Half Word	STRH	Store Half Word
LDRB	Load Byte	STRB	Store Byte

Table 12-3. Thumb Instruction Mnemonic List

Mnemonic	Operation	Mnemonic	Operation
LDRSH	Load Signed Halfword	LDRSB	Load Signed Byte
LDMIA	Load Multiple	STMIA	Store Multiple
PUSH	Push Register to stack	POP	Pop Register from stack

13. CAP7E Debug and Test

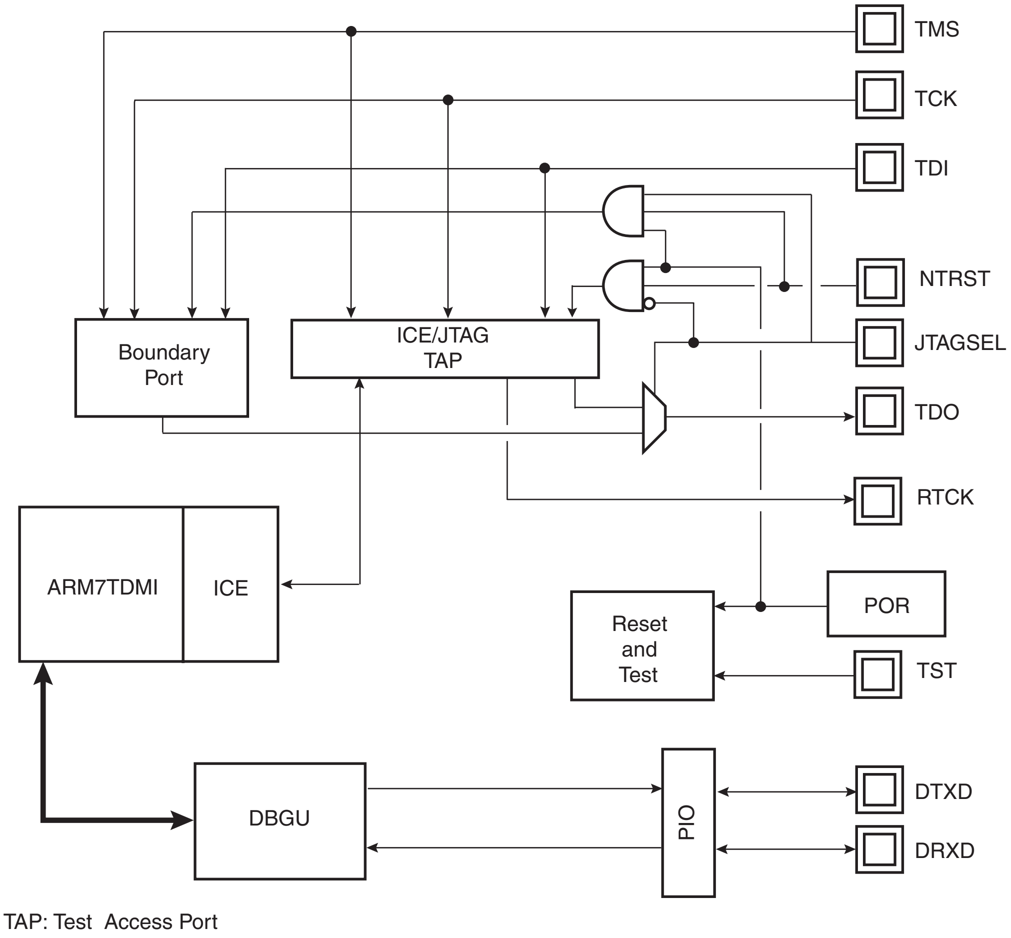
13.1 Overview

The AT91CAP7E features a number of complementary debug and test capabilities. A common JTAG/ICE (In-Circuit Emulator) port is used for standard debugging functions, such as downloading code and single-stepping through programs. The Debug Unit provides a two-pin UART that can be used to upload an application into internal SRAM. It manages the interrupt handling of the internal COMMTX and COMMRX signals that trace the activity of the Debug Communication Channel.

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

13.2 Block Diagram

Figure 13-1. Debug and Test Block Diagram

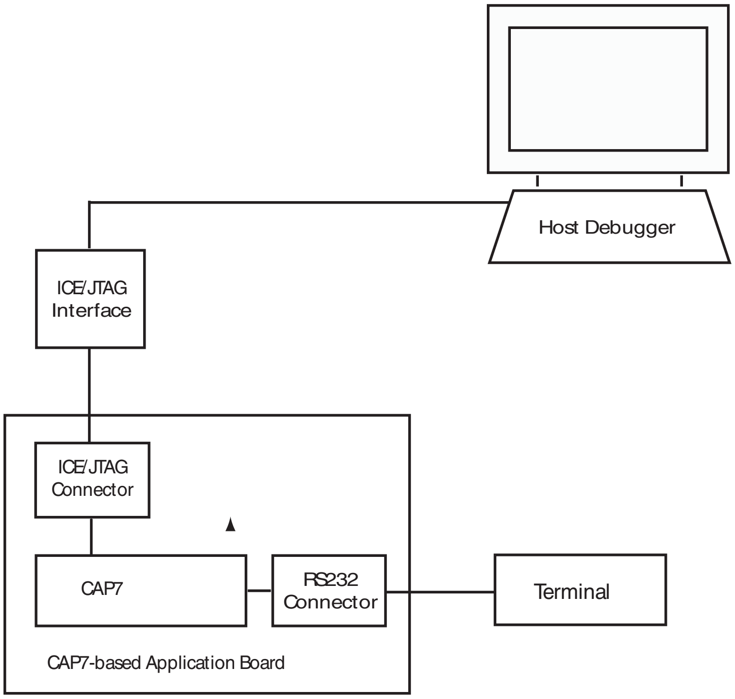

TMS
TCK
TDI
NTRST
JTAGSEL
TDO
RTCK
Boundary Port
ICE/JTAG TAP
ARM7TDMI
ICE
Reset and Test
POR
TST
DBGU
PIO
DTXD
DRXD
TAP: Test Access Port

13.3 Application Examples

13.3.1 Debug Environment

Figure 13-2 on page 62 shows a complete debug environment example. The ICE/JTAG interface is used for standard debugging functions, such as downloading code and single-stepping through the program. A software debugger running on a personal computer provides the user interface for ICE/JTAG interface.

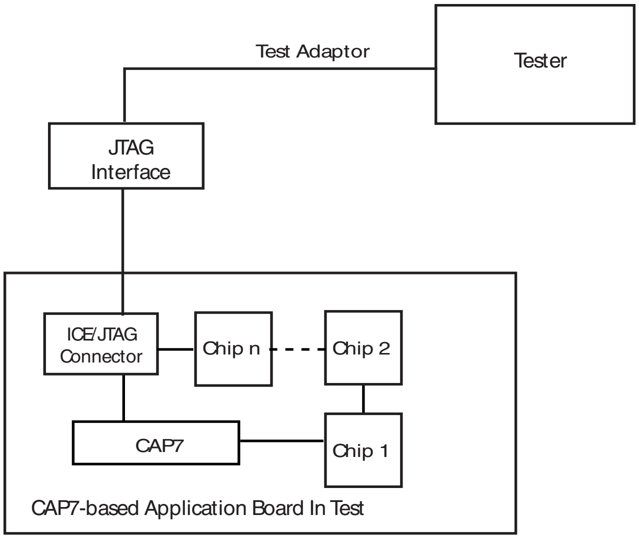
Figure 13-2. Application Debug and Trace Environment Example


Host Debugger
ICE/JTAG Interface
ICE/JTAG Connector
CAP7
RS232 Connector
Terminal
CAP7-based Application Board

13.3.2 Test Environment

Figure 13-3 on page 63 shows a test environment example. Test vectors are sent and interpreted by the tester. In this example, the “board in test” is designed using a number of JTAG-compliant devices. These devices can be connected to form a single scan chain.

Figure 13-3. Application Test Environment Example


Test Adaptor
Tester
JTAG Interface
ICE/JTAG Connector
Chip n
Chip 2
CAP7
Chip 1
CAP7-based Application Board In Test

13.4 Debug and Test Pin Description

Table 13-1. Debug and Test Pin List

Pin Name	Function	Type	Active Level
Reset/Test			
NRST	Microcontroller Reset	Input/Output	Low
TST	Test Mode Select	Input	High
ICE and JTAG			
TCK	Test Clock	Input	
TDI	Test Data In	Input	
TDO	Test Data Out	Output	
TMS	Test Mode Select	Input	
NTRST	Test Reset Signal	Input	Low
JTAGSEL	JTAG Selection	Input	
Debug Unit			
DRXD	Debug Receive Data	Input	
DTXD	Debug Transmit Data	Output	

13.5 Functional Description

13.5.1 Test Pin

One dedicated pin, TST, is used to define the device operating mode. The user must make sure that this pin is tied at low level to ensure normal operating conditions. Other values associated with this pin are reserved for manufacturing test.

13.5.2 Embedded In-circuit Emulator

The ARM7TDMI Embedded ICE is supported via the ICE/JTAG port. The internal state of the ARM7TDMI is examined through an ICE/JTAG port.

The ARM7TDMI processor contains hardware extensions for advanced debugging features: • In halt mode, a store-multiple (STM) can be inserted into the instruction pipeline. This exports the contents of the ARM7TDMI registers. This data can be serially shifted out without affecting the rest of the system. • In monitor mode, the JTAG interface is used to transfer data between the debugger and a simple monitor program running on the ARM7TDMI processor.

There are three scan chains inside the ARM7TDMI processor which support testing, debugging, and programming of the Embedded ICE. The scan chains are controlled by the ICE/JTAG port.

Embedded ICE mode is selected when JTAGSEL is low. It is not possible to switch directly between ICE and JTAG operations. A chip reset must be performed after JTAGSEL is changed.

For further details on the Embedded In-Circuit-Emulator, see the ARM document: ARM7TDMI (Rev 4) Technical Reference Manual (DDI0210B).

13.5.3 Debug Unit

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

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

A specific register, the Debug Unit Chip ID Register (DBGU_CIDR), gives information about the product's internal configuration and its version.

The AT91CAP7E Debug Unit Chip ID value is 0x8377 09xx on 32-bit width (1000 0011 0111 0111 0000 1001 010x xxxx). The last five bits of the register are reserved for a version number.

For further details on the Debug Unit, see the Debug Unit section.

13.5.4 IEEE 1149.1 JTAG Boundary Scan

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

IEEE 1149.1 JTAG Boundary Scan is enabled when JTAGSEL is high. The SAMPLE, EXTEST and BYPASS functions are implemented. In ICE debug mode, the ARM processor responds with a non-JTAG chip ID that identifies the processor to the ICE system. This is not IEEE 1149.1 JTAG-compliant.

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

A Boundary-scan Descriptor Language (BSDL) file is provided to set up test.

13.5.4.1 JTAG Boundary-scan Register

The Boundary-scan Register (BSR) contains bits that correspond to active pins and associated control signals.

Each AT91CAP7E input/output pin corresponds to a 3-bit register in the BSR. The OUTPUT bit contains data that can be forced on the pad. The INPUT bit facilitates the observability of data applied to the pad. The CONTROL bit selects the direction of the pad. Each customer's AT91CAP7E product may have its own unique BSR. For a full description of this BSR, see the appropriate product-specifc BSDL file.

13.5.5 ID Code Register

Access: Read-only

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

- **VERSION [31:28]: Product Version Number**

Set to 0x0.

- **PART NUMBER [27:12]: Product Part Number**

Personalization dependent

- **MANUFACTURER IDENTITY [11:1]**

Set to 0x01F.

- **Bit[0] Required by IEEE Std. 1149.1.**

Set to 0x1.

JTAG ID Code value is unique for each CAP7 personalization.

14. Reset Controller (RSTC)

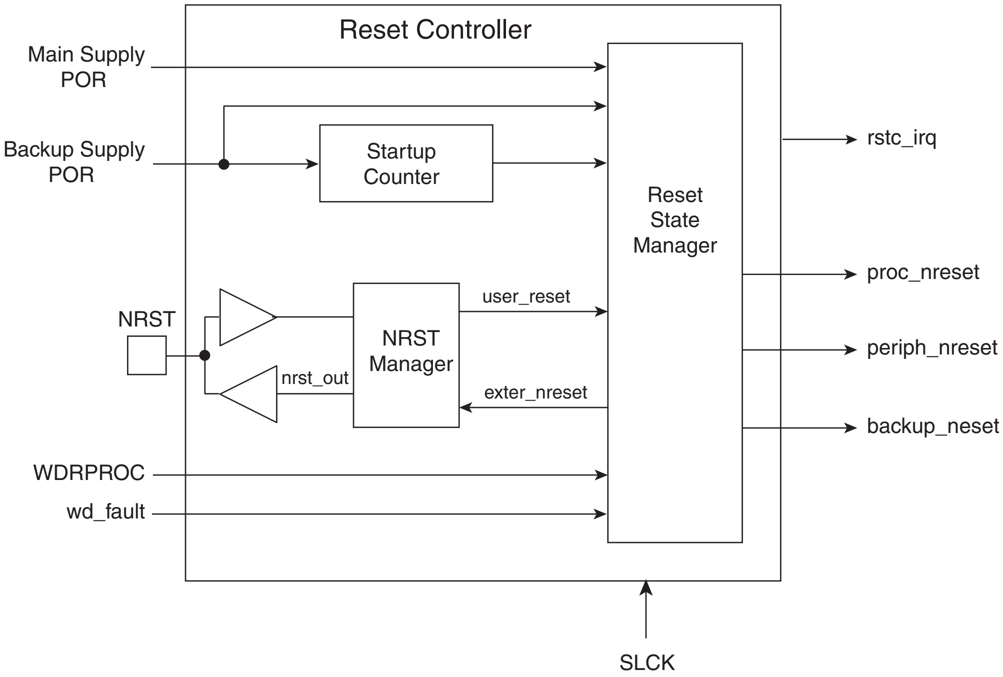
14.1 Description

The Reset Controller (RSTC), based on power-on reset cells, handles all the resets of the system without any external components. It reports which reset occurred last.

The Reset Controller also drives independently or simultaneously the external reset and the peripheral and processor resets.

14.2 Block Diagram

Figure 14-1. Reset Controller Block Diagram


Reset Controller
Main Supply POR
Backup Supply POR
Startup Counter
Reset State Manager
rstc_irq
proc_nreset
periph_nreset
backup_neset
NRST
NRST Manager
user_reset
nrst_out
exter_nreset
WDRPROC
wd_fault
SLCK

14.3 Functional Description

14.3.1 Reset Controller Overview

The Reset Controller is made up of an NRST Manager, a Startup Counter and a Reset State Manager. It runs at Slow Clock and generates the following reset signals:

- proc_nreset: Processor reset line. It also resets the Watchdog Timer.
- backup_nreset: Affects all the peripherals powered by VDDBU.
- periph_nreset: Affects the whole set of embedded peripherals.
- nrst_out: Drives the NRST pin.

These reset signals are asserted by the Reset Controller, either on external events or on software action. The Reset State Manager controls the generation of reset signals and provides a signal to the NRST Manager when an assertion of the NRST pin is required.

The NRST Manager shapes the NRST assertion during a programmable time, thus controlling external device resets.

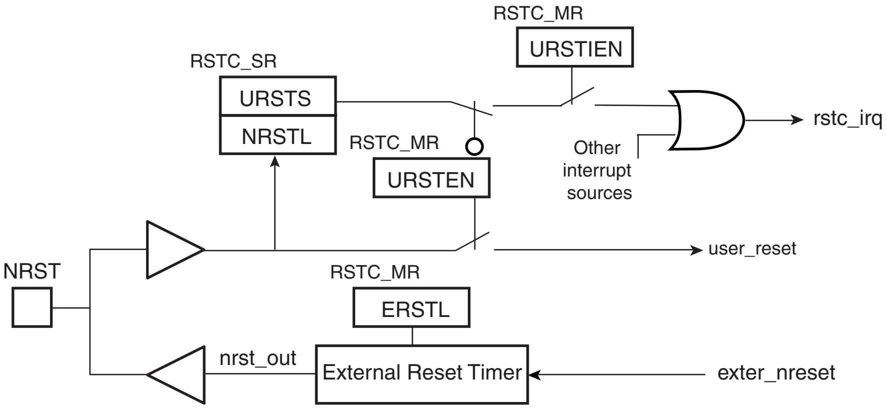
The startup counter waits for the complete crystal oscillator startup. The wait delay is given by the crystal oscillator startup time maximum value that can be found in the section Crystal Oscillator Characteristics in the Electrical Characteristics section of the product documentation.

The Reset Controller Mode Register (RSTC_MR), allowing the configuration of the Reset Controller, is powered with VDDBU, so that its configuration is saved as long as VDDBU is on.

14.3.2 NRST Manager

The NRST Manager samples the NRST input pin and drives this pin low when required by the Reset State Manager. Figure 14-2 shows the block diagram of the NRST Manager.

Figure 14-2. NRST Manager



14.3.2.1 *NRST Signal or Interrupt*

The NRST Manager samples the NRST pin at Slow Clock speed. When the line is detected low, a User Reset is reported to the Reset State Manager.

However, the NRST Manager can be programmed to not trigger a reset when an assertion of NRST occurs. Writing the bit URSTEN at 0 in RSTC_MR disables the User Reset trigger.

The level of the pin NRST can be read at any time in the bit NRSTL (NRST level) in RSTC_SR. As soon as the pin NRST is asserted, the bit URSTS in RSTC_SR is set. This bit clears only when RSTC_SR is read.

The Reset Controller can also be programmed to generate an interrupt instead of generating a reset. To do so, the bit URSTIEN in RSTC_MR must be written at 1.

14.3.2.2 *NRST External Reset Control*

The Reset State Manager asserts the signal ext_nreset to assert the NRST pin. When this occurs, the “nrst_out” signal is driven low by the NRST Manager for a time programmed by the field ERSTL in RSTC_MR. This assertion duration, named EXTERNAL_RESET_LENGTH, lasts $2^{(ERSTL+1)}$ Slow Clock cycles. This gives the approximate duration of an assertion between 60 µs and 2 seconds. Note that ERSTL at 0 defines a two-cycle duration for the NRST pulse.

This feature allows the Reset Controller to shape the NRST pin level, and thus to guarantee that the NRST line is driven low for a time compliant with potential external devices connected on the system reset.

As the field is within RSTC_MR, which is backed-up, this field can be used to shape the system power-up reset for devices requiring a longer startup time than the Slow Clock Oscillator.

14.3.3 Reset States

The Reset State Manager handles the different reset sources and generates the internal reset signals. It reports the reset status in the field RSTTYP of the Status Register (RSTC_SR). The update of the field RSTTYP is performed when the processor reset is released.

14.3.3.1 General Reset

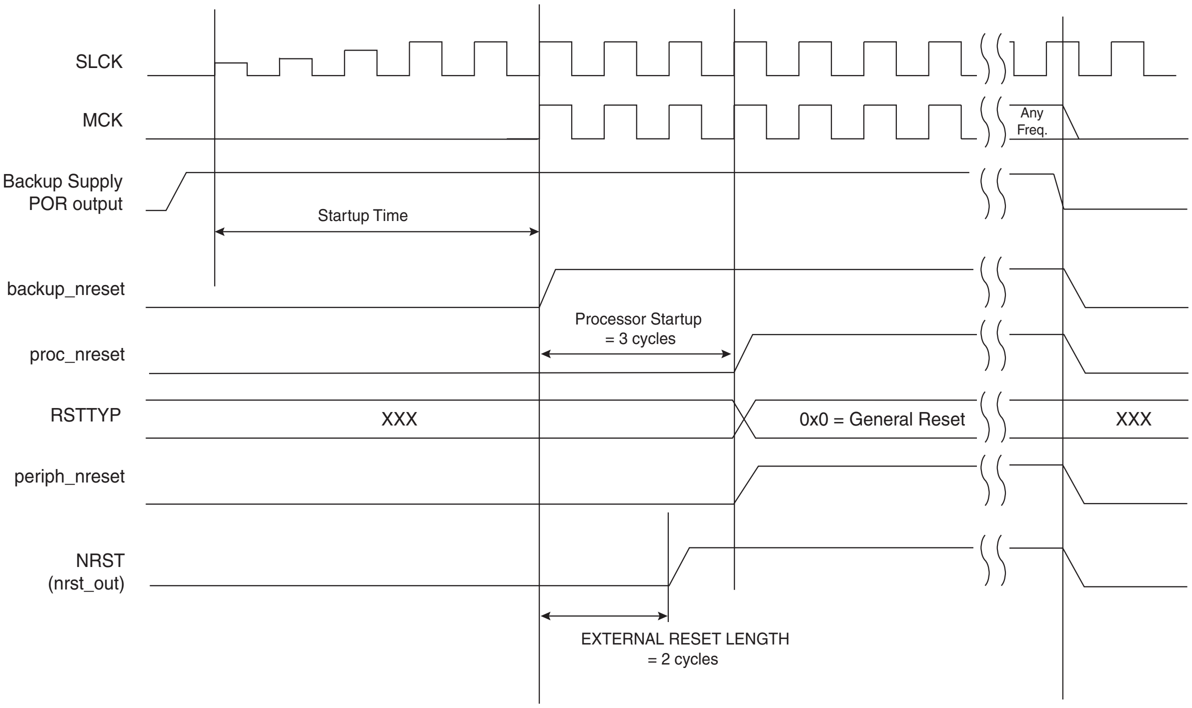
A general reset occurs when VDDBU is powered on. The backup supply POR cell output rises and is filtered with a Startup Counter, which operates at Slow Clock. The purpose of this counter is to make sure the Slow Clock oscillator is stable before starting up the device. The length of startup time is hardcoded to comply with the RC Oscillator startup time of 8 slow clock cycles.

After this time, the processor clock is released at Slow Clock and all the other signals remains valid for 2 cycles for proper processor and logic reset. Then, all the reset signals are released and the field RSTTYP in RSTC_SR reports a General Reset. As the RSTC_MR is reset, the NRST line rises 2 cycles after the backup_nreset, as ERSTL defaults at value 0x0.

When VDDBU is detected low by the Backup Supply POR Cell, all resets signals are immediately asserted, even if the Main Supply POR Cell does not report a Main Supply shut down.

Figure 14-3 shows how the General Reset affects the reset signals.

Figure 14-3. General Reset State


SLCK
MCK
Any Freq.
Backup Supply POR output
Startup Time
backup_nreset
Processor Startup = 3 cycles
proc_nreset
RSTTYP
XXX
0x0 = General Reset
XXX
periph_nreset
NRST (nrst_out)
EXTERNAL RESET LENGTH = 2 cycles

14.3.3.2 Wake-up Reset

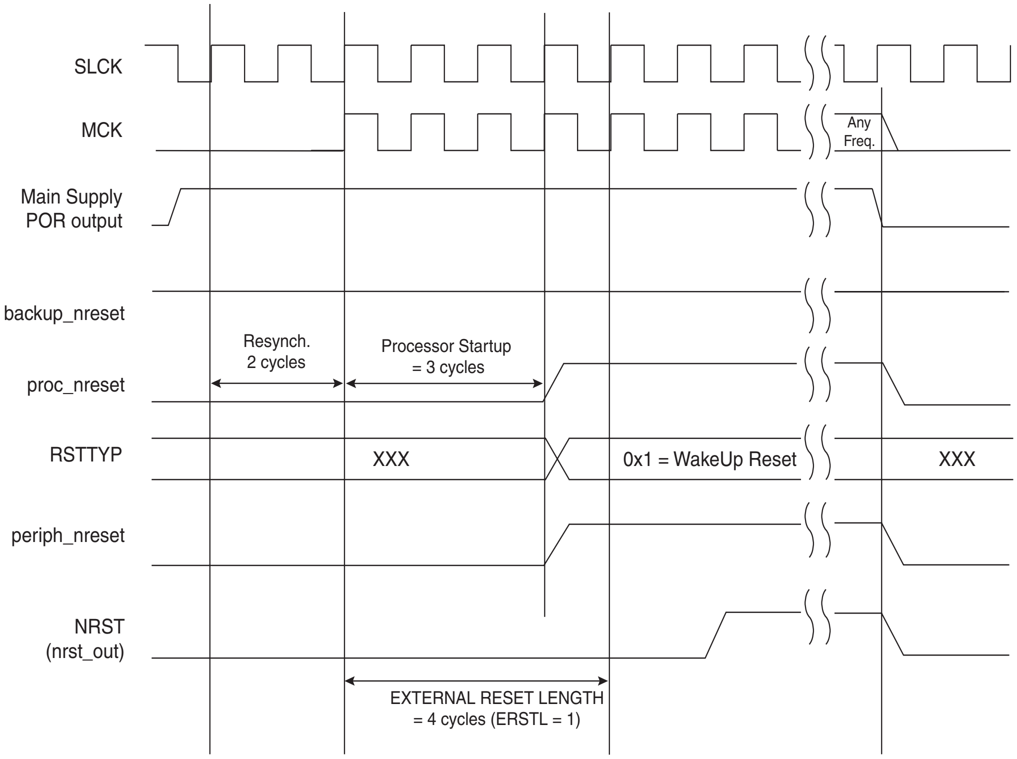
The Wake-up Reset occurs when the Main Supply is down. When the Main Supply POR output is active, all the reset signals are asserted except backup_nreset. When the Main Supply powers up, the POR output is resynchronized on Slow Clock. The processor clock is then re-enabled during 2 Slow Clock cycles, depending on the requirements of the ARM processor.

At the end of this delay, the processor and other reset signals rise. The field RSTTYP in RSTC_SR is updated to report a Wake-up Reset.

The “nrst_out” remains asserted for EXTERNAL_RESET_LENGTH cycles. As RSTC_MR is backed-up, the programmed number of cycles is applicable.

When the Main Supply is detected falling, the reset signals are immediately asserted. This transition is synchronous with the output of the Main Supply POR.

Figure 14-4. Wake-up State


SLCK
MCK
Any Freq.
Main Supply POR output
backup_nreset
Resynch. 2 cycles
Processor Startup = 3 cycles
proc_nreset
RSTTYP
XXX
0x1 = WakeUp Reset
XXX
periph_nreset
NRST (nrst_out)
EXTERNAL RESET LENGTH = 4 cycles (ERSTL = 1)

14.3.3.3 User Reset

The User Reset is entered when a low level is detected on the NRST pin and the bit URSTEN in RSTC_MR is at 1. The NRST input signal is resynchronized with SLCK to insure proper behavior of the system.

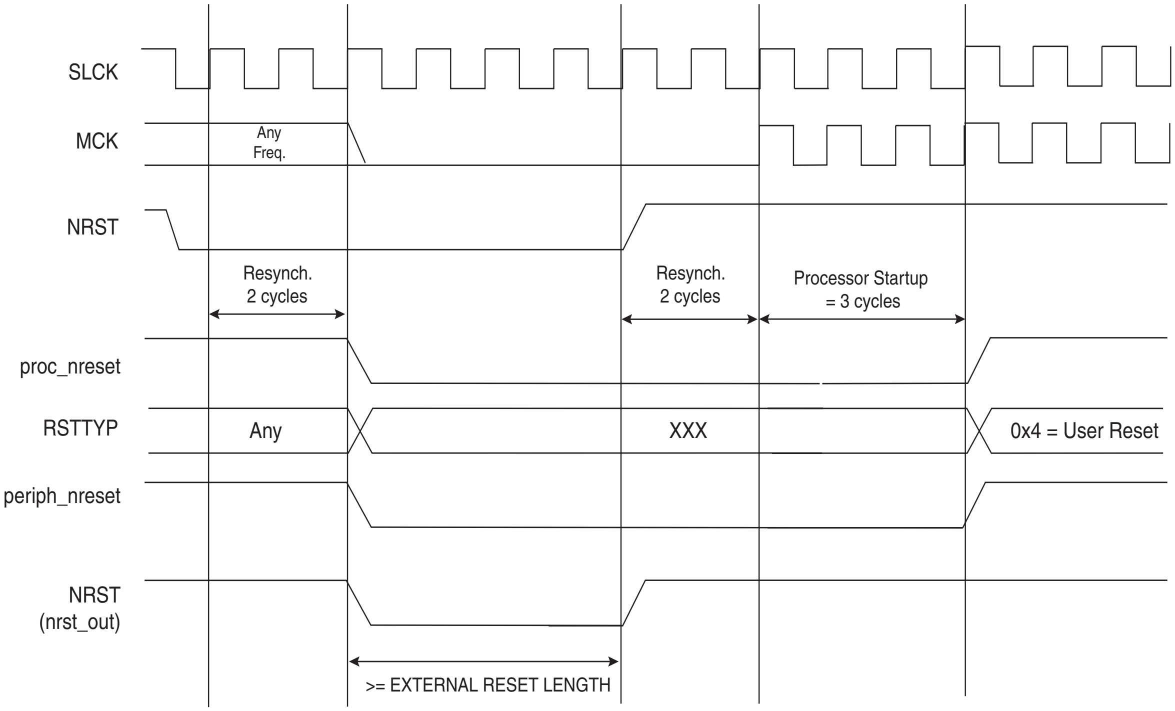
The User Reset is entered as soon as a low level is detected on NRST. The Processor Reset and the Peripheral Reset are asserted.

The User Reset is left when NRST rises, after a two-cycle resynchronization time and a three-cycle processor startup. The processor clock is re-enabled as soon as NRST is confirmed high.

When the processor reset signal is released, the RSTTYP field of the Status Register (RSTC_SR) is loaded with the value 0x4, indicating a User Reset.

The NRST Manager guarantees that the NRST line is asserted for EXTERNAL_RESET_LENGTH Slow Clock cycles, as programmed in the field ERSTL. However, if NRST does not rise after EXTERNAL_RESET_LENGTH because it is driven low externally, the internal reset lines remain asserted until NRST actually rises.

Figure 14-5. User Reset State


SLCK
MCK
Any
Freq.
NRST
Resynch.
2 cycles
Resynch.
2 cycles
Processor Startup
= 3 cycles
proc_nreset
RSTTYP
Any
XXX
0x4 = User Reset
periph_nreset
NRST
(nrst_out)
>= EXTERNAL RESET LENGTH

14.3.3.4 *Software Reset*

The Reset Controller offers several commands used to assert the different reset signals. These commands are performed by writing the Control Register (RSTC_CR) with the following bits at 1:

- PROCRST: Writing PROCRST at 1 resets the processor and the watchdog timer.
- PERRST: Writing PERRST at 1 resets all the embedded peripherals, including the memory system, and, in particular, the Remap Command. The Peripheral Reset is generally used for debug purposes.
- EXTRST: Writing EXTRST at 1 asserts low the NRST pin during a time defined by the field ERSTL in the Mode Register (RSTC_MR).

The software reset is entered if at least one of these bits is set by the software. All these commands can be performed independently or simultaneously. The software reset lasts 2 Slow Clock cycles.

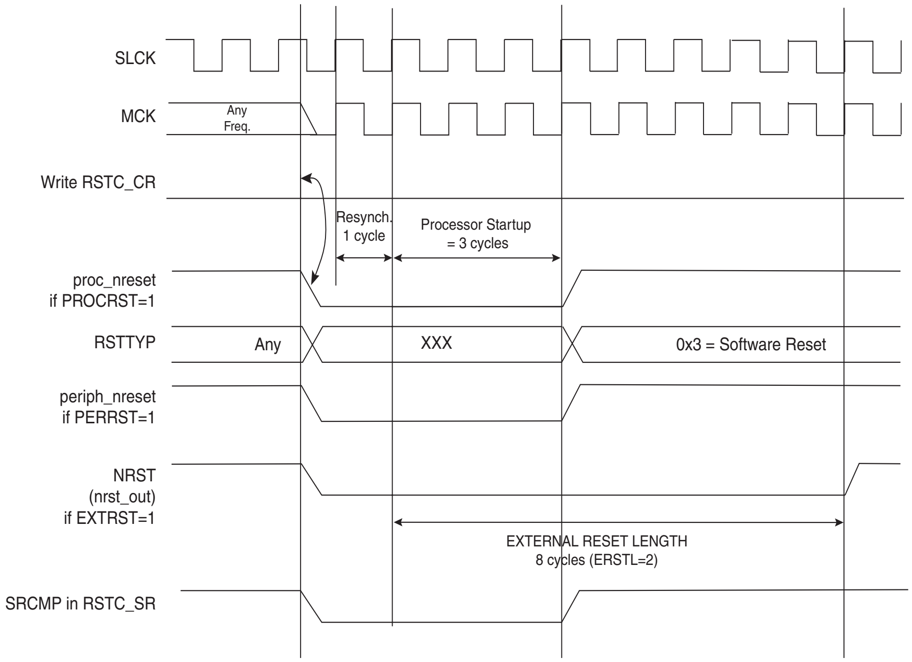
The internal reset signals are asserted as soon as the register write is performed. This is detected on the Master Clock (MCK). They are released when the software reset is left, i.e.; synchronously to SLCK.

If EXTRST is set, the nrst_out signal is asserted depending on the programming of the field ERSTL. However, the resulting falling edge on NRST does not lead to a User Reset.

If and only if the PROCRST bit is set, the Reset Controller reports the software status in the field RSTTYP of the Status Register (RSTC_SR). Other Software Resets are not reported in RSTTYP.

As soon as a software operation is detected, the bit SRCMP (Software Reset Command in Progress) is set in the Status Register (RSTC_SR). It is cleared as soon as the software reset is left. No other software reset can be performed while the SRCMP bit is set, and writing any value in RSTC_CR has no effect.

Figure 14-6. Software Reset


SLCK
MCK
Any Freq.
Write RSTC_CR
Resynch. 1 cycle
Processor Startup = 3 cycles
proc_nreset if PROCRST=1
RSTTYP
Any
XXX
0x3 = Software Reset
periph_nreset if PERRST=1
NRST (nrst_out) if EXTRST=1
EXTERNAL RESET LENGTH 8 cycles (ERSTL=2)
SRCMP in RSTC_SR

14.3.3.5 Watchdog Reset

The Watchdog Reset is entered when a watchdog fault occurs. This state lasts 2 Slow Clock cycles.

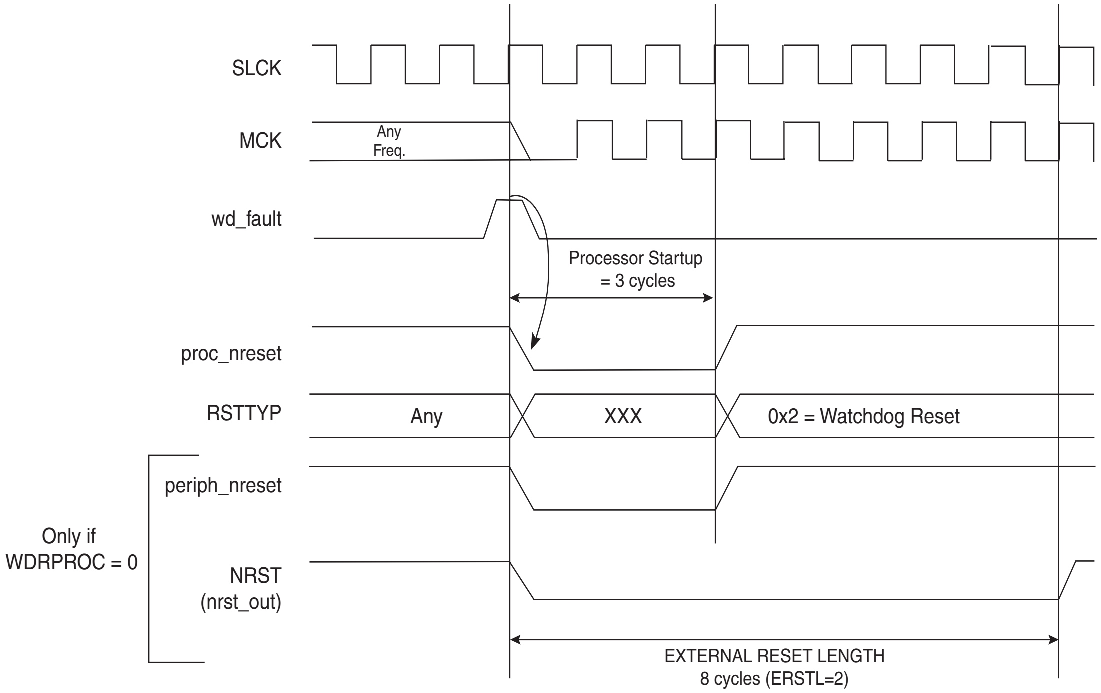
When in Watchdog Reset, assertion of the reset signals depends on the WDRPROC bit in WDT_MR:

- If WDRPROC is 0, the Processor Reset and the Peripheral Reset are asserted. The NRST line is also asserted, depending on the programming of the field ERSTL. However, the resulting low level on NRST does not result in a User Reset state.
- If WDRPROC = 1, only the processor reset is asserted.

The Watchdog Timer is reset by the proc_nreset signal. As the watchdog fault always causes a processor reset if WDRSTEN is set, the Watchdog Timer is always reset after a Watchdog Reset, and the Watchdog is enabled by default and with a period set to a maximum.

When the WDRSTEN in WDT_MR bit is reset, the watchdog fault has no impact on the reset controller.

Figure 14-7. Watchdog Reset


SLCK
MCK
Any
Freq.
wd_fault
Processor Startup
= 3 cycles
proc_nreset
RSTTYP
Any
XXX
0x2 = Watchdog Reset
periph_nreset
Only if
WDRPROC = 0
NRST
(nrst_out)
EXTERNAL RESET LENGTH
8 cycles (ERSTL=2)

14.3.4 Reset State Priorities

The Reset State Manager manages the following priorities between the different reset sources, given in descending order:

- Backup Reset
- Wake-up Reset
- Watchdog Reset
- Software Reset
- User Reset

Particular cases are listed below:

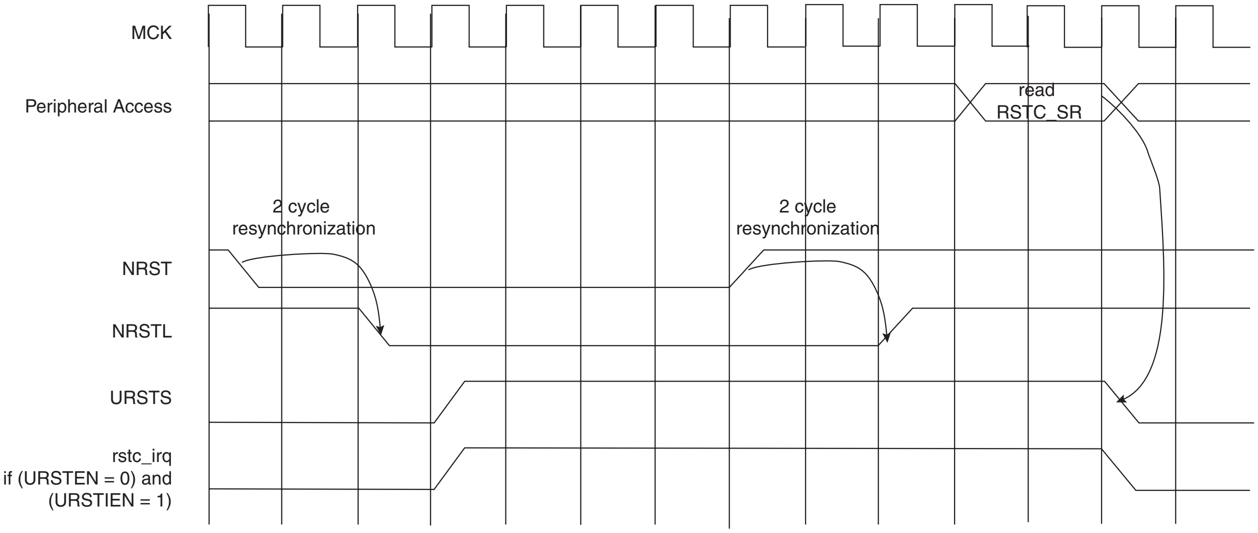
- When in User Reset:
  - A watchdog event is impossible because the Watchdog Timer is being reset by the proc_nreset signal.
  - A software reset is impossible, since the processor reset is being activated.
- When in Software Reset:
  - A watchdog event has priority over the current state.
  - The NRST has no effect.
- When in Watchdog Reset:
  - The processor reset is active and so a Software Reset cannot be programmed.
  - A User Reset cannot be entered.

14.3.5 Reset Controller Status Register

The Reset Controller status register (RSTC_SR) provides several status fields:

- RSTTYP field: This field gives the type of the last reset, as explained in previous sections.
- SRCMP bit: This field indicates that a Software Reset Command is in progress and that no further software reset should be performed until the end of the current one. This bit is automatically cleared at the end of the current software reset.
- NRSTL bit: The NRSTL bit of the Status Register gives the level of the NRST pin sampled on each MCK rising edge.
- URSTS bit: A high-to-low transition of the NRST pin sets the URSTS bit of the RSTC_SR register. This transition is also detected on the Master Clock (MCK) rising edge (see Figure 14-8). If the User Reset is disabled (URSTEN = 0) and if the interruption is enabled by the URSTIEN bit in the RSTC_MR register, the URSTS bit triggers an interrupt. Reading the RSTC_SR status register resets the URSTS bit and clears the interrupt.

Figure 14-8. Reset Controller Status and Interrupt



14.4 Reset Controller (RSTC) User Interface

Table 14-1. Reset Controller (RSTC) Register Mapping

Offset	Register	Name	Access	Reset Value	Back-up Reset Value
0x00	Control Register	RSTC_CR	Write-only	-	
0x04	Status Register	RSTC_SR	Read-only	0x0000_0001	0x0000_0000
0x08	Mode Register	RSTC_MR	Read/Write	-	0x0000_0000

Note: 1. The reset value of RSTC_SR either reports a General Reset or a Wake-up Reset depending on last rising power supply.

14.4.1 Reset Controller Control Register

Register Name: RSTC_CR

Access Type: Write-only

31	30	29	28	27	26	25	24
KEY							

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–		–

7	6	5	4	3	2	1	0
–	–	–	–	EXTRST	PERRST	–	PROCRST

- **PROCRST: Processor Reset**

0 = No effect.

1 = If KEY is correct, resets the processor.

- **PERRST: Peripheral Reset**

0 = No effect.

1 = If KEY is correct, resets the peripherals.

- **EXTRST: External Reset**

0 = No effect.

1 = If KEY is correct, asserts the NRST pin.

- **KEY: Password**

Should be written at value 0xA5. Writing any other value in this field aborts the write operation.

14.4.2 Reset Controller Status Register

Register Name: RSTC_SR

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	RSTTYP		

7	6	5	4	3	2	1	0
–	–	–	–	–	–		URSTS

- **URSTS: User Reset Status**

0 = No high-to-low edge on NRST happened since the last read of RSTC_SR.

1 = At least one high-to-low transition of NRST has been detected since the last read of RSTC_SR.

- **RSTTYP: Reset Type**

Reports the cause of the last processor reset. Reading this RSTC_SR does not reset this field.

RSTTYP			Reset Type	Comments
0	0	0	General Reset	Both VDDCORE and VDDBU rising
0	0	1	Wake Up Reset	VDDCORE rising
0	1	0	Watchdog Reset	Watchdog fault occurred
0	1	1	Software Reset	Processor reset required by the software
1	0	0	User Reset	NRST pin detected low

- **NRSTL: NRST Pin Level**

Registers the NRST Pin Level at Master Clock (MCK).

- **SRCMP: Software Reset Command in Progress**

0 = No software command is being performed by the reset controller. The reset controller is ready for a software command.

1 = A software reset command is being performed by the reset controller. The reset controller is busy.

14.4.3 Reset Controller Mode Register

Register Name: RSTC_MR

Access Type: Read/Write

31	30	29	28	27	26	25	24
KEY							

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

15	14	13	12	11	10	9	8
–	–	–	–	ERSTL			

7	6	5	4	3	2	1	0
–	–		URSTIEN	–	–	–	URSTEN

- **URSTEN: User Reset Enable**

0 = The detection of a low level on the pin NRST does not generate a User Reset.

1 = The detection of a low level on the pin NRST triggers a User Reset.

- **URSTIEN: User Reset Interrupt Enable**

0 = USRTS bit in RSTC_SR at 1 has no effect on rstc_irq.

1 = USRTS bit in RSTC_SR at 1 asserts rstc_irq if URSTEN = 0.

- **ERSTL: External Reset Length**

This field defines the external reset length. The external reset is asserted during a time of $2^{(ERSTL+1)}$ Slow Clock cycles. This allows assertion duration to be programmed between 60 µs and 2 seconds.

- **KEY: Password**

Should be written at value 0xA5. Writing any other value in this field aborts the write operation.

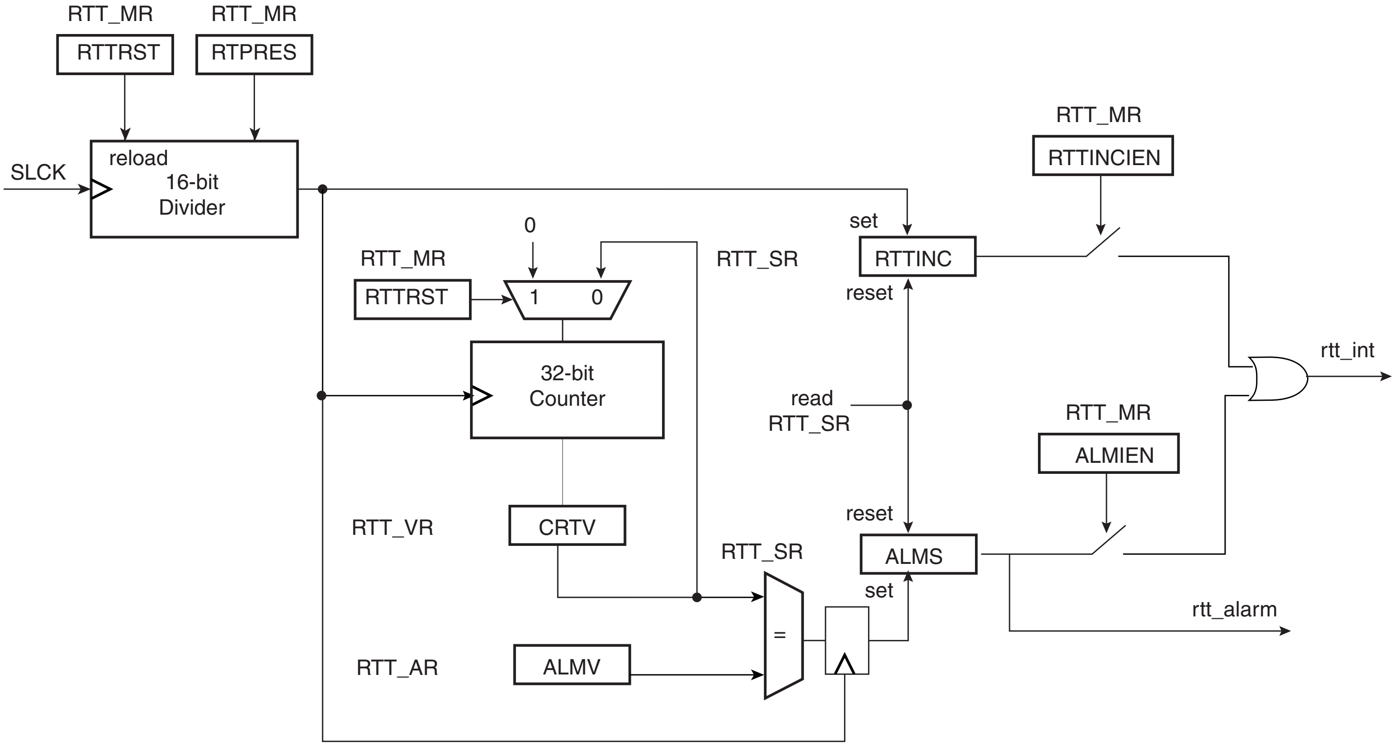
15. Real-time Timer (RTT)

15.1 Description

The Real-time Timer is built around a 32-bit counter and used to count elapsed seconds. It generates a periodic interrupt and/or triggers an alarm on a programmed value.

15.2 Block Diagram

Figure 15-1. Real-time Timer


RTT_MR
RTTRST
RTT_MR
RTPRES
SLCK
reload
16-bit
Divider
RTT_MR
RTTINCIEN
0
RTT_MR
RTTRST
1
0
RTT_SR
set
RTTINC
reset
32-bit
Counter
read
RTT_SR
rtt_int
RTT_MR
ALMIEN
RTT_VR
CRTV
reset
RTT_SR
ALMS
set
=
rtt_alarm
RTT_AR
ALMV

15.3 Functional Description

The Real-time Timer is used to count elapsed seconds. It is built around a 32-bit counter fed by Slow Clock divided by a programmable 16-bit value. The value can be programmed in the field RTPRES of the Real-time Mode Register (RTT_MR).

Programming RTPRES at 0x00008000 corresponds to feeding the real-time counter with a 1 Hz signal (if the Slow Clock is 32.768 Hz). The 32-bit counter can count up to 2^{32} seconds, corresponding to more than 136 years, then roll over to 0.

The Real-time Timer can also be used as a free-running timer with a lower time-base. The best accuracy is achieved by writing RTPRES to 3. Programming RTPRES to 1 or 2 is possible, but may result in losing status events because the status register is cleared two Slow Clock cycles after read. Thus if the RTT is configured to trigger an interrupt, the interrupt occurs during 2 Slow Clock cycles after reading RTT_SR. To prevent several executions of the interrupt handler, the interrupt must be disabled in the interrupt handler and re-enabled when the status register is clear.

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

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

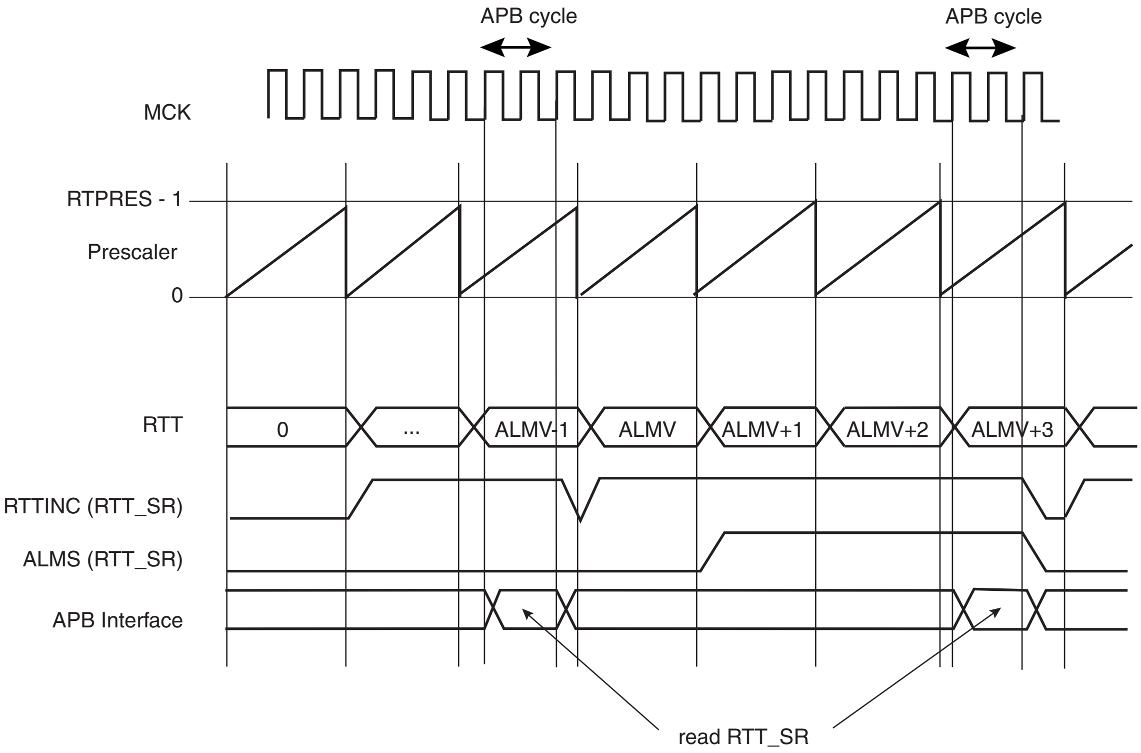
The bit RTTINC in RTT_SR is set each time the Real-time Timer counter is incremented. This bit can be used to start a periodic interrupt, the period being one second when the RTPRES is programmed with 0x8000 and Slow Clock equal to 32.768 Hz.

Reading the RTT_SR status register resets the RTTINC and ALMS fields.

Writing the bit RTTRST in RTT_MR immediately reloads and restarts the clock divider with the new programmed value. This also resets the 32-bit counter.

Note: Because of the asynchronism between the Slow Clock (SCLK) and the System Clock (MCK):
1) The restart of the counter and the reset of the RTT_VR current value register is effective only 2 slow clock cycles after the write of the RTTRST bit in the RTT_MR register.
2) The status register flags reset is taken into account only 2 slow clock cycles after the read of the RTT_SR (Status Register).

Figure 15-2. RTT Counting



15.4 Real-time Timer User Interface

15.4.1 Register Mapping

Table 15-1. Real-time Timer Register Mapping

Offset	Register	Name	Access	Reset Value
0x00	Mode Register	RTT_MR	Read/Write	0x0000_8000
0x04	Alarm Register	RTT_AR	Read/Write	0xFFFF_FFFF
0x08	Value Register	RTT_VR	Read-only	0x0000_0000
0x0C	Status Register	RTT_SR	Read-only	0x0000_0000

15.4.2 Real-time Timer Mode Register

Register Name: RTT_MR

Access Type: Read/Write

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

23	22	21	20	19	18	17	16
–	–	–	–	–	RTTRST	RTTINCIEN	ALMIEN

15	14	13	12	11	10	9	8
RTPRES							

7	6	5	4	3	2	1	0
RTPRES							

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

Defines the number of SLCK periods required to increment the Real-time timer. RTPRES is defined as follows:

RTPRES = 0: The prescaler period is equal to 2^{16}

RTPRES …0: The prescaler period is equal to RTPRES.

- **ALMIEN: Alarm Interrupt Enable**

0 = The bit ALMS in RTT_SR has no effect on interrupt.

1 = The bit ALMS in RTT_SR asserts interrupt.

- **RTTINCIEN: Real-time Timer Increment Interrupt Enable**

0 = The bit RTTINC in RTT_SR has no effect on interrupt.

1 = The bit RTTINC in RTT_SR asserts interrupt.

- **RTTRST: Real-time Timer Restart**

1 = Reloads and restarts the clock divider with the new programmed value. This also resets the 32-bit counter.

15.4.3 Real-time Timer Alarm Register

Register Name: RTT_AR

Access Type: Read/Write

31	30	29	28	27	26	25	24
ALMV							

23	22	21	20	19	18	17	16
ALMV							

15	14	13	12	11	10	9	8
ALMV							

7	6	5	4	3	2	1	0
ALMV							

- **ALMV: Alarm Value**

Defines the alarm value (ALMV+1) compared with the Real-time Timer.

15.4.4 Real-time Timer Value Register

Register Name: RTT_VR

Access Type: Read-only

31	30	29	28	27	26	25	24
CRTV							

23	22	21	20	19	18	17	16
CRTV							

15	14	13	12	11	10	9	8
CRTV							

7	6	5	4	3	2	1	0
CRTV							

- **CRTV: Current Real-time Value**

Returns the current value of the Real-time Timer.

15.4.5 Real-time Timer Status Register

Register Name: RTT_SR

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	RTTINC	ALMS

- **ALMS: Real-time Alarm Status**

0 = The Real-time Alarm has not occured since the last read of RTT_SR.

1 = The Real-time Alarm occured since the last read of RTT_SR.

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

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

1 = The Real-time Timer has been incremented since the last read of the RTT_SR.

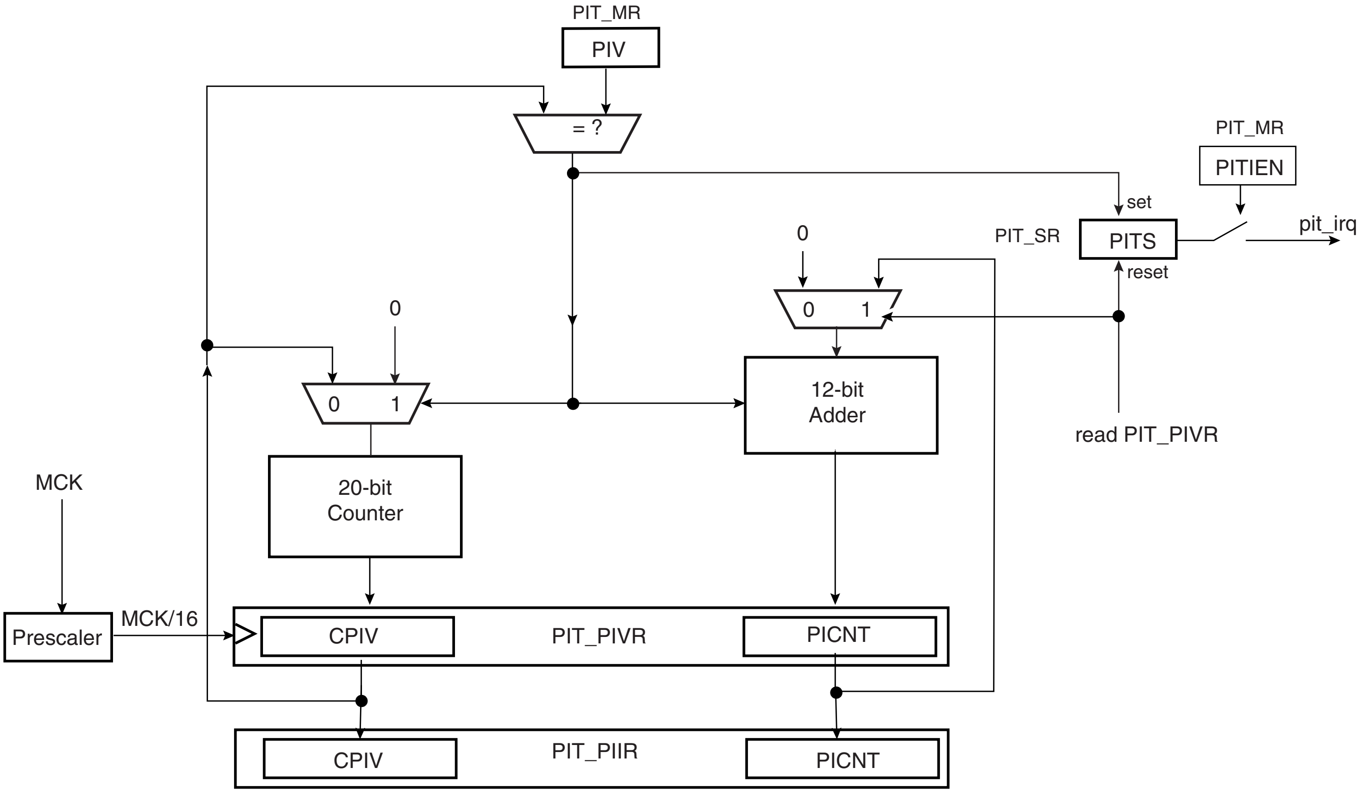
16. Periodic Interval Timer (PIT)

16.1 Description

The Periodic Interval Timer (PIT) provides the operating system's scheduler interrupt. It is designed to offer maximum accuracy and efficient management, even for systems with long response time.

16.2 Block Diagram

Figure 16-1. Periodic Interval Timer


PIT_MR
PIV
= ?
PIT_MR
PITIEN
set
PIT_SR
PITS
pit_irq
reset
0
0 1
12-bit Adder
read PIT_PIVR
0
0 1
20-bit Counter
MCK
Prescaler
MCK/16
CPIV
PIT_PIVR
PICNT
CPIV
PIT_PIIR
PICNT

16.3 Functional Description

The Periodic Interval Timer aims at providing periodic interrupts for use by operating systems.

The PIT provides a programmable overflow counter and a reset-on-read feature. It is built around two counters: a 20-bit CPIV counter and a 12-bit PICNT counter. Both counters work at Master Clock /16.

The first 20-bit CPIV counter increments from 0 up to a programmable overflow value set in the field PIV of the Mode Register (PIT_MR). When the counter CPIV reaches this value, it resets to 0 and increments the Periodic Interval Counter, PICNT. The status bit PITS in the Status Register (PIT_SR) rises and triggers an interrupt, provided the interrupt is enabled (PITIEN in PIT_MR).

Writing a new PIV value in PIT_MR does not reset/restart the counters.

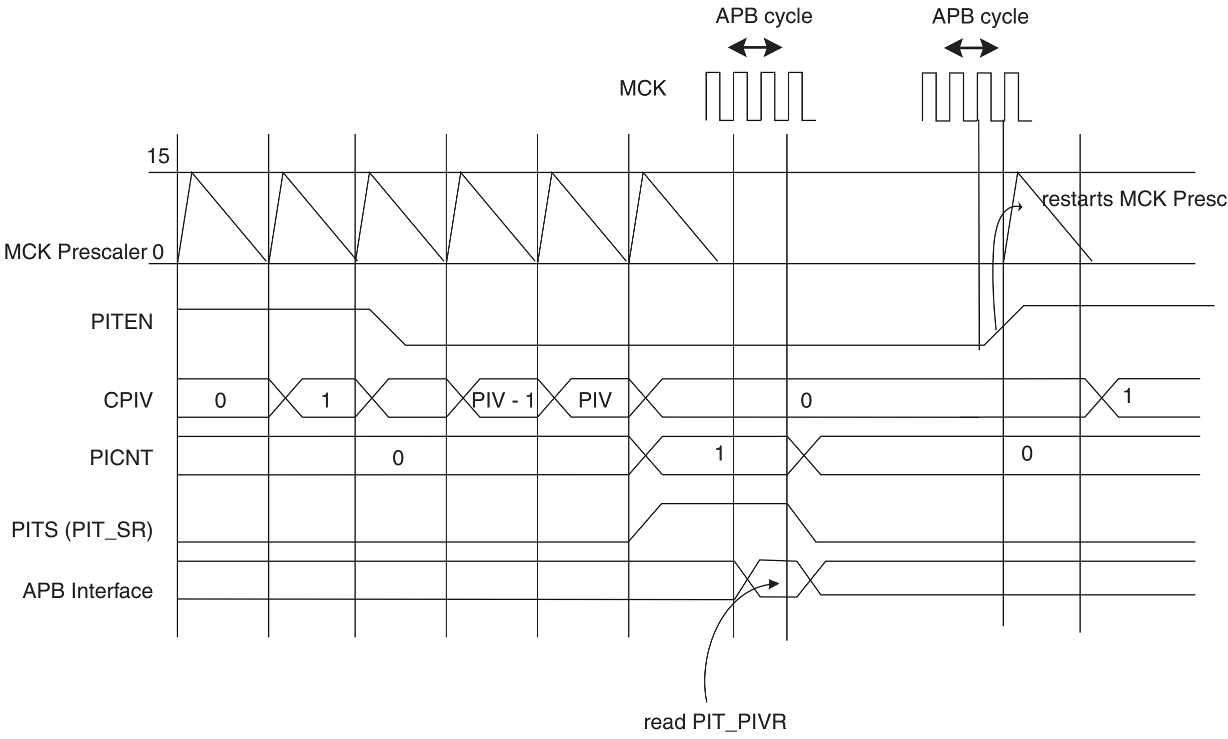
When CPIV and PICNT values are obtained by reading the Periodic Interval Value Register (PIT_PIVR), the overflow counter (PICNT) is reset and the PITS is cleared, thus acknowledging the interrupt. The value of PICNT gives the number of periodic intervals elapsed since the last read of PIT_PIVR.

When CPIV and PICNT values are obtained by reading the Periodic Interval Image Register (PIT_PIIR), there is no effect on the counters CPIV and PICNT, nor on the bit PITS. For example, a profiler can read PIT_PIIR without clearing any pending interrupt, whereas a timer interrupt clears the interrupt by reading PIT_PIVR.

The PIT may be enabled/disabled using the PITEN bit in the PIT_MR register (disabled on reset). The PITEN bit only becomes effective when the CPIV value is 0. Figure 16-2 illustrates the PIT counting. After the PIT Enable bit is reset (PITEN= 0), the CPIV goes on counting until the PIV value is reached, and is then reset. PIT restarts counting, only if the PITEN is set again.

The PIT is stopped when the core enters debug state.

Figure 16-2. Enabling/Disabling PIT with PITEN


APB cycle
APB cycle
MCK
15
MCK Prescaler 0
restarts MCK Prescaler
PITEN
CPIV
0
1
PIV - 1
PIV
0
1
PICNT
0
1
0
PITS (PIT_SR)
APB Interface
read PIT_PIVR

16.4 Periodic Interval Timer (PIT) User Interface

Table 16-1. Periodic Interval Timer (PIT) Register Mapping

Offset	Register	Name	Access	Reset Value
0x00	Mode Register	PIT_MR	Read/Write	0x000F_FFFF
0x04	Status Register	PIT_SR	Read-only	0x0000_0000
0x08	Periodic Interval Value Register	PIT_PIVR	Read-only	0x0000_0000
0x0C	Periodic Interval Image Register	PIT_PIIR	Read-only	0x0000_0000

16.4.1 Periodic Interval Timer Mode Register

Register Name: PIT_MR

Access Type: Read/Write

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

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

15	14	13	12	11	10	9	8
PIV							

7	6	5	4	3	2	1	0
PIV							

- **PIV: Periodic Interval Value**

Defines the value compared with the primary 20-bit counter of the Periodic Interval Timer (CPIV). The period is equal to (PIV + 1).

- **PITEN: Period Interval Timer Enabled**

0 = The Periodic Interval Timer is disabled when the PIV value is reached.

1 = The Periodic Interval Timer is enabled.

- **PITIEN: Periodic Interval Timer Interrupt Enable**

0 = The bit PITS in PIT_SR has no effect on interrupt.

1 = The bit PITS in PIT_SR asserts interrupt.

16.4.2 Periodic Interval Timer Status Register

Register Name: PIT_SR

Access Type: Read-only

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

- **PITS: Periodic Interval Timer Status**

0 = The Periodic Interval timer has not reached PIV since the last read of PIT_PIVR.

1 = The Periodic Interval timer has reached PIV since the last read of PIT_PIVR.

16.4.3 Periodic Interval Timer Value Register

Register Name: PIT_PIVR

Access Type: Read-only

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

Reading this register clears PITS in PIT_SR.

- **CPIV: Current Periodic Interval Value**

Returns the current value of the periodic interval timer.

- **PICNT: Periodic Interval Counter**

Returns the number of occurrences of periodic intervals since the last read of PIT_PIVR.

16.4.4 Periodic Interval Timer Image Register

Register Name: PIT_PIIR

Access Type: Read-only

31	30	29	28	27	26	25	24
PICNT							

23	22	21	20	19	18	17	16
PICNT				CPIV			

15	14	13	12	11	10	9	8
CPIV							

7	6	5	4	3	2	1	0
CPIV							

- **CPIV: Current Periodic Interval Value**

Returns the current value of the periodic interval timer.

- **PICNT: Periodic Interval Counter**

Returns the number of occurrences of periodic intervals since the last read of PIT_PIVR.

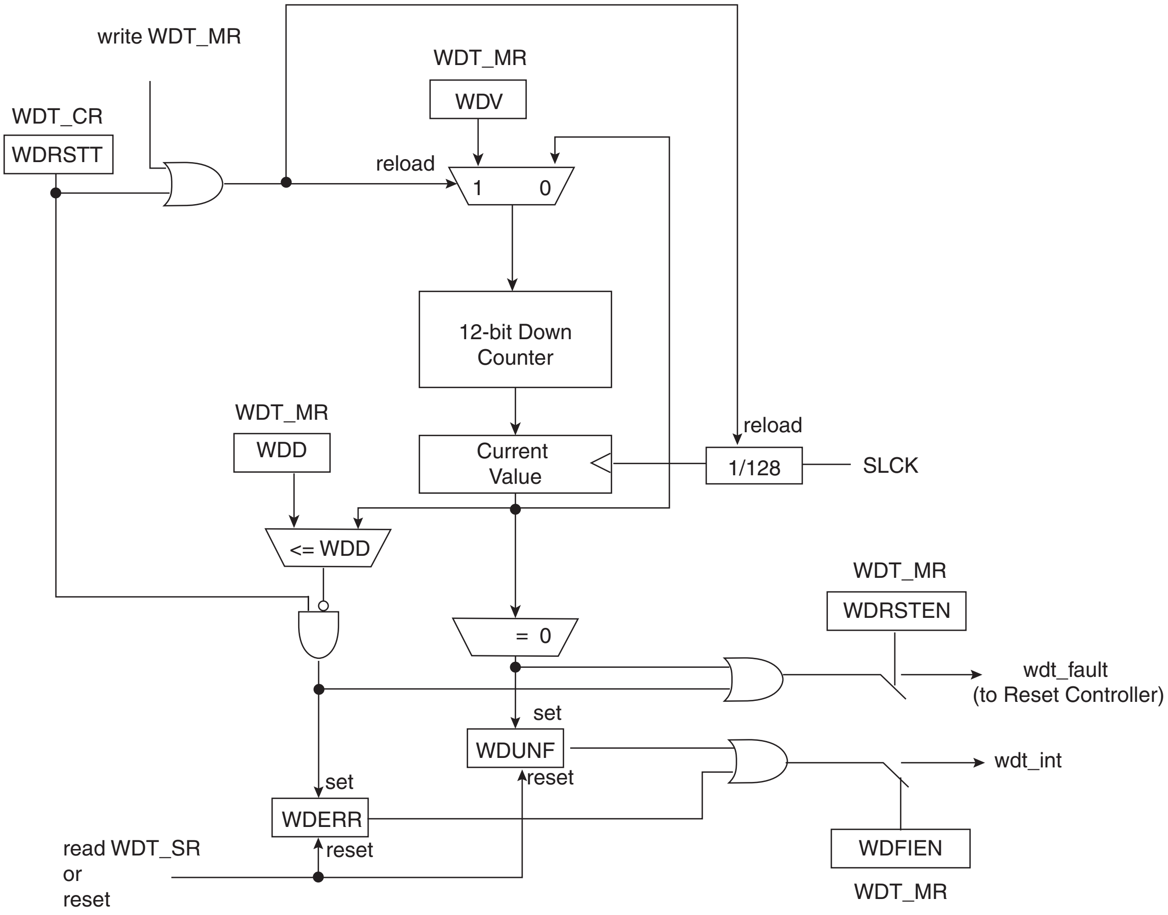
17. Watchdog Timer (WDT)

17.1 Description

The Watchdog Timer can be used to prevent system lock-up if the software becomes trapped in a deadlock. It features a 12-bit down counter that allows a watchdog period of up to 16 seconds (slow clock at 32.768 kHz). It can generate a general reset or a processor reset only. In addition, it can be stopped while the processor is in debug mode or idle mode.

17.2 Block Diagram

Figure 17-1. Watchdog Timer Block Diagram


write WDT_MR
WDT_MR
WDV
WDT_CR
WDRSTT
reload
1
0
12-bit Down Counter
WDT_MR
WDD
Current Value
reload
1/128
SLCK
<= WDD
WDT_MR
WDRSTEN
= 0
wdt_fault
(to Reset Controller)
set
WDUNF
reset
wdt_int
set
WDERR
reset
read WDT_SR or reset
WDFIEN
WDT_MR

17.3 Functional Description

The Watchdog Timer can be used to prevent system lock-up if the software becomes trapped in a deadlock. It is supplied with VDDCORE. It restarts with initial values on processor reset.

The Watchdog is built around a 12-bit down counter, which is loaded with the value defined in the field WDV of the Mode Register (WDT_MR). The Watchdog Timer uses the Slow Clock divided by 128 to establish the maximum Watchdog period to be 16 seconds (with a typical Slow Clock of 32.768 kHz).

After a Processor Reset, the value of WDV is 0xFFF, corresponding to the maximum value of the counter with the external reset generation enabled (field WDRSTEN at 1 after a Backup Reset). This means that a default Watchdog is running at reset, i.e., at power-up. The user must either disable it (by setting the WDDIS bit in WDT_MR) if he does not expect to use it or must reprogram it to meet the maximum Watchdog period the application requires.

The Watchdog Mode Register (WDT_MR) can be written only once. Only a processor reset resets it. Writing the WDT_MR register reloads the timer with the newly programmed mode parameters.

In normal operation, the user reloads the Watchdog at regular intervals before the timer underflow occurs, by writing the Control Register (WDT_CR) with the bit WDRSTT to 1. The Watchdog counter is then immediately reloaded from WDT_MR and restarted, and the Slow Clock 128 divider is reset and restarted. The WDT_CR register is write-protected. As a result, writing WDT_CR without the correct hard-coded key has no effect. If an underflow does occur, the “wdt_fault” signal to the Reset Controller is asserted if the bit WDRSTEN is set in the Mode Register (WDT_MR). Moreover, the bit WDUNF is set in the Watchdog Status Register (WDT_SR).

To prevent a software deadlock that continuously triggers the Watchdog, the reload of the Watchdog must occur while the Watchdog counter is within a window between 0 and WDD, WDD is defined in the WatchDog Mode Register WDT_MR.

Any attempt to restart the Watchdog while the Watchdog counter is between WDV and WDD results in a Watchdog error, even if the Watchdog is disabled. The bit WDERR is updated in the WDT_SR and the “wdt_fault” signal to the Reset Controller is asserted.

Note that this feature can be disabled by programming a WDD value greater than or equal to the WDV value. In such a configuration, restarting the Watchdog Timer is permitted in the whole range [0; WDV] and does not generate an error. This is the default configuration on reset (the WDD and WDV values are equal).

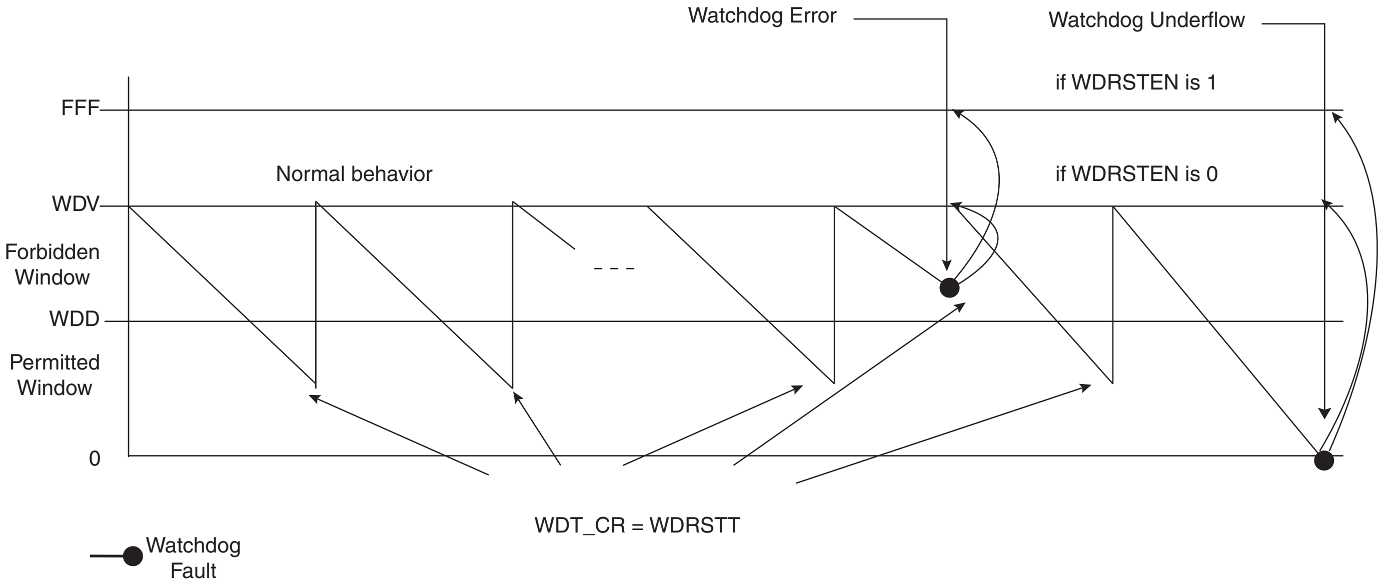
The status bits WDUNF (Watchdog Underflow) and WDERR (Watchdog Error) trigger an interrupt, provided the bit WDFIEN is set in the mode register. The signal “wdt_fault” to the reset controller causes a Watchdog reset if the WDRSTEN bit is set as already explained in the reset controller programmer Datasheet. In that case, the processor and the Watchdog Timer are reset, and the WDERR and WDUNF flags are reset.

If a reset is generated or if WDT_SR is read, the status bits are reset, the interrupt is cleared, and the “wdt_fault” signal to the reset controller is deasserted.

Writing the WDT_MR reloads and restarts the down counter.

While the processor is in debug state or in idle mode, the counter may be stopped depending on the value programmed for the bits WDIDLEHLT and WDDBGHLT in the WDT_MR.

Figure 17-2. Watchdog Behavior


Watchdog Error
Watchdog Underflow
if WDRSTEN is 1
FFF
Normal behavior
if WDRSTEN is 0
WDV
Forbidden Window
WDD
Permitted Window
0
– – –
WDT_CR = WDRSTT
Watchdog Fault

17.4 User Interface

17.4.1 Register Mapping

Table 17-1. Watchdog Timer Registers

Offset	Register	Name	Access	Reset Value
0x00	Control Register	WDT_CR	Write-only	-
0x04	Mode Register	WDT_MR	Read/Write Once	0x3FFF_2FFF
0x08	Status Register	WDT_SR	Read-only	0x0000_0000

17.4.2 Watchdog Timer Control Register

Register Name: WDT_CR

Access Type: Write-only

31	30	29	28	27	26	25	24
KEY							

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WDRSTT

- **WDRSTT: Watchdog Restart**

0: No effect.

1: Restarts the Watchdog.

- **KEY: Password**

Should be written at value 0xA5. Writing any other value in this field aborts the write operation.

17.4.3 Watchdog Timer Mode Register

Register Name: WDT_MR

Access Type: Read/Write Once

31	30	29	28	27	26	25	24
		WDIDLEHLT	WDDBGHLT	WDD			

23	22	21	20	19	18	17	16
WDD							

15	14	13	12	11	10	9	8
WDDIS	WDRPROC	WDRSTEN	WDFIEN	WDV			

7	6	5	4	3	2	1	0
WDV							

- **WDV: Watchdog Counter Value**

Defines the value loaded in the 12-bit Watchdog Counter.

- **WDFIEN: Watchdog Fault Interrupt Enable**

0: A Watchdog fault (underflow or error) has no effect on interrupt.

1: A Watchdog fault (underflow or error) asserts interrupt.

- **WDRSTEN: Watchdog Reset Enable**

0: A Watchdog fault (underflow or error) has no effect on the resets.

1: A Watchdog fault (underflow or error) triggers a Watchdog reset.

- **WDRPROC: Watchdog Reset Processor**

0: If WDRSTEN is 1, a Watchdog fault (underflow or error) activates all resets.

1: If WDRSTEN is 1, a Watchdog fault (underflow or error) activates the processor reset.

- **WDD: Watchdog Delta Value**

Defines the permitted range for reloading the Watchdog Timer.

If the Watchdog Timer value is less than or equal to WDD, writing WDT_CR with WDRSTT = 1 restarts the timer.

If the Watchdog Timer value is greater than WDD, writing WDT_CR with WDRSTT = 1 causes a Watchdog error.

- **WDDBGHLT: Watchdog Debug Halt**

0: The Watchdog runs when the processor is in debug state.

1: The Watchdog stops when the processor is in debug state.

- **WDIDLEHLT: Watchdog Idle Halt**

0: The Watchdog runs when the system is in idle mode.

1: The Watchdog stops when the system is in idle state.

- **WDDIS: Watchdog Disable**

0: Enables the Watchdog Timer.

1: Disables the Watchdog Timer.

17.4.4 Watchdog Timer Status Register

Register Name: WDT_SR

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	WDERR	WDUNF

- **WDUNF: Watchdog Underflow**

0: No Watchdog underflow occurred since the last read of WDT_SR.

1: At least one Watchdog underflow occurred since the last read of WDT_SR.

- **WDERR: Watchdog Error**

0: No Watchdog error occurred since the last read of WDT_SR.

1: At least one Watchdog error occurred since the last read of WDT_SR.

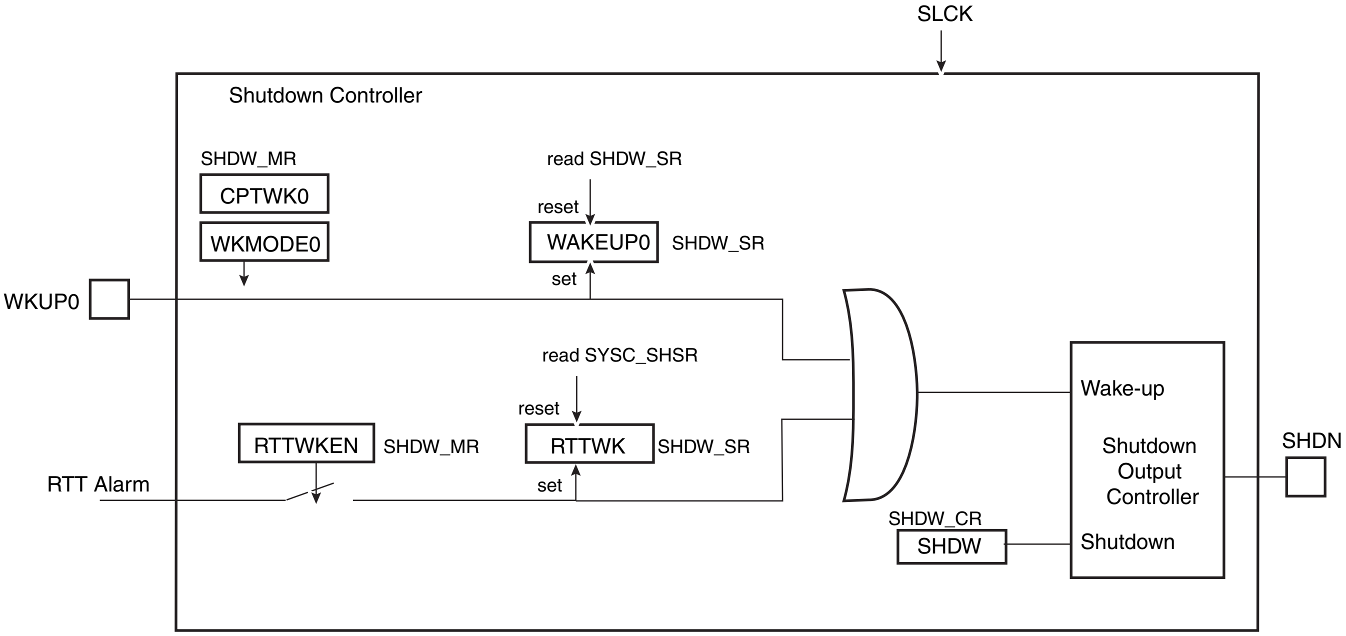
18. Shutdown Controller (SHDWC)

18.1 Description

The Shutdown Controller controls the power supplies VDDIO and VDDCORE and the wake-up detection on debounced input lines.

18.2 Block Diagram

Figure 18-1. Shutdown Controller Block Diagram


SLCK
Shutdown Controller
SHDW_MR
CPTWK0
WKMODE0
read SHDW_SR
reset
WAKEUP0
SHDW_SR
set
WKUP0
read SYSC_SHSR
reset
RTTWKEN
SHDW_MR
RTTWK
SHDW_SR
set
RTT Alarm
Wake-up
Shutdown Output Controller
Shutdown
SHDW_CR
SHDW
SHDN

18.3 I/O Lines Description

Table 18-1. I/O Lines Description

Name	Description	Type
WKUP0	Wake-up 0 input	Input
SHDN	Shutdown output	Output

18.4 Product Dependencies

18.4.1 Power Management

The Shutdown Controller is continuously clocked by Slow Clock. The Power Management Controller has no effect on the behavior of the Shutdown Controller.

18.5 Functional Description

The Shutdown Controller manages the main power supply. To do so, it is supplied with VDDBU and manages wake-up input pins and one output pin, SHDN.

A typical application connects the pin SHDN to the shutdown input of the DC/DC Converter providing the main power supplies of the system, and especially VDDCORE and/or VDDIO. The wake-up inputs (WKUP0) connect to any push-buttons or signal that wake up the system.

The software is able to control the pin SHDN by writing the Shutdown Control Register (SHDW_CR) with the bit SHDW at 1. The shutdown is taken into account only 2 slow clock cycles after the write of SHDW_CR. This register is password-protected and so the value written should contain the correct key for the command to be taken into account. As a result, the system should be powered down.

A level change on WKUP0 is used as wake-up. Wake-up is configured in the Shutdown Mode Register (SHDW_MR). The transition detector can be programmed to detect either a positive or negative transition or any level change on WKUP0. The detection can also be disabled. Programming is performed by defining WKMODE0.

Moreover, a debouncing circuit can be programmed for WKUP0. The debouncing circuit filters pulses on WKUP0 shorter than the programmed number of 16 SLCK cycles in CPTWK0 of the SHDW_MR register. If the programmed level change is detected on a pin, a counter starts. When the counter reaches the value programmed in the corresponding field, CPTWK0, the SHDN pin is released. If a new input change is detected before the counter reaches the corresponding value, the counter is stopped and cleared. WAKEUP0 of the Status Register (SHDW_SR) reports the detection of the programmed events on WKUP0 with a reset after the read of SHDW_SR.

The Shutdown Controller can be programmed so as to activate the wake-up using the RTT alarm (the detection of the rising edge of the RTT alarm is synchronized with SLCK). This is done by writing the SHDW_MR register using the RTTWKEN fields. When enabled, the detection of the RTT alarm is reported in the RTTWK bit of the SHDW_SR Status register. It is reset after the read of SHDW_SR. When using the RTT alarm to wake up the system, the user must ensure that the RTT alarm status flag is cleared before shutting down the system. Otherwise, no rising edge of the status flag may be detected and the wake-up fails.

18.6 Shutdown Controller (SHDWC) User Interface

18.6.1 Register Mapping

Table 18-2. Shutdown Controller (SHDWC) Registers

Offset	Register	Name	Access	Reset Value
0x00	Shutdown Control Register	SHDW_CR	Write-only	-
0x04	Shutdown Mode Register	SHDW_MR	Read-Write	0x0000_0303
0x08	Shutdown Status Register	SHDW_SR	Read-only	0x0000_0000

18.6.2 Shutdown Control Register

Register Name: SHDW_CR

Access Type: Write-only

31	30	29	28	27	26	25	24
KEY							

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	SHDW

- **SHDW: Shutdown Command**

0 = No effect.

1 = If KEY is correct, asserts the SHDN pin.

- **KEY: Password**

Should be written at value 0xA5. Writing any other value in this field aborts the write operation.

18.6.3 Shutdown Mode Register

Register Name: SHDW_MR

Access Type: Read/Write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
CPTWK0				–	–	WKMODE0	

- **WKMODE0: Wake-up Mode 0**

Table 1.

WKMODE[1:0]		Wake-up Input Transition Selection
0	0	None. No detection is performed on the wake-up input
0	1	Low to high level
1	0	High to low level
1	1	Both levels change

- **CPTWK0: Counter on Wake-up 0**

Defines the number of 16 Slow Clock cycles, the level detection on the corresponding input pin shall last before the wake-up event occurs. Because of the internal synchronization of WKUP0, the SHDN pin is released (CPTWK x 16 + 1) Slow Clock cycles after the event on WKUP.

- **RTTWKEN: Real-time Timer Wake-up Enable**

0 = The RTT Alarm signal has no effect on the Shutdown Controller.

1 = The RTT Alarm signal forces the de-assertion of the SHDN pin.

18.6.4 Shutdown Status Register

Register Name: SHDW_SR

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WAKEUP0

- **WAKEUP0: Wake-up 0 Status**

0 = No wake-up event occurred on the corresponding wake-up input since the last read of SHDW_SR.

1 = At least one wake-up event occurred on the corresponding wake-up input since the last read of SHDW_SR.

- **RTTWK: Real-time Timer Wake-up**

0 = No wake-up alarm from the RTT occurred since the last read of SHDW_SR.

1 = At least one wake-up alarm from the RTT occurred since the last read of SHDW_SR.

19. Bus Matrix

19.1 Description

The Bus Matrix implements a multi-layer AHB, based on the AHB-Lite protocol, that enables parallel access paths between multiple AHB masters and slaves in a system, thus increasing the overall bandwidth. The Bus Matrix interconnects up to 4 AHB Masters to up to 8 AHB Slaves. The normal latency to connect a master to a slave is one cycle except for the default master of the accessed slave which is connected directly (zero cycle latency). The Bus Matrix user interface is compliant with ARM® Advance Peripheral Bus and provides 6 Special Function Registers (MATRIX_SFR) that allow the Bus Matrix to support application specific features.

19.2 Memory Mapping

The Bus Matrix provides one decoder for every AHB Master Interface. The decoder offers each AHB Master several memory mappings. In fact, depending on the product, each memory area may be assigned to several slaves. Booting at the same address while using different AHB slaves (i.e. external RAM, internal ROM or internal Flash, etc.) becomes possible.

The Bus Matrix user interface provides Master Remap Control Register (MATRIX_MRCR) that performs remap action for every master independently.

19.3 Special Bus Granting Mechanism

The Bus Matrix provides some speculative bus granting techniques in order to anticipate access requests from some masters. This mechanism reduces latency at first access of a burst or single transfer. This bus granting mechanism sets a different default master for every slave.

At the end of the current access, if no other request is pending, the slave remains connected to its associated default master. A slave can be associated with three kinds of default masters: no default master, last access master and fixed default master.

19.3.1 No Default Master

At the end of the current access, if no other request is pending, the slave is disconnected from all masters. No Default Master suits low-power mode.

19.3.2 Last Access Master

At the end of the current access, if no other request is pending, the slave remains connected to the last master that performed an access request.

19.3.3 Fixed Default Master

At the end of the current access, if no other request is pending, the slave connects to its fixed default master. Unlike last access master, the fixed master does not change unless the user modifies it by a software action (field FIXED_DEFMSTR of the related MATRIX_SCFG).

To change from one kind of default master to another, the Bus Matrix user interface provides the Slave Configuration Registers, one for each slave, that set a default master for each slave. The Slave Configuration Register contains two fields: DEFMSTR_TYPE and FIXED_DEFMSTR. The 2-bit DEFMSTR_TYPE field selects the default master type (no default, last access master, fixed default master), whereas the 4-bit FIXED_DEFMSTR field selects a fixed default master provided that DEFMSTR_TYPE is set to fixed default master. Please refer to the Bus Matrix user interface description.

19.4 Arbitration

The Bus Matrix provides an arbitration mechanism that reduces latency when conflict cases occur, i.e. when two or more masters try to access the same slave at the same time. One arbiter per AHB slave is provided, thus arbitrating each slave differently.

The Bus Matrix provides the user with the possibility of choosing between 2 arbitration types for each slave:

1. Round-Robin Arbitration (default)
2. Fixed Priority Arbitration

This choice is made via the field ARBT of the Slave Configuration Registers (MATRIX_SCFG).

Each algorithm may be complemented by selecting a default master configuration for each slave.

When a re-arbitration must be done, specific conditions apply. See Section 19.5 ”Arbitration Rules” on page 104.

19.5 Arbitration Rules

Each arbiter has the ability to arbitrate between two or more different master requests. In order to avoid burst breaking and also to provide the maximum throughput for slave interfaces, arbitration may only take place during the following cycles:

1. Idle Cycles: When a slave is not connected to any master or is connected to a master which is not currently accessing it.
2. Single Cycles: When a slave is currently doing a single access.
3. End of Burst Cycles: When the current cycle is the last cycle of a burst transfer. For defined length burst, predicted end of burst matches the size of the transfer but is managed differently for undefined length burst. See “Undefined Length Burst Arbitration” on page 104.
4. Slot Cycle Limit: When the slot cycle counter has reached the limit value indicating that the current master access is too long and must be broken. See “Slot Cycle Limit Arbitration” on page 105.

19.5.1 Undefined Length Burst Arbitration

In order to avoid long slave handling during undefined length bursts (INCR), the Bus Matrix provides specific logic in order to re-arbitrate before the end of the INCR transfer. A predicted end of burst is used as a defined length burst transfer and can be selected from among the following five possibilities:

1. Infinite: No predicted end of burst is generated and therefore INCR burst transfer will never be broken.
2. One beat bursts: Predicted end of burst is generated at each single transfer inside the INCP transfer.
3. Four beat bursts: Predicted end of burst is generated at the end of each four beat boundary inside INCR transfer.
4. Eight beat bursts: Predicted end of burst is generated at the end of each eight beat boundary inside INCR transfer.
5. Sixteen beat bursts: Predicted end of burst is generated at the end of each sixteen beat boundary inside INCR transfer.

This selection can be done through the field ULBT of the Master Configuration Registers (MATRIX_MCFG).

19.5.2 Slot Cycle Limit Arbitration

The Bus Matrix contains specific logic to break long accesses, such as very long bursts on a very slow slave (e.g., an external low speed memory). At the beginning of the burst access, a counter is loaded with the value previously written in the SLOT_CYCLE field of the related Slave Configuration Register (MATRIX_SCFG) and decreased at each clock cycle. When the counter reaches zero, the arbiter has the ability to re-arbitrate at the end of the current byte, half word or word transfer.

19.5.3 Round-Robin Arbitration

This algorithm allows the Bus Matrix arbiters to dispatch the requests from different masters to the same slave in a round-robin manner. If two or more master requests arise at the same time, the master with the lowest number is first serviced, then the others are serviced in a round-robin manner.

There are three round-robin algorithms implemented:

- Round-Robin arbitration without default master
- Round-Robin arbitration with last default master
- Round-Robin arbitration with fixed default master

19.5.3.1 Round-Robin Arbitration without Default Master

This is the main algorithm used by Bus Matrix arbiters. It allows the Bus Matrix to dispatch requests from different masters to the same slave in a pure round-robin manner. At the end of the current access, if no other request is pending, the slave is disconnected from all masters. This configuration incurs one latency cycle for the first access of a burst. Arbitration without default master can be used for masters that perform significant bursts.

19.5.3.2 Round-Robin Arbitration with Last Default Master

This is a biased round-robin algorithm used by Bus Matrix arbiters. It allows the Bus Matrix to remove the one latency cycle for the last master that accessed the slave. In fact, at the end of the current transfer, if no other master request is pending, the slave remains connected to the last master that performed the access. Other non privileged masters still get one latency cycle if they want to access the same slave. This technique can be used for masters that mainly perform single accesses.

19.5.3.3 Round-Robin Arbitration with Fixed Default Master

This is another biased round-robin algorithm. It allows the Bus Matrix arbiters to remove the one latency cycle for the fixed default master per slave. At the end of the current access, the slave remains connected to its fixed default master. Every request attempted by this fixed default master will not cause any latency whereas other non privileged masters will still get one latency cycle. This technique can be used for masters that mainly perform single accesses.

19.5.4 Fixed Priority Arbitration

This algorithm allows the Bus Matrix arbiters to dispatch the requests from different masters to the same slave by using the fixed priority defined by the user. If two or more master requests are active at the same time, the master with the highest priority number is serviced first. If two or more master requests with the same priority are active at the same time, the master with the highest number is serviced first.

For each slave, the priority of each master may be defined through the Priority Registers for Slaves (MATRIX_PRAS and MATRIX_PRBS).

19.6 AHB Generic Bus Matrix User Interface

Table 19-1. Register Mapping

Offset	Register	Name	Access	Reset Value
0x0000	Master Configuration Register 0	MATRIX_MCFG0	Read/Write	0x00000002
0x0004	Master Configuration Register 1	MATRIX_MCFG1	Read/Write	0x00000002
0x0008	Master Configuration Register 2	MATRIX_MCFG2	Read/Write	0x00000002
0x000C	Master Configuration Register 3	MATRIX_MCFG3	Read/Write	0x00000002
0x0010 - 0x0014	Unused Master Configuration Registers	-	-	-
0x0018 - 0x003C	Reserved	-	-	-
0x0040	Slave Configuration Register 0	MATRIX_SCFG0	Read/Write	0x00000010
0x0044	Slave Configuration Register 1	MATRIX_SCFG1	Read/Write	0x00000010
0x0048	Slave Configuration Register 2	MATRIX_SCFG2	Read/Write	0x00000010
0x004C	Slave Configuration Register 3	MATRIX_SCFG3	Read/Write	0x00000010
0x0050	Slave Configuration Register 4	MATRIX_SCFG4	Read/Write	0x00000010
0x0054	Slave Configuration Register 5	MATRIX_SCFG5	Read/Write	0x00000010
0x0058	Slave Configuration Register 6	MATRIX_SCFG6	Read/Write	0x00000010
0x005C	Unused - Slave Configuration Register 7	-	-	-
0x0060	Slave Configuration Register 8	MATRIX_SCFG8	Read/Write	0x00000010
0x0064	Slave Configuration Register 9	MATRIX_SCFG9	Read/Write	0x00000010
0x0068 - 0x007C	Reserved	-	-	-
0x0080	Priority Register A for Slave 0	MATRIX_PRAS0	Read/Write	0x00000000
0x0084	Priority Register B for Slave 0	MATRIX_PRBS0	Read/Write	0x00000000
0x0088	Priority Register A for Slave 1	MATRIX_PRAS1	Read/Write	0x00000000
0x008C	Priority Register B for Slave 1	MATRIX_PRBS1	Read/Write	0x00000000
0x0090	Priority Register A for Slave 2	MATRIX_PRAS2	Read/Write	0x00000000
0x0094	Priority Register B for Slave 2	MATRIX_PRBS2	Read/Write	0x00000000
0x0098	Priority Register A for Slave 3	MATRIX_PRAS3	Read/Write	0x00000000
0x009C	Priority Register B for Slave 3	MATRIX_PRBS3	Read/Write	0x00000000
0x00A0	Priority Register A for Slave 4	MATRIX_PRAS4	Read/Write	0x00000000
0x00A4	Priority Register B for Slave 4	MATRIX_PRBS4	Read/Write	0x00000000
0x00A8	Priority Register A for Slave 5	MATRIX_PRAS5	Read/Write	0x00000000
0x00AC	Priority Register B for Slave 5	MATRIX_PRBS5	Read/Write	0x00000000
0x00B0	Priority Register A for Slave 6	MATRIX_PRAS6	Read/Write	0x00000000
0x00B4	Priority Register B for Slave 6	MATRIX_PRBS6	Read/Write	0x00000000
0x00B8	Unused - Priority Register A for Slave 7	-	-	-
0x00BC	Unused - Priority Register B for Slave 7	-	-	-
0x00C0	Priority Register A for Slave 8	MATRIX_PRAS8	Read/Write	0x00000000
0x00C4	Priority Register B for Slave 8	MATRIX_PRBS8	Read/Write	0x00000000

Offset	Register	Name	Access	Reset Value
0x00C8	Priority Register A for Slave 9	MATRIX_PRAS9	Read/Write	0x00000000
0x00CC	Priority Register B for Slave 9	MATRIX_PRBS9	Read/Write	0x00000000
0x00D0 -0x00FC	Reserved	-	-	-
0x0100	Master Remap Control Register	MATRIX_MRCR	Read/Write	0x00000000
0x0104 - 0x012C	Reserved	–	–	–
0x0130	EBI Chip Select Assignment Register	MATRIX_EBICSA	Read/Write	0x00000000
0x0134	USB Pull-up Control Register	MATRIX_USBPCR	Read/Write	0x00000000
0x0138 - 0x01F8	Reserved	–	–	–

19.6.1 Bus Matrix Master Configuration Registers

Register Name: MATRIX_MCFG0...MATRIX_MCFG3

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
–	–	–	–	–	ULBT		

- **ULBT: Undefined Length Burst Type**

0: Infinite Length Burst

No predicted end of burst is generated and therefore INCR bursts coming from this master cannot be broken.

1: Single Access

The undefined length burst is treated as a succession of single accesses, allowing re-arbitration at each beat of the INCR burst.

2: Four Beat Burst

The undefined length burst is split into a four-beat burst, allowing re-arbitration at each four-beat burst end.

3: Eight Beat Burst

The undefined length burst is split into an eight-beat burst, allowing re-arbitration at each eight-beat burst end.

4: Sixteen Beat Burst

The undefined length burst is split into a sixteen-beat burst, allowing re-arbitration at each sixteen-beat burst end.

19.6.2 Bus Matrix Slave Configuration Registers

Register Name: MATRIX_SCFG0...MATRIX_SCFG9 (MATRIX_SCFG7 unused)

Access Type: Read/Write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
SLOT_CYCLE							

- **SLOT_CYCLE: Maximum Number of Allowed Cycles for a Burst**

When the SLOT_CYCLE limit is reached for a burst, it may be broken by another master trying to access this slave.

This limit has been placed to avoid locking a very slow slave when very long bursts are used.

This limit must not be very small. Unreasonably small values break every burst and the Bus Matrix arbitrates without performing any data transfer. 16 cycles is a reasonable value for SLOT_CYCLE.

- **DEFMSTR_TYPE: Default Master Type**

0: No Default Master

At the end of the current slave access, if no other master request is pending, the slave is disconnected from all masters.

This results in a one cycle latency for the first access of a burst transfer or for a single access.

1: Last Default Master

At the end of the current slave access, if no other master request is pending, the slave stays connected to the last master having accessed it.

This results in not having one cycle latency when the last master tries to access the slave again.

2: Fixed Default Master

At the end of the current slave access, if no other master request is pending, the slave connects to the fixed master the number that has been written in the FIXED_DEFMSTR field.

This results in not having one cycle latency when the fixed master tries to access the slave again.

- **FIXED_DEFMSTR: Fixed Default Master**

This is the number of the Default Master for this slave. Only used if DEFMSTR_TYPE is 2. Specifying the number of a master which is not connected to the selected slave is equivalent to setting DEFMSTR_TYPE to 0.

- **ARBT: Arbitration Type**

0: Round-Robin Arbitration

1: Fixed Priority Arbitration

2: Reserved

3: Reserved

19.6.3 Bus Matrix Priority Registers A For Slaves

Register Name: MATRIX_PRAS0...MATRIX_PRAS8 (MATRIX_PRAS7 unused)

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	M7PR		–	–	M6PR	

23	22	21	20	19	18	17	16
–	–	M5PR		–	–	M4PR	

15	14	13	12	11	10	9	8
–	–	M3PR		–	–	M2PR	

7	6	5	4	3	2	1	0
–	–	M1PR		–	–	M0PR	

- **MxPR: Master x Priority**

Fixed priority of Master x for accessing the selected slave. The higher the number, the higher the priority.

19.6.4 Bus Matrix Priority Registers B For Slaves

Register Name: MATRIX_PRBS0...MATRIX_PRBS8 (MATRIX_PRBS7 unused)

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	M15PR		–	–	M14PR	

23	22	21	20	19	18	17	16
–	–	M13PR		–	–	M12PR	

15	14	13	12	11	10	9	8
–	–	M11PR		–	–	M10PR	

7	6	5	4	3	2	1	0
–	–	M9PR		–	–	M8PR	

- **MxPR: Master x Priority**

Fixed priority of Master x for accessing the selected slave. The higher the number, the higher the priority.

19.6.5 Bus Matrix Master Remap Control Register

Register Name: MATRIX_MRCR

Access Type: Read/Write

Reset: 0x0000_0000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–5	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	RCB5	RCB4	RCB3	RCB2	RCB1	RCB0

- **RCB: Remap Command Bit for Master x**

0: Disable remapped address decoding for the selected Master

1: Enable remapped address decoding for the selected Master

19.6.6 EBI Chip Select Assignment Register

Register Name: MATRIX_EBICSA

Access Type: Read/Write

Reset: 0x0000_0000

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

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

15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	EBI_DBPUC

7	6	5	4	3	2	1	0
-	-	EBI_CS5A	EBI_CS4A	EBI_CS3A	-	EBI_CS1A	-

- **EBI_CS1A: EBI Chip Select 1 Assignment**

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

1 = EBI Chip Select 1 is assigned to the SDRAM Controller.

- **EBI_CS3A: EBI Chip Select 3 Assignment**

0 = EBI Chip Select 3 is only assigned to the Static Memory Controller and EBI_NCS3 behaves as defined by the SMC.

1 = EBI Chip Select 3 is assigned to the Static Memory Controller and the SmartMedia Logic is activated.

- **EBI_CS4A: EBI Chip Select 4 Assignment**

0 = EBI Chip Select 4 is only assigned to the Static Memory Controller and EBI_NCS4 behaves as defined by the SMC.

1 = EBI Chip Select 4 is assigned to the Static Memory Controller and the CompactFlash Logic (first slot) is activated.

- **EBI_CS5A: EBI Chip Select 5 Assignment**

0 = EBI Chip Select 5 is only assigned to the Static Memory Controller and EBI_NCS5 behaves as defined by the SMC.

1 = EBI Chip Select 5 is assigned to the Static Memory Controller and the CompactFlash Logic (second slot) is activated.

- **EBI_DBPUC: EBI Data Bus Pull-Up Configuration**

0 = EBI D0 - D15 Data Bus bits are internally pulled-up to the VDDIO power supply.

1 = EBI D0 - D15 Data Bus bits are not internally pulled-up.

19.6.7 Matrix USB Pad Pull-up Control Register

Register Name: MATRIX_USBPCR

Access Type: Read/Write

Reset: 0x0000_0000

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

- **PUP_IDLE: Pull-up Idle**

0: Pad pull-up set on higher resistance

1: Pad pull-up set on lower resistance

- **UDP_PUP_ON: UDP Pad Pull-up Enable**

0: Pad pull-up disabled

1: Pad pull-up enabled

20. External Bus Interface (EBI)

20.1 Overview

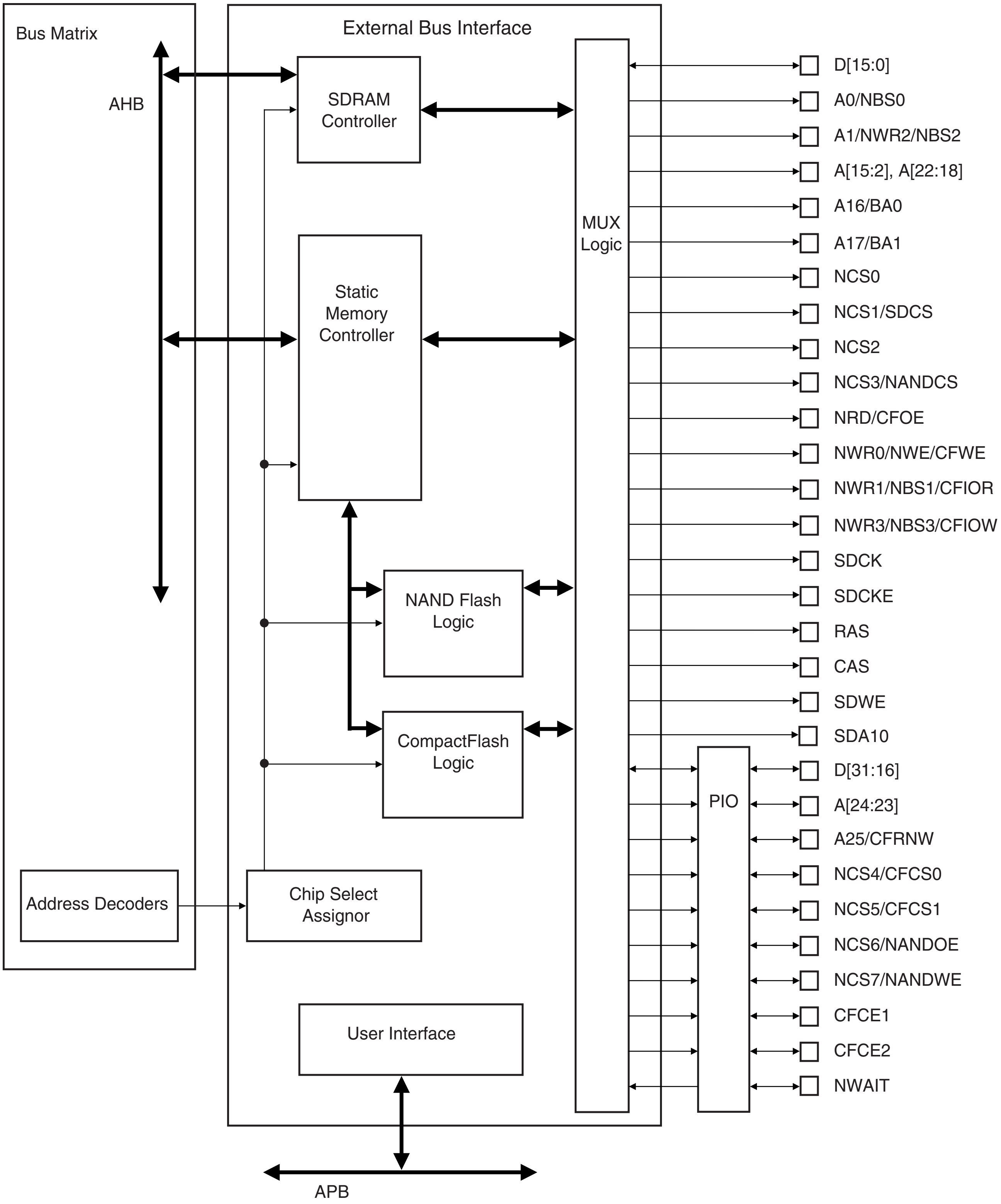
The External Bus Interface (EBI) is designed to ensure the successful data transfer between several external devices and the embedded Memory Controller of an ARM®-based device. The Static Memory and SDRAM Controllers are all featured external Memory Controllers on the EBI. These external Memory Controllers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

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

20.2 Block Diagram

Figure 20-1 shows the organization of the External Bus Interface.

Figure 20-1. Organization of the External Bus Interface


Bus Matrix
AHB
Address Decoders
External Bus Interface
SDRAM Controller
Static Memory Controller
NAND Flash Logic
CompactFlash Logic
Chip Select Assignor
User Interface
MUX Logic
PIO
APB
D[15:0]
A0/NBS0
A1/NWR2/NBS2
A[15:2], A[22:18]
A16/BA0
A17/BA1
NCS0
NCS1/SDCS
NCS2
NCS3/NANDCS
NRD/CFOE
NWR0/NWE/CFWE
NWR1/NBS1/CFIOR
NWR3/NBS3/CFIOW
SDCK
SDCKE
RAS
CAS
SDWE
SDA10
D[31:16]
A[24:23]
A25/CFRNW
NCS4/CFCS0
NCS5/CFCS1
NCS6/NANDOE
NCS7/NANDWE
CFCE1
CFCE2
NWAIT

20.3 I/O Lines Description

Table 20-1. I/O Lines Description

Name	Function	Type	Active Level
EBI			
D0 - D31	Data Bus	I/O	
A0 - A25	Address Bus	Output	
NWAIT	External Wait Signal	Input	Low
SMC			
NCS0 - NCS7	Chip Select Lines	Output	Low
NWR0 - NWR3	Write Signals	Output	Low
NRD	Read Signal	Output	Low
NWE	Write Enable	Output	Low
NBS0 - NBS3	Byte Mask Signals	Output	Low
EBI for CompactFlash Support			
CFCE1 - CFCE2	CompactFlash Chip Enable	Output	Low
CFOE	CompactFlash Output Enable	Output	Low
CFWE	CompactFlash Write Enable	Output	Low
CFIOR	CompactFlash I/O Read Signal	Output	Low
CFIOW	CompactFlash I/O Write Signal	Output	Low
CFRNW	CompactFlash Read Not Write Signal	Output	
CFCS0 - CFCS1	CompactFlash Chip Select Lines	Output	Low
EBI for NAND Flash Support			
NANDCS	NAND Flash Chip Select Line	Output	Low
NANDOE	NAND Flash Output Enable	Output	Low
NANDWE	NAND Flash Write Enable	Output	Low
SDRAM Controller			
SDCK	SDRAM Clock	Output	
SDCKE	SDRAM Clock Enable	Output	High
SDCS	SDRAM Controller Chip Select Line	Output	Low
BA0 - BA1	Bank Select	Output	
SDWE	SDRAM Write Enable	Output	Low
RAS - CAS	Row and Column Signal	Output	Low
NWR0 - NWR3	Write Signals	Output	Low
NBS0 - NBS3	Byte Mask Signals	Output	Low
SDA10	SDRAM Address 10 Line	Output	

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

Table 20-2 on page 118 details the connections between the two Memory Controllers and the EBI pins.

Table 20-2. EBI Pins and Memory Controllers I/O Lines Connections

EBI Pins	SDRAMC I/O Lines	SMC I/O Lines
NWR1/NBS1/CFIOR	NBS1	NWR1/NUB
A0/NBS0	Not Supported	SMC_A0/NLB
A1/NBS2/NWR2	Not Supported	SMC_A1
A[11:2]	SDRAMC_A[9:0]	SMC_A[11:2]
SDA10	SDRAMC_A10	Not Supported
A12	Not Supported	SMC_A12
A[14:13]	SDRAMC_A[12:11]	SMC_A[14:13]
A[25:15]	Not Supported	SMC_A[25:15]
D[31:16]	D[31:16]	D[31:16]
D[15:0]	D[15:0]	D[15:0]

20.4 Application Example

20.4.1 Hardware Interface

Table 20-3 and Table 20-4 detail the connections to be applied between the EBI pins and the external devices for each Memory Controller.

Table 20-3. EBI Pins and External Static Devices Connections

	Pins of the Interfaced Device					
Pins	8-bit Static Device	2 x 8-bit Static Devices	16-bit Static Device	4 x 8-bit Static Devices	2 x 16-bit Static Devices	32-bit Static Device
Controller	**SMC**					
D0 - D7	D0 - D7	D0 - D7	D0 - D7	D0 - D7	D0 - D7	D0 - D7
D8 - D15	–	D8 - D15	D8 - D15	D8 - D15	D8 - 15	D8 - 15
D16 - D23	–	–	–	D16 - D23	D16 - D23	D16 - D23
D24 - D31	–	–	–	D24 - D31	D24 - D31	D24 - D31
A0/NBS0	A0	–	NLB	–	NLB[3]	BE0[5]
A1/NWR2/NBS2	A1	A0	A0	WE[2]	NLB[4]	BE2[5]
A2 - A25	A[2:25]	A[1:24]	A[1:24]	A[0:23]	A[0:23]	A[0:23]
NCS0	CS	CS	CS	CS	CS	CS
NCS1/SDCS	CS	CS	CS	CS	CS	CS
NCS2	CS	CS	CS	CS	CS	CS
NCS3/NANDCS	CS	CS	CS	CS	CS	CS
NCS4/CFCS0	CS	CS	CS	CS	CS	CS

Table 20-3. EBI Pins and External Static Devices Connections (Continued)

Pins	Pins of the Interfaced Device					
	8-bit Static Device	2 x 8-bit Static Devices	16-bit Static Device	4 x 8-bit Static Devices	2 x 16-bit Static Devices	32-bit Static Device
Controller	**SMC**					
NCS5/CFCS1	CS	CS	CS	CS	CS	CS
NCS6/NANDOE	CS	CS	CS	CS	CS	CS
NCS7/NANDWE	CS	CS	CS	CS	CS	CS
NRD/CFOE	OE	OE	OE	OE	OE	OE
NWR0/NWE	WE	WE(1)	WE	WE(2)	WE	WE
NWR1/NBS1	–	WE(1)	NUB	WE(2)	NUB(3)	BE1(5)
NWR3/NBS3	–	–	–	WE(2)	NUB(4)	BE3(5)

Notes:
1. NWR1 enables upper byte writes. NWR0 enables lower byte writes.
2. NWRx enables corresponding byte x writes. (x = 0, 1, 2 or 3)
3. NBS0 and NBS1 enable respectively lower and upper bytes of the lower 16-bit word.
4. NBS2 and NBS3 enable respectively lower and upper bytes of the upper 16-bit word.
5. BEx: Byte x Enable (x = 0,1,2 or 3)

Table 20-4. EBI Pins and External Devices Connections

Pins	Pins of the Interfaced Device			
	SDRAM	Compact Flash	Compact Flash True IDE Mode	NAND Flash
Controller	**SDRAMC**	**SMC**		
D0 - D7	D0 - D7	D0 - D7	D0 - D7	AD0-AD7
D8 - D15	D8 - D15	D8 - 15	D8 - 15	AD8-AD15
D16 - D31	D16 - D31	–	–	–
A0/NBS0	DQM0	A0	A0	–
A1/NWR2/NBS2	DQM2	A1	A1	–
A2 - A10	A[0:8]	A[2:10]	A[2:10]	–
A11	A9	–	–	–
SDA10	A10	–	–	–
A12	–	–	–	–
A13 - A14	A[11:12]	–	–	–
A15	–	–	–	–
A16/BA0	BA0	–	–	–
A17/BA1	BA1	–	–	–
A18 - A20	–	–	–	–
A21	–	–	–	ALE(3)
A22	–	REG	REG	CLE(3)

Table 20-4. EBI Pins and External Devices Connections (Continued)

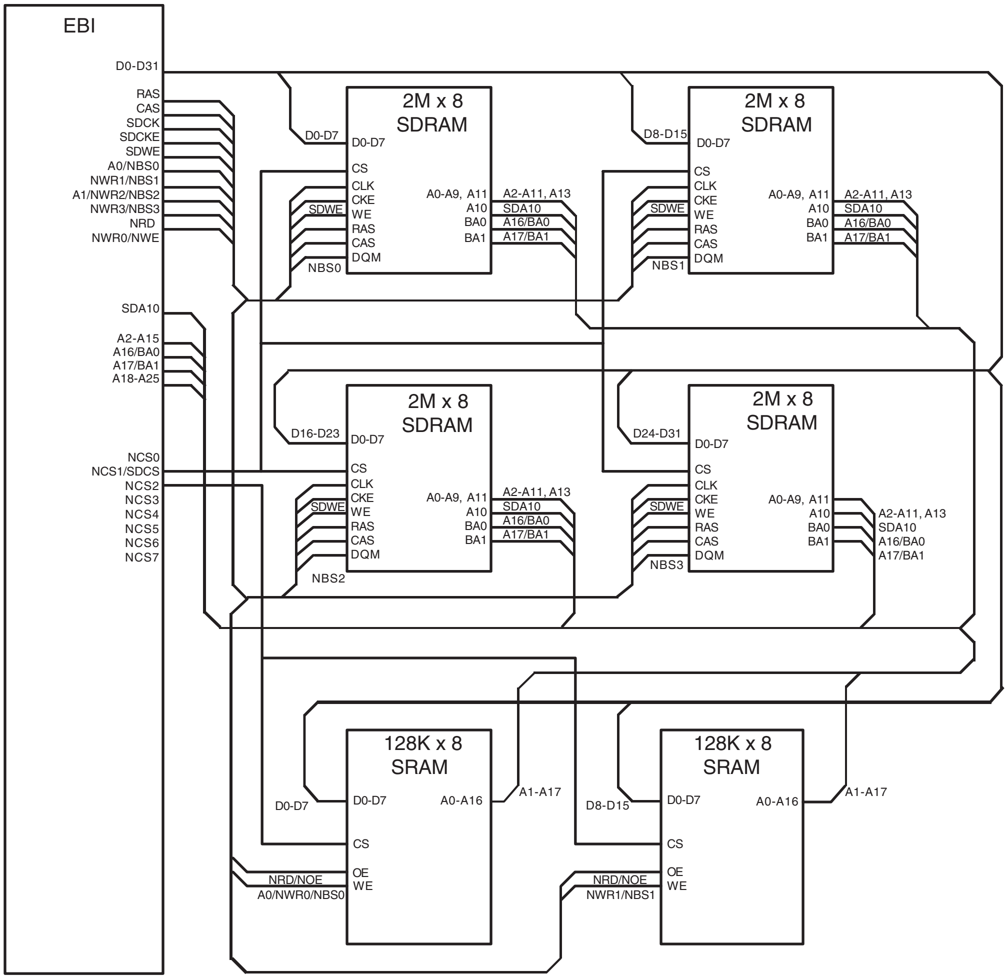
	Pins of the Interfaced Device			
Pins	SDRAM	Compact Flash	Compact Flash True IDE Mode	NAND Flash
Controller	**SDRAMC**	**SMC**		
A23 - A24	–	–	–	–
A25	–	CFRNW(1)	CFRNW(1)	–
NCS0	–	–	–	–
NCS1/SDCS	CS	–	–	–
NCS2	–	–	–	–
NCS3/NANDCS	–	–	–	–
NCS4/CFCS0	–	CFCS0(1)	CFCS0(1)	–
NCS5/CFCS1	–	CFCS1(1)	CFCS1(1)	–
NCS6/NANDOE	–	–	–	OE
NCS7/NANDWE	–	–	–	WE
NRD/CFOE	–	OE	–	–
NWR0/NWE/CFWE	–	WE	WE	–
NWR1/NBS1/CFIOR	DQM1	IOR	IOR	–
NWR3/NBS3/CFIOW	DQM3	IOW	IOW	–
CFCE1	–	CE1	CS0	–
CFCE2	–	CE2	CS1	–
SDCK	CLK	–	–	–
SDCKE	CKE	–	–	–
RAS	RAS	–	–	–
CAS	CAS	–	–	–
SDWE	WE	–	–	–
NWAIT	–	WAIT	WAIT	–
Pxx(2)	–	CD1 or CD2	CD1 or CD2	–
Pxx(2)	–	–	–	CE
Pxx(2)	–	–	–	RDY

Note:
1. Not directly connected to the CompactFlash slot. Permits the control of the bidirectional buffer between the EBI data bus and the CompactFlash slot.
2. Any PIO line.
3. The CLE and ALE signals of the NAND Flash device may be driven by any address bit. For details, see "NAND Flash Support" on page 127.

20.4.2 Connection Examples

Figure 20-2 shows an example of connections between the EBI and external devices.

Figure 20-2. EBI Connections to Memory Devices


EBI
D0-D31
RAS
CAS
SDCK
SDCKE
SDWE
A0/NBS0
NWR1/NBS1
A1/NWR2/NBS2
NWR3/NBS3
NRD
NWR0/NWE
SDA10
A2-A15
A16/BA0
A17/BA1
A18-A25
NCS0
NCS1/SDCS
NCS2
NCS3
NCS4
NCS5
NCS6
NCS7
2M x 8
SDRAM
D0-D7
CS
CLK
CKE
WE
RAS
CAS
DQM
A0-A9, A11
A10
BA0
BA1
A2-A11, A13
SDA10
A16/BA0
A17/BA1
SDWE
NBS0
NBS1
NBS2
NBS3
D8-D15
D16-D23
D24-D31
128K x 8
SRAM
A0-A16
A1-A17
OE
NRD/NOE
A0/NWR0/NBS0
NWR1/NBS1

20.5 Product Dependencies

20.5.1 I/O Lines

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

20.6 Functional Description

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

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

20.6.1 Bus Multiplexing

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

Multiplexing is specifically organized in order to guarantee the maintenance of the address and output control lines at a stable state while no external access is being performed. Multiplexing is also designed to respect the data float times defined in the Memory Controllers. Furthermore, refresh cycles of the SDRAM are executed independently by the SDRAM Controller without delaying the other external Memory Controller accesses.

20.6.2 Pull-up Control

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

20.6.3 Static Memory Controller

For information on the Static Memory Controller, refer to the Static Memory Controller section.

20.6.4 SDRAM Controller

For information on the SDRAM Controller, refer to the SDRAM section.

20.6.5 CompactFlash Support

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

The CompactFlash logic is driven by the Static Memory Controller (SMC) on the NCS4 and/or NCS5 address space. Programming the CS4A and/or CS5A bit of the CSA Register to the appropriate value enables this logic. For details on this register, refer to the Bus Matrix User Interface section. Access to an external CompactFlash device is then made by accessing the address space reserved to NCS4 and/or NCS5 (i.e., between 0x5000 0000 and 0x5FFF FFFF for NCS4 and between 0x6000 0000 and 0x6FFF FFFF for NCS5).

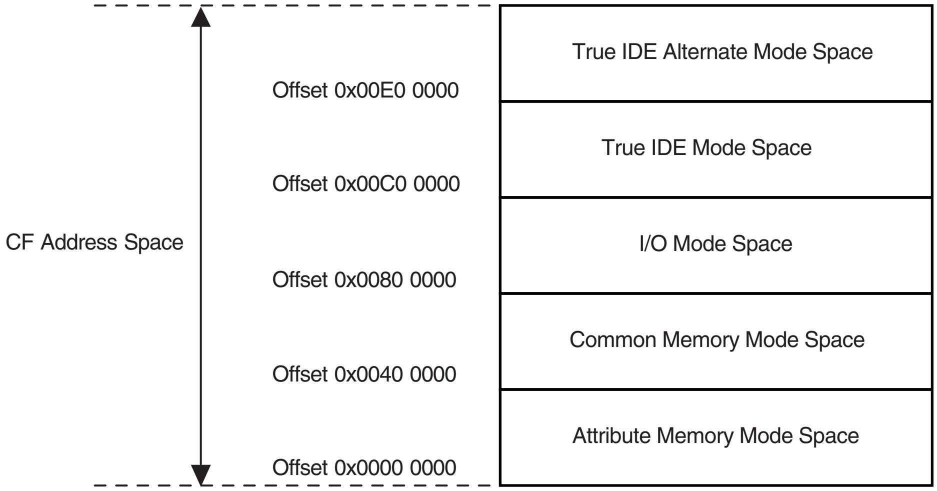
All CompactFlash modes (Attribute Memory, Common Memory, I/O and True IDE) are supported but the signals _IOIS16 (I/O and True IDE modes) and _ATA SEL (True IDE mode) are not handled.

20.6.5.1 *I/O Mode, Common Memory Mode, Attribute Memory Mode and True IDE Mode*

Within the NCS4 and/or NCS5 address space, the current transfer address is used to distinguish I/O mode, common memory mode, attribute memory mode and True IDE mode.

The different modes are accessed through a specific memory mapping as illustrated on Figure 20-3. A[23:21] bits of the transfer address are used to select the desired mode as described in Table 20-5 on page 123.

Figure 20-3. CompactFlash Memory Mapping


True IDE Alternate Mode Space
Offset 0x00E0 0000
True IDE Mode Space
Offset 0x00C0 0000
CF Address Space
I/O Mode Space
Offset 0x0080 0000
Common Memory Mode Space
Offset 0x0040 0000
Attribute Memory Mode Space
Offset 0x0000 0000

Note: The A22 pin of the EBI is used to drive the REG signal of the CompactFlash Device (except in True IDE mode).

Table 20-5. CompactFlash Mode Selection

A[23:21]	Mode Base Address
000	Attribute Memory
010	Common Memory
100	I/O Mode
110	True IDE Mode
111	Alternate True IDE Mode

20.6.5.2 *CFCE1 and CFCE2 signals*

To cover all types of access, the SMC must be alternatively set to drive 8-bit data bus or 16-bit data bus. The odd byte access on the D[7:0] bus is only possible when the SMC is configured to drive 8-bit memory devices on the corresponding NCS pin (NCS4 and or NCS5). The Chip Select Register (DBW field in the corresponding Chip Select Mode Register) of the NCS4 and/or NCS5 address space must be set as shown in Table 20-6 to enable the required access type.

NBS1 and NBS0 are the byte selection signals from SMC and are available when the SMC is set in Byte Select mode on the corresponding Chip Select.

The CFCE1 and CFCE2 waveforms are identical to the corresponding NCSx waveform. For details on these waveforms and timings, refer to the Static Memory Controller section.

Table 20-6. CFCE1 and CFCE2 Truth Table

Mode	CFCE2	CFCE1	DBW	Comment	SMC Access Mode
Attribute Memory	NBS1	NBS0	16 bits	Access to Even Byte on D[7:0]	Byte Select
Common Memory	NBS1	NBS0	16bits	Access to Even Byte on D[7:0] Access to Odd Byte on D[15:8]	Byte Select
	1	0	8 bits	Access to Odd Byte on D[7:0]	
I/O Mode	NBS1	NBS0	16 bits	Access to Even Byte on D[7:0] Access to Odd Byte on D[15:8]	Byte Select
	1	0	8 bits	Access to Odd Byte on D[7:0]	
True IDE Mode					
Task File	1	0	8 bits	Access to Even Byte on D[7:0] Access to Odd Byte on D[7:0]	
Data Register	1	0	16 bits	Access to Even Byte on D[7:0] Access to Odd Byte on D[15:8]	Byte Select
Alternate True IDE Mode					
Control Register Alternate Status Read	0	1	Don't Care	Access to Even Byte on D[7:0]	Don't Care
Drive Address	0	1	8 bits	Access to Odd Byte on D[7:0]	
Standby Mode or Address Space is not assigned to CF	1	1	–	–	–

20.6.5.3 Read/Write Signals

In I/O mode and True IDE mode, the CompactFlash logic drives the read and write command signals of the SMC on CFIOR and CFIOW signals, while the CFOE and CFWE signals are deactivated. Likewise, in common memory mode and attribute memory mode, the SMC signals are driven on the CFOE and CFWE signals, while the CFIOR and CFIOW are deactivated. Figure 20-4 on page 125 demonstrates a schematic representation of this logic.

Attribute memory mode, common memory mode and I/O mode are supported by setting the address setup and hold time on the NCS4 (and/or NCS5) chip select to the appropriate values.

Figure 20-4. CompactFlash Read/Write Control Signals

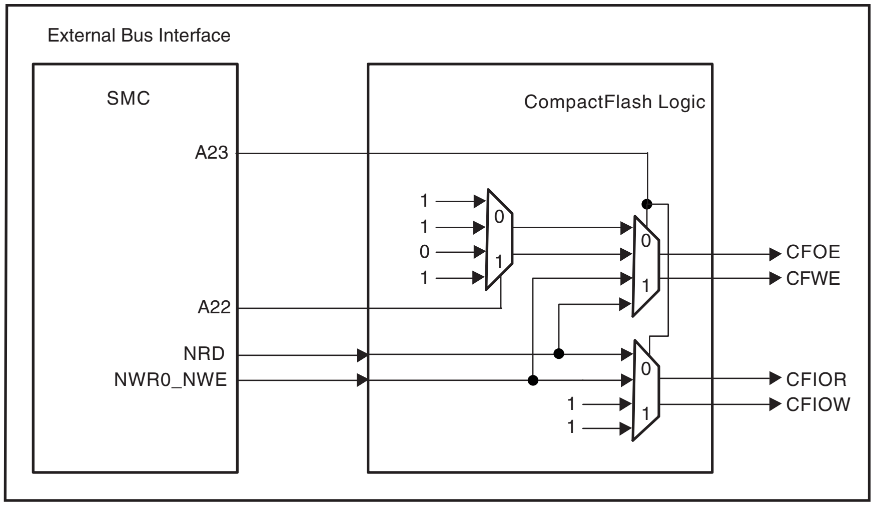


Table 20-7. CompactFlash Mode Selection

Mode Base Address	CFOE	CFWE	CFIOR	CFIOW
Attribute Memory Common Memory	NRD	NWR0_NWE	1	1
I/O Mode	1	1	NRD	NWR0_NWE
True IDE Mode	0	1	NRD	NWR0_NWE

20.6.5.4 Multiplexing of CompactFlash Signals on EBI Pins

Table 20-8 on page 125 and Table 20-9 on page 126 illustrate the multiplexing of the CompactFlash logic signals with other EBI signals on the EBI pins. The EBI pins in Table 20-8 are strictly dedicated to the CompactFlash interface as soon as the CS4A and/or CS5A field of the CSA Register is set. These pins must not be used to drive any other memory devices.

The EBI pins in Table 20-9 on page 126 remain shared between all memory areas when the corresponding CompactFlash interface is enabled (CS4A = 1 and/or CS5A = 1).

Table 20-8. Dedicated CompactFlash Interface Multiplexing

Pins	CompactFlash Signals		EBI Signals	
	CS4A = 1	**CS5A = 1**	**CS4A = 0**	**CS5A = 0**
NCS4/CFCS0	CFCS0		NCS4	
NCS5/CFCS1		CFCS1		NCS5

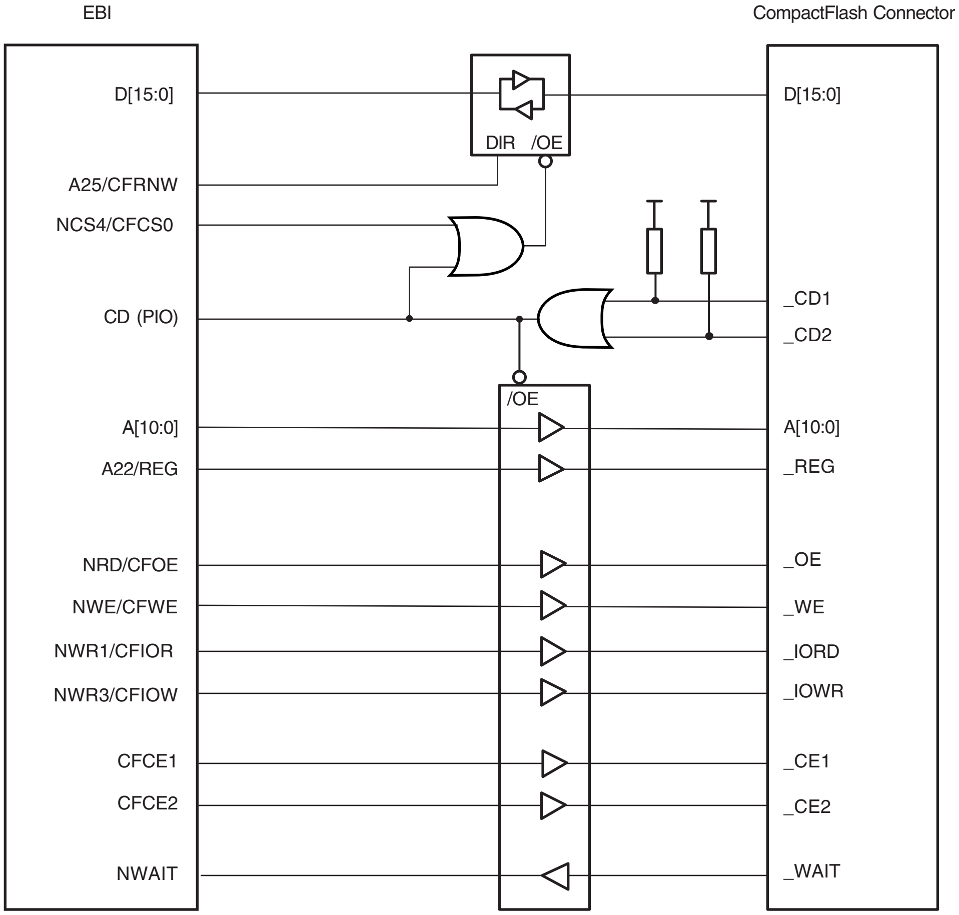
Table 20-9. Shared CompactFlash Interface Multiplexing

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

20.6.5.5 Application Example

Figure 20-5 on page 127 illustrates an example of a CompactFlash application. CFCS0 and CFRNW signals are not directly connected to the CompactFlash slot 0, but do control the direction and the output enable of the buffers between the EBI and the CompactFlash Device. The timing of the CFCS0 signal is identical to the NCS4 signal. Moreover, the CFRNW signal remains valid throughout the transfer, as does the address bus. The CompactFlash _WAIT signal is connected to the NWAIT input of the Static Memory Controller. For details on these waveforms and timings, refer to the Static Memory Controller section.

Figure 20-5. CompactFlash Application Example


EBI
CompactFlash Connector
D[15:0]
DIR
/OE
A25/CFRNW
NCS4/CFCS0
CD (PIO)
A[10:0]
A22/REG
NRD/CFOE
NWE/CFWE
NWR1/CFIOR
NWR3/CFIOW
CFCE1
CFCE2
NWAIT
/OE
D[15:0]
_CD1
_CD2
A[10:0]
_REG
_OE
_WE
_IORD
_IOWR
_CE1
_CE2
_WAIT

20.6.6 NAND Flash Support

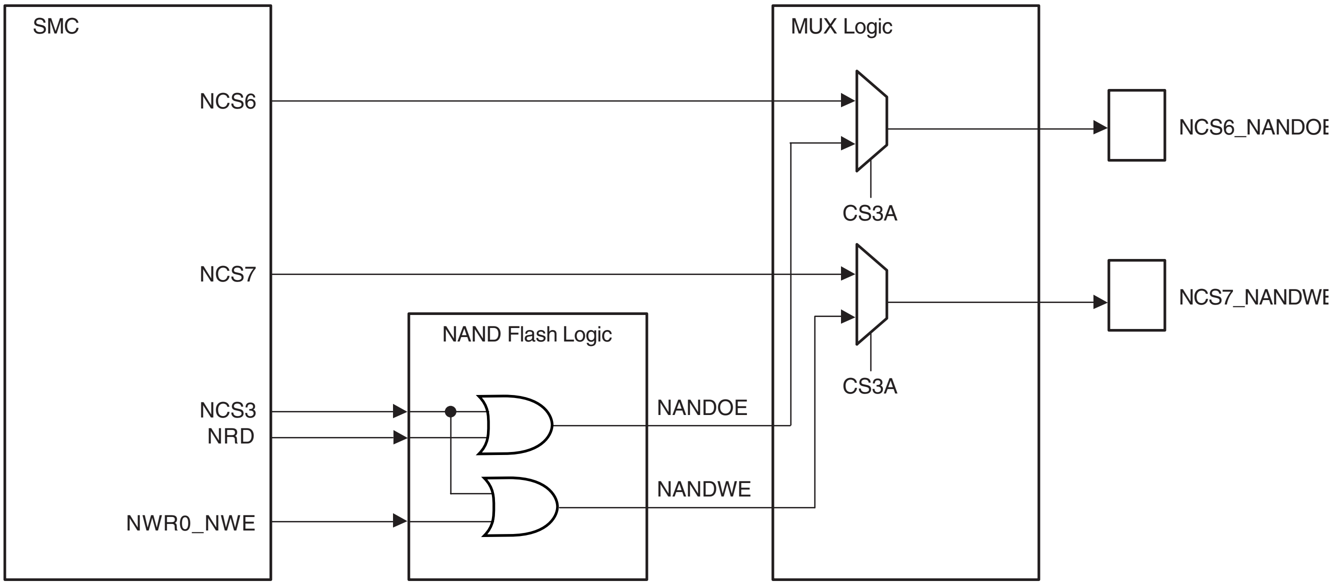
The EBI integrates circuitry that interfaces to NAND Flash devices.

The NAND Flash logic is driven by the Static Memory Controller on the NCS3 address space. Programming the CS3A field in the CSA Register in the Bus Matrix User Interface to the appropriate value enables the NAND Flash logic. For details on this register, refer to the Bus Matrix User Interface section. Access to an external NAND Flash device is then made by accessing the address space reserved to NCS3 (i.e., between 0x40000000 and 0x4FFF FFFF).

The NAND Flash Logic drives the read and write command signals of the SMC on the NANDOE and NANDWE signals when the NCS3 signal is active. NANDOE and NANDWE are invalidated as soon as the transfer address fails to lie in the NCS3 address space. For details on these waveforms, refer to the Static Memory Controller section.

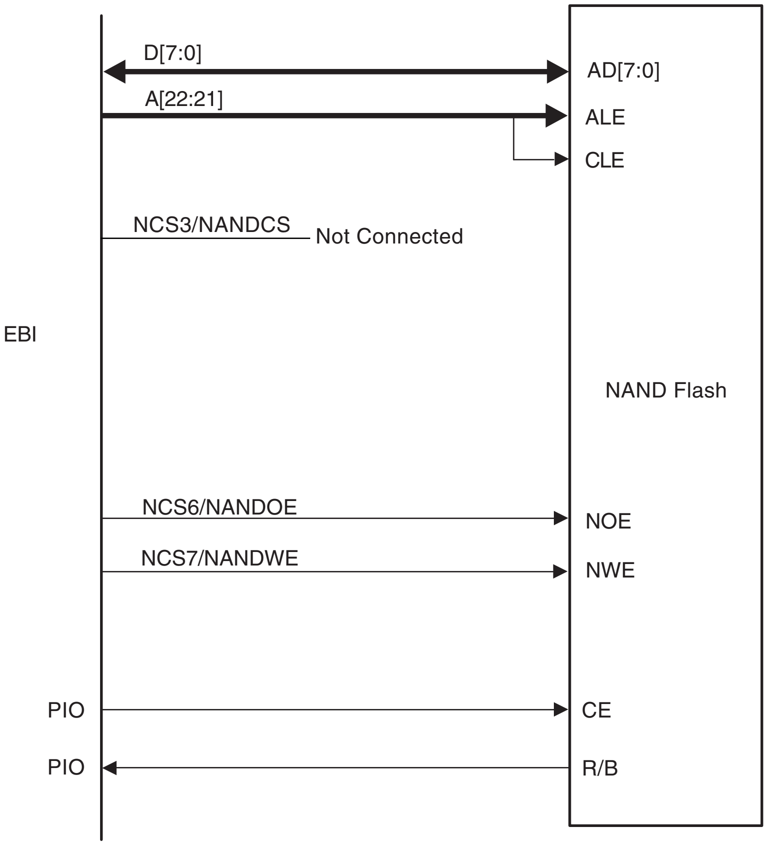
The NANDOE and NANDWE signals are multiplexed with NCS6 and NCS7 signals of the Static Memory Controller. This multiplexing is controlled in the MUX logic part of the EBI by the CS3A bit in the in the CSA Register For details on this register, refer to the Bus Matrix User Interface Section. NCS6 and NCS7 become unavailable. Performing an access within the address space reserved to NCS6 and NCS7 (i.e., between 0x70000000 and 0x8FFF FFFF) may lead to an unpredictable outcome.

Figure 20-6. NAND Flash Signal Multiplexing on EBI Pins


SMC
NCS6
NCS7
NCS3
NRD
NWR0_NWE
NAND Flash Logic
NANDOE
NANDWE
MUX Logic
CS3A
CS3A
NCS6_NANDOE
NCS7_NANDWE

The address latch enable and command latch enable signals on the NAND Flash device are driven by address bits A22 and A21 of the EBI address bus. The user should note that any bit on the EBI address bus can also be used for this purpose. The command, address or data words on the data bus of the NAND Flash device are distinguished by using their address within the NCS3 address space. The chip enable (CE) signal of the device and the ready/busy (R/B) signals are connected to PIO lines. The CE signal then remains asserted even when NCS3 is not selected, preventing the device from returning to standby mode.

Figure 20-7. NAND Flash Application Example


D[7:0]
A[22:21]
AD[7:0]
ALE
CLE
NCS3/NANDCS
Not Connected
EBI
NAND Flash
NCS6/NANDOE
NOE
NCS7/NANDWE
NWE
PIO
CE
PIO
R/B

Note: The External Bus Interface is also able to support 16-bits devices.

21. Static Memory Controller (SMC)

21.1 Description

The Static Memory Controller (SMC) generates the signals that control the access to the external memory devices or peripheral devices. It has 8 Chip Selects and a 26-bit address bus. The 32-bit data bus can be configured to interface with 8-, 16-, or 32-bit external devices. Separate read and write control signals allow for direct memory and peripheral interfacing. Read and write signal waveforms are fully parametrizable.

The SMC can manage wait requests from external devices to extend the current access. The SMC is provided with an automatic slow clock mode. In slow clock mode, it switches from user-programmed waveforms to slow-rate specific waveforms on read and write signals. The SMC supports asynchronous burst read in page mode access for page size up to 32 bytes.

21.2 I/O Lines Description

Table 21-1. I/O Line Description

Name	Description	Type	Active Level
NCS[7:0]	Static Memory Controller Chip Select Lines	Output	Low
NRD	Read Signal	Output	Low
NWR0/NWE	Write 0/Write Enable Signal	Output	Low
A0/NBS0	Address Bit 0/Byte 0 Select Signal	Output	Low
NWR1/NBS1	Write 1/Byte 1 Select Signal	Output	Low
A1/NWR2/NBS2	Address Bit 1/Write 2/Byte 2 Select Signal	Output	Low
NWR3/NBS3	Write 3/Byte 3 Select Signal	Output	Low
A[25:2]	Address Bus	Output	
D[31:0]	Data Bus	I/O	
NWAIT	External Wait Signal	Input	Low

21.3 Multiplexed Signals

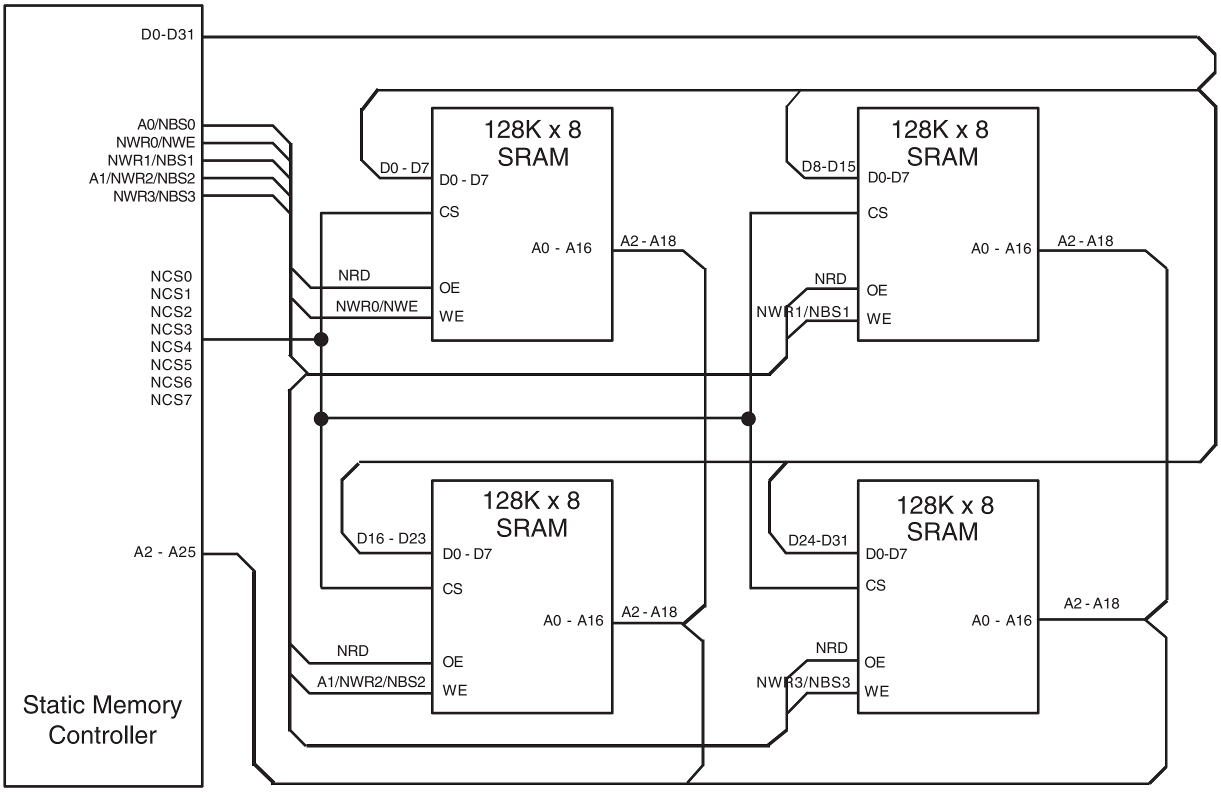
Table 21-2. Static Memory Controller (SMC) Multiplexed Signals

Multiplexed Signals			Related Function
NWR0	NWE		Byte-write or byte-select access, see "Byte Write or Byte Select Access" on page 133
A0	NBS0		8-bit or 16-/32-bit data bus, see "Data Bus Width" on page 133
NWR1	NBS1		Byte-write or byte-select access see "Byte Write or Byte Select Access" on page 133
A1	NWR2	NBS2	8-/16-bit or 32-bit data bus, see "Data Bus Width" on page 133. Byte-write or byte-select access, see "Byte Write or Byte Select Access" on page 133
NWR3	NBS3		Byte-write or byte-select access see "Byte Write or Byte Select Access" on page 133

21.4 Application Example

21.4.1 Hardware Interface

Figure 21-1. SMC Connections to Static Memory Devices


D0-D31
A0/NBS0
NWR0/NWE
NWR1/NBS1
A1/NWR2/NBS2
NWR3/NBS3
NCS0
NCS1
NCS2
NCS3
NCS4
NCS5
NCS6
NCS7
A2 - A25
Static Memory Controller
128K x 8 SRAM
D0 - D7
D0 - D7
CS
A0 - A16
A2 - A18
NRD
OE
NWR0/NWE
WE
128K x 8 SRAM
D8-D15
D0-D7
CS
A0 - A16
A2 - A18
NRD
OE
NWR1/NBS1
WE
128K x 8 SRAM
D16 - D23
D0 - D7
CS
A0 - A16
A2 - A18
NRD
OE
A1/NWR2/NBS2
WE
128K x 8 SRAM
D24-D31
D0-D7
CS
A0 - A16
A2 - A18
NRD
OE
NWR3/NBS3
WE

21.5 Product Dependencies

21.5.1 I/O Lines

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

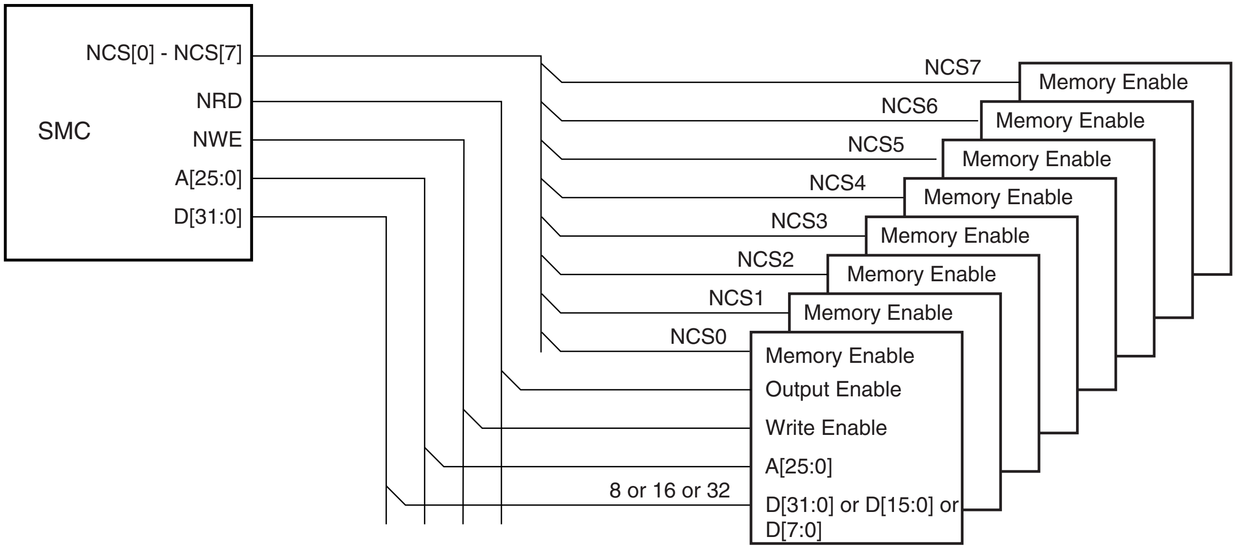
21.6 External Memory Mapping

The SMC provides up to 26 address lines, A[25:0]. This allows each chip select line to address up to 64 Mbytes of memory.

If the physical memory device connected on one chip select is smaller than 64 Mbytes, it wraps around and appears to be repeated within this space. The SMC correctly handles any valid access to the memory device within the page (see Figure 21-2).

A[25:0] is only significant for 8-bit memory, A[25:1] is used for 16-bit memory, A[25:2] is used for 32-bit memory.

Figure 21-2. Memory Connections for Eight External Devices


SMC
NCS[0] - NCS[7]
NRD
NWE
A[25:0]
D[31:0]
NCS7
NCS6
NCS5
NCS4
NCS3
NCS2
NCS1
NCS0
Memory Enable
Memory Enable
Memory Enable
Memory Enable
Memory Enable
Memory Enable
Memory Enable
Memory Enable
Output Enable
Write Enable
A[25:0]
8 or 16 or 32
D[31:0] or D[15:0] or D[7:0]

21.7 Connection to External Devices

21.7.1 Data Bus Width

A data bus width of 8, 16, or 32 bits can be selected for each chip select. This option is controlled by the field DBW in SMC_MODE (Mode Register) for the corresponding chip select.

Figure 21-3 shows how to connect a 512K x 8-bit memory on NCS2. Figure 21-4 shows how to connect a 512K x 16-bit memory on NCS2. Figure 21-5 shows two 16-bit memories connected as a single 32-bit memory

21.7.2 Byte Write or Byte Select Access

Each chip select with a 16-bit or 32-bit data bus can operate with one of two different types of write access: byte write or byte select access. This is controlled by the BAT field of the SMC_MODE register for the corresponding chip select.

Figure 21-3. Memory Connection for an 8-bit Data Bus

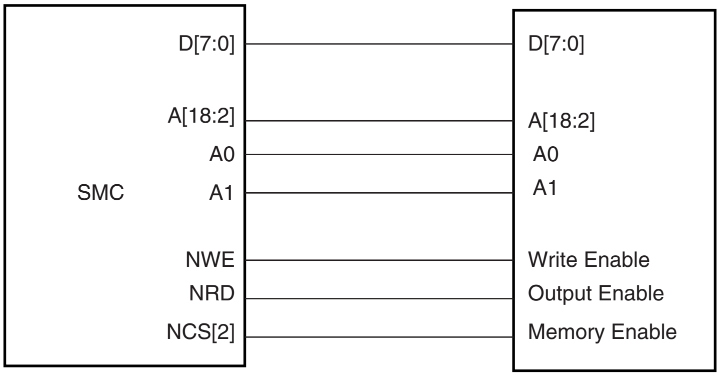

SMC
D[7:0]
A[18:2]
A0
A1
NWE
NRD
NCS[2]
D[7:0]
A[18:2]
A0
A1
Write Enable
Output Enable
Memory Enable

Figure 21-4. Memory Connection for a 16-bit Data Bus

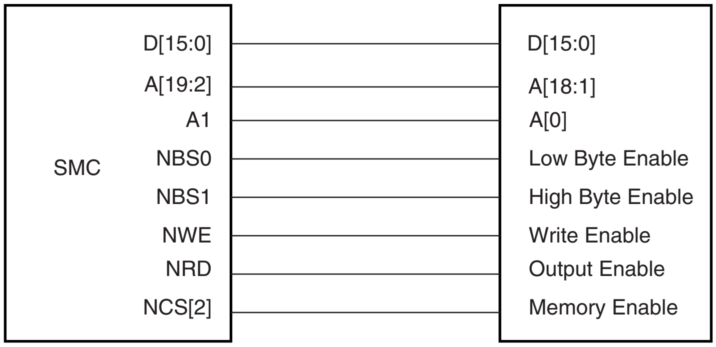
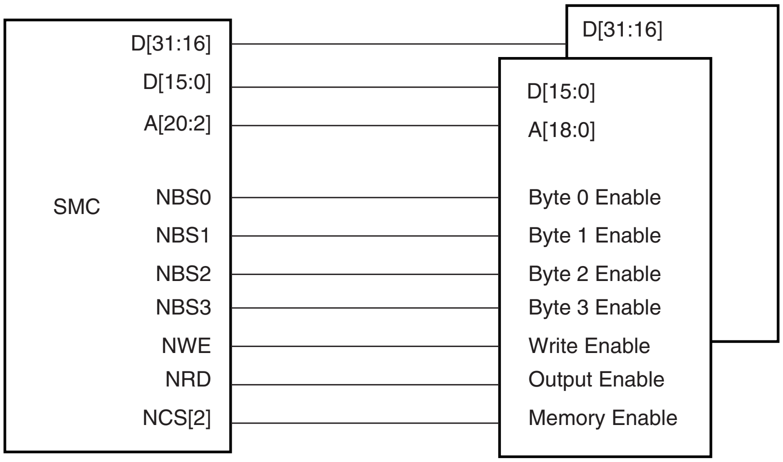

SMC
D[15:0]
A[19:2]
A1
NBS0
NBS1
NWE
NRD
NCS[2]
D[15:0]
A[18:1]
A[0]
Low Byte Enable
High Byte Enable
Write Enable
Output Enable
Memory Enable

Figure 21-5. Memory Connection for a 32-bit Data Bus


SMC
D[31:16]
D[15:0]
A[20:2]
NBS0
NBS1
NBS2
NBS3
NWE
NRD
NCS[2]
D[31:16]
D[15:0]
A[18:0]
Byte 0 Enable
Byte 1 Enable
Byte 2 Enable
Byte 3 Enable
Write Enable
Output Enable
Memory Enable

21.7.2.1 Byte Write Access

Byte write access supports one byte write signal per byte of the data bus and a single read signal.

Note that the SMC does not allow boot in Byte Write Access mode.

- For 16-bit devices: the SMC provides NWR0 and NWR1 write signals for respectively byte0 (lower byte) and byte1 (upper byte) of a 16-bit bus. One single read signal (NRD) is provided.

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

- For 32-bit devices: NWR0, NWR1, NWR2 and NWR3, are the write signals of byte0 (lower byte), byte1, byte2 and byte 3 (upper byte) respectively. One single read signal (NRD) is provided.

  Byte Write Access is used to connect 4 x 8-bit devices as a 32-bit memory.

  Byte Write option is illustrated on Figure 21-6.

21.7.2.2 Byte Select Access

In this mode, read/write operations can be enabled/disabled at a byte level. One byte-select line per byte of the data bus is provided. One NRD and one NWE signal control read and write.

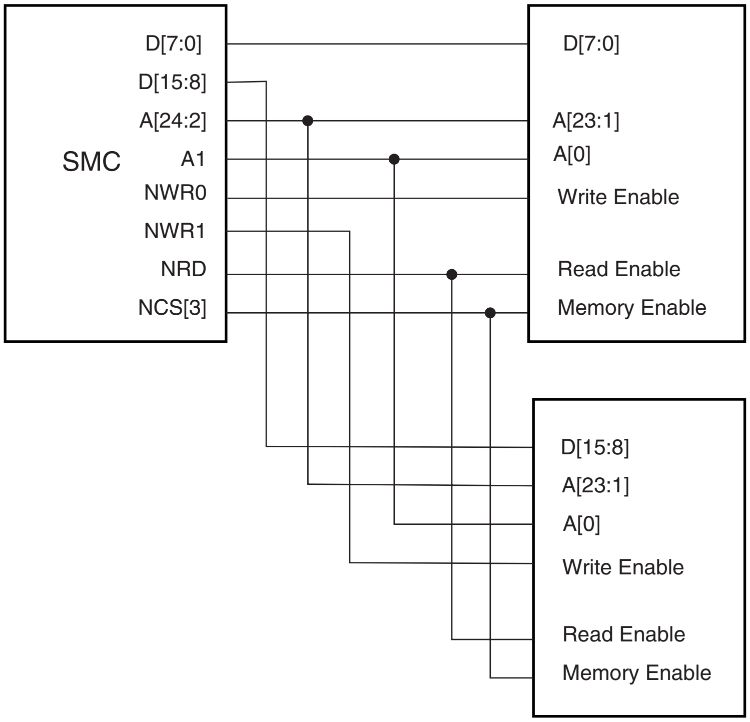
- For 16-bit devices: the SMC provides NBS0 and NBS1 selection signals for respectively byte0 (lower byte) and byte1 (upper byte) of a 16-bit bus.

  Byte Select Access is used to connect one 16-bit device.

- For 32-bit devices: NBS0, NBS1, NBS2 and NBS3, are the selection signals of byte0 (lower byte), byte1, byte2 and byte 3 (upper byte) respectively. Byte Select Access is used to connect two 16-bit devices.

Figure 21-7 shows how to connect two 16-bit devices on a 32-bit data bus in Byte Select Access mode, on NCS3 (BAT = Byte Select Access).

Figure 21-6. Connection of 2 x 8-bit Devices on a 16-bit Bus: Byte Write Option


SMC
D[7:0]
D[15:8]
A[24:2]
A1
NWR0
NWR1
NRD
NCS[3]
D[7:0]
A[23:1]
A[0]
Write Enable
Read Enable
Memory Enable
D[15:8]
A[23:1]
A[0]
Write Enable
Read Enable
Memory Enable

21.7.2.3 Signal Multiplexing

Depending on the BAT, only the write signals or the byte select signals are used. To save IOs at the external bus interface, control signals at the SMC interface are multiplexed. Table 21-3 shows signal multiplexing depending on the data bus width and the byte access type.

For 32-bit devices, bits A0 and A1 are unused. For 16-bit devices, bit A0 of address is unused. When Byte Select Option is selected, NWR1 to NWR3 are unused. When Byte Write option is selected, NBS0 to NBS3 are unused.

Figure 21-7. Connection of 2x16-bit Data Bus on a 32-bit Data Bus (Byte Select Option)

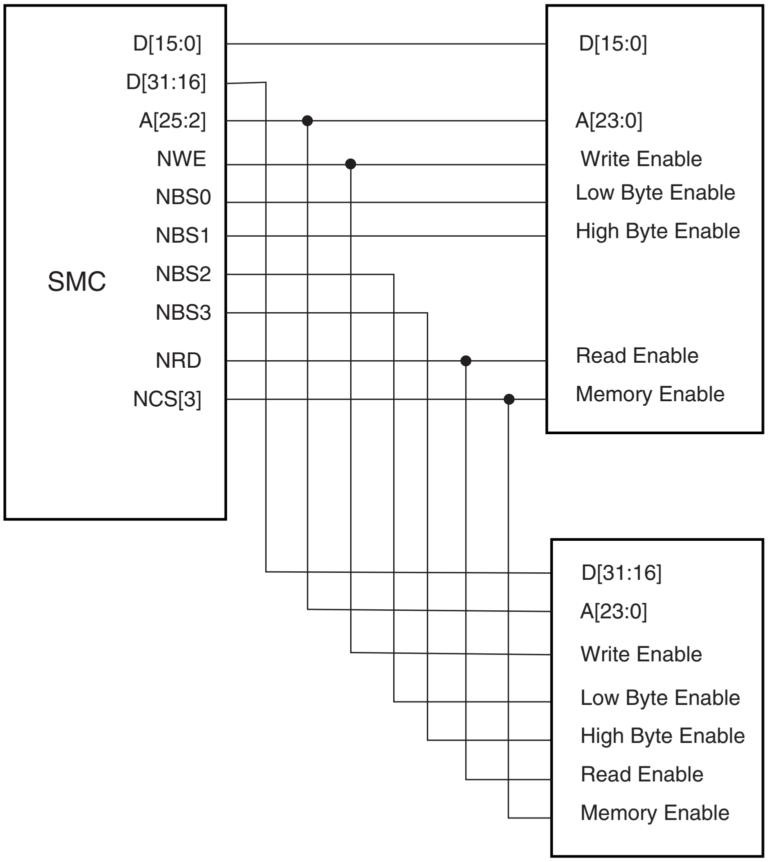

SMC
D[15:0]
D[31:16]
A[25:2]
NWE
NBS0
NBS1
NBS2
NBS3
NRD
NCS[3]
D[15:0]
A[23:0]
Write Enable
Low Byte Enable
High Byte Enable
Read Enable
Memory Enable
D[31:16]
A[23:0]
Write Enable
Low Byte Enable
High Byte Enable
Read Enable
Memory Enable

Table 21-3. SMC Multiplexed Signal Translation

Signal Name	**32-bit Bus**			**16-bit Bus**		**8-bit Bus**
Device Type	**1x32-bit**	**2x16-bit**	**4 x 8-bit**	**1x16-bit**	**2 x 8-bit**	**1 x 8-bit**
Byte Access Type (BAT)	Byte Select	Byte Select	Byte Write	Byte Select	Byte Write	
NBS0_A0	NBS0	NBS0		NBS0		A0
NWE_NWR0	NWE	NWE	NWR0	NWE	NWR0	NWE
NBS1_NWR1	NBS1	NBS1	NWR1	NBS1	NWR1	
NBS2_NWR2_A1	NBS2	NBS2	NWR2	A1	A1	A1
NBS3_NWR3	NBS3	NBS3	NWR3			

21.8 Standard Read and Write Protocols

In the following sections, the byte access type is not considered. Byte select lines (NBS0 to NBS3) always have the same timing as the A address bus. NWE represents either the NWE signal in byte select access type or one of the byte write lines (NWR0 to NWR3) in byte write access type. NWR0 to NWR3 have the same timings and protocol as NWE. In the same way, NCS represents one of the NCS[0..7] chip select lines.

21.8.1 Read Waveforms

The read cycle is shown on Figure 21-8.

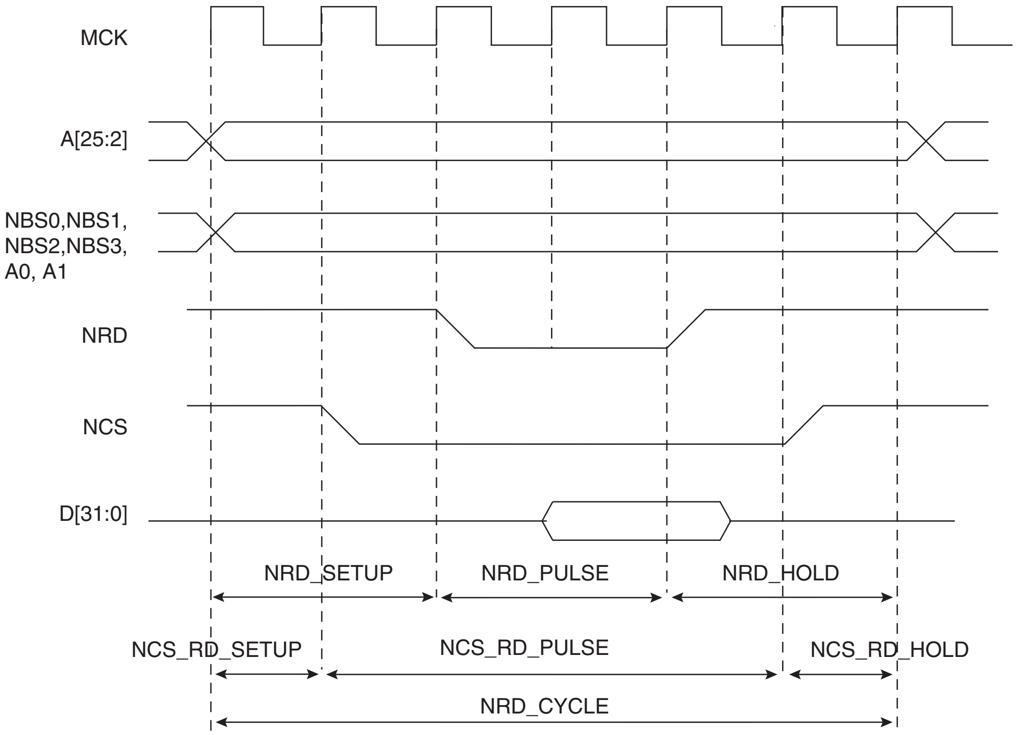
The read cycle starts with the address setting on the memory address bus, i.e.:

{A[25:2], A1, A0} for 8-bit devices

{A[25:2], A1} for 16-bit devices

A[25:2] for 32-bit devices.

Figure 21-8. Standard Read Cycle


MCK
A[25:2]
NBS0,NBS1,
NBS2,NBS3,
A0, A1
NRD
NCS
D[31:0]
NRD_SETUP
NRD_PULSE
NRD_HOLD
NCS_RD_SETUP
NCS_RD_PULSE
NCS_RD_HOLD
NRD_CYCLE

21.8.1.1 NRD Waveform

The NRD signal is characterized by a setup timing, a pulse width and a hold timing.

1. NRD_SETUP: the NRD setup time is defined as the setup of address before the NRD falling edge;
2. NRD_PULSE: the NRD pulse length is the time between NRD falling edge and NRD rising edge;
3. NRD_HOLD: the NRD hold time is defined as the hold time of address after the NRD rising edge.

21.8.1.2 NCS Waveform

Similarly, the NCS signal can be divided into a setup time, pulse length and hold time:

1. NCS_RD_SETUP: the NCS setup time is defined as the setup time of address before the NCS falling edge.
2. NCS_RD_PULSE: the NCS pulse length is the time between NCS falling edge and NCS rising edge;
3. NCS_RD_HOLD: the NCS hold time is defined as the hold time of address after the NCS rising edge.

21.8.1.3 Read Cycle

The NRD_CYCLE time is defined as the total duration of the read cycle, i.e., from the time where address is set on the address bus to the point where address may change. The total read cycle time is equal to:

NRD_CYCLE = NRD_SETUP + NRD_PULSE + NRD_HOLD

= NCS_RD_SETUP + NCS_RD_PULSE + NCS_RD_HOLD

All NRD and NCS timings are defined separately for each chip select as an integer number of Master Clock cycles. To ensure that the NRD and NCS timings are coherent, user must define the total read cycle instead of the hold timing. NRD_CYCLE implicitly defines the NRD hold time and NCS hold time as:

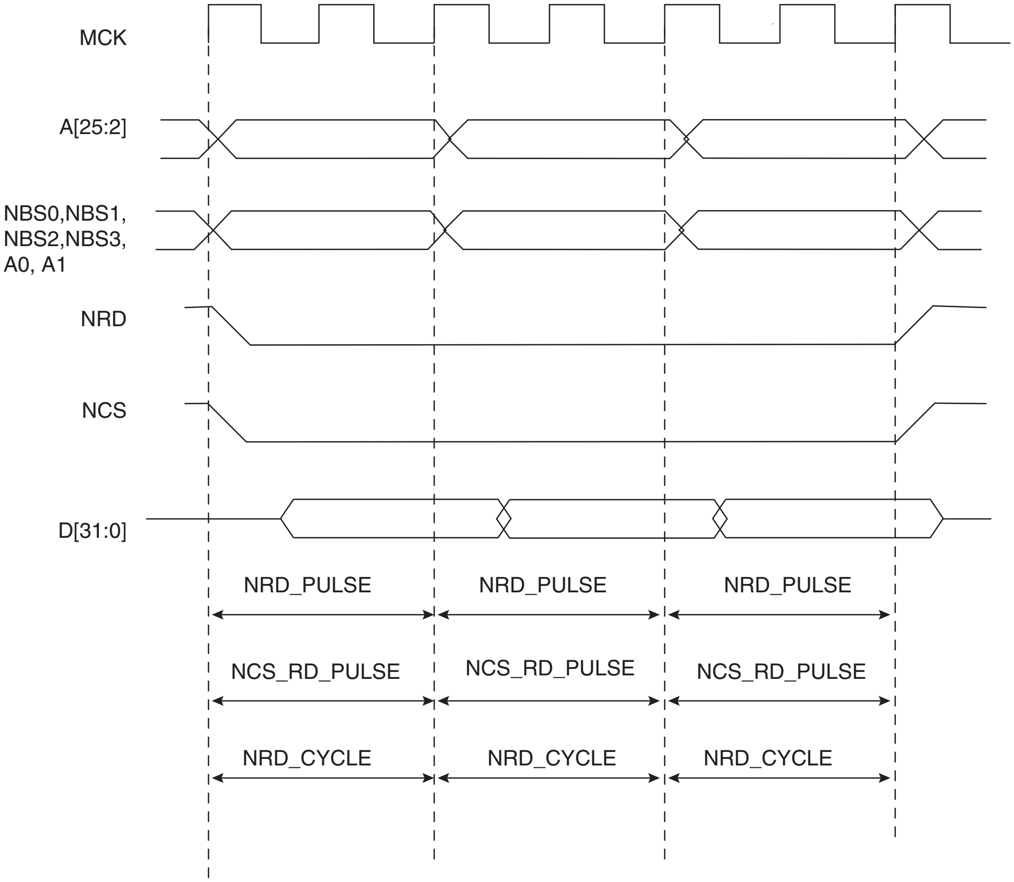
NRD_HOLD = NRD_CYCLE - NRD SETUP - NRD PULSE

NCS_RD_HOLD = NRD_CYCLE - NCS_RD_SETUP - NCS_RD_PULSE

21.8.1.4 Null Delay Setup and Hold

If null setup and hold parameters are programmed for NRD and/or NCS, NRD and NCS remain active continuously in case of consecutive read cycles in the same memory (see Figure 21-9).

Figure 21-9. No Setup, No Hold On NRD and NCS Read Signals


MCK
A[25:2]
NBS0,NBS1,
NBS2,NBS3,
A0, A1
NRD
NCS
D[31:0]
NRD_PULSE
NRD_PULSE
NRD_PULSE
NCS_RD_PULSE
NCS_RD_PULSE
NCS_RD_PULSE
NRD_CYCLE
NRD_CYCLE
NRD_CYCLE

21.8.1.5 Null Pulse

Programming null pulse is not permitted. Pulse must be at least set to 1. A null value leads to unpredictable behavior.

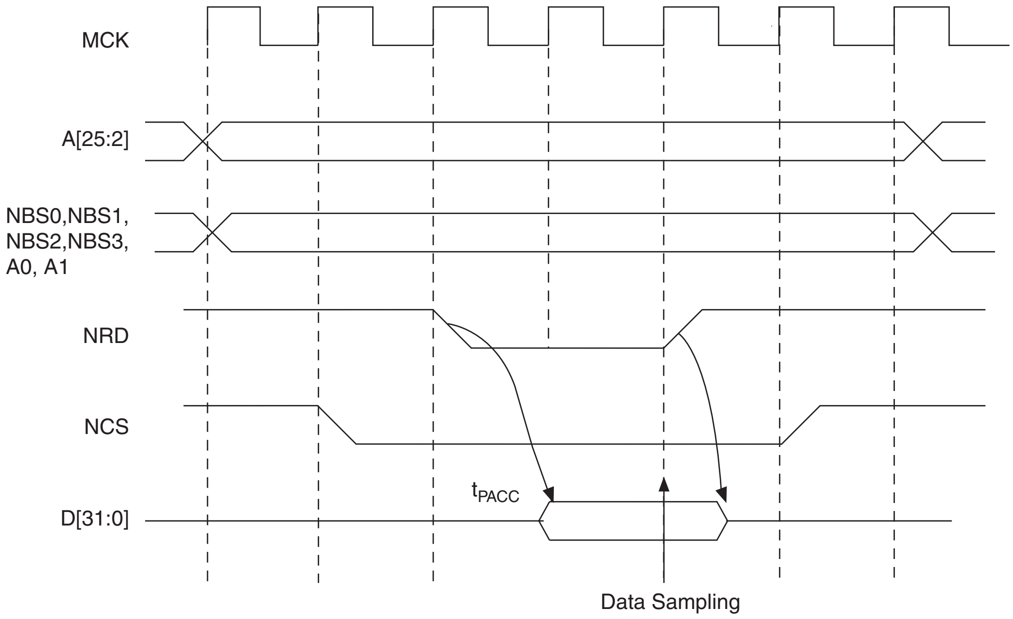
21.8.2 Read Mode

As NCS and NRD waveforms are defined independently of one other, the SMC needs to know when the read data is available on the data bus. The SMC does not compare NCS and NRD timings to know which signal rises first. The READ_MODE parameter in the SMC_MODE register of the corresponding chip select indicates which signal of NRD and NCS controls the read operation.

21.8.2.1 Read is Controlled by NRD (READ_MODE = 1):

Figure 21-10 shows the waveforms of a read operation of a typical asynchronous RAM. The read data is available t_{PACC} after the falling edge of NRD, and turns to 'Z' after the rising edge of NRD. In this case, the READ_MODE must be set to 1 (read is controlled by NRD), to indicate that data is available with the rising edge of NRD. The SMC samples the read data internally on the rising edge of Master Clock that generates the rising edge of NRD, whatever the programmed waveform of NCS may be.

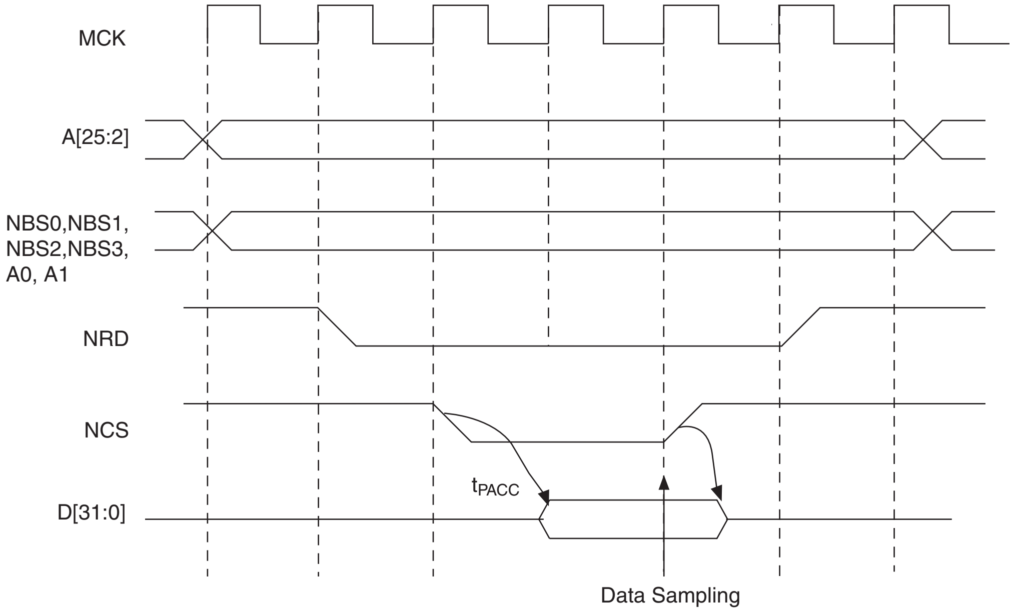
Figure 21-10. READ_MODE = 1: Data is sampled by SMC before the rising edge of NRD


MCK
A[25:2]
NBS0,NBS1,
NBS2,NBS3,
A0, A1
NRD
NCS
t_{PACC}
D[31:0]
Data Sampling

21.8.2.2 Read is Controlled by NCS (READ_MODE = 0)

Figure 21-11 shows the typical read cycle of an LCD module. The read data is valid t_{PACC} after the falling edge of the NCS signal and remains valid until the rising edge of NCS. Data must be sampled when NCS is raised. In that case, the READ_MODE must be set to 0 (read is controlled by NCS): the SMC internally samples the data on the rising edge of Master Clock that generates the rising edge of NCS, whatever the programmed waveform of NRD may be.

Figure 21-11. READ_MODE = 0: Data is sampled by SMC before the rising edge of NCS


MCK
A[25:2]
NBS0,NBS1,
NBS2,NBS3,
A0, A1
NRD
NCS
t_{PACC}
D[31:0]
Data Sampling

21.8.3 Write Waveforms

The write protocol is similar to the read protocol. It is depicted in Figure 21-12. The write cycle starts with the address setting on the memory address bus.

21.8.3.1 NWE Waveforms

The NWE signal is characterized by a setup timing, a pulse width and a hold timing.

1. NWE_SETUP: the NWE setup time is defined as the setup of address and data before the NWE falling edge;
2. NWE_PULSE: The NWE pulse length is the time between NWE falling edge and NWE rising edge;
3. NWE_HOLD: The NWE hold time is defined as the hold time of address and data after the NWE rising edge.

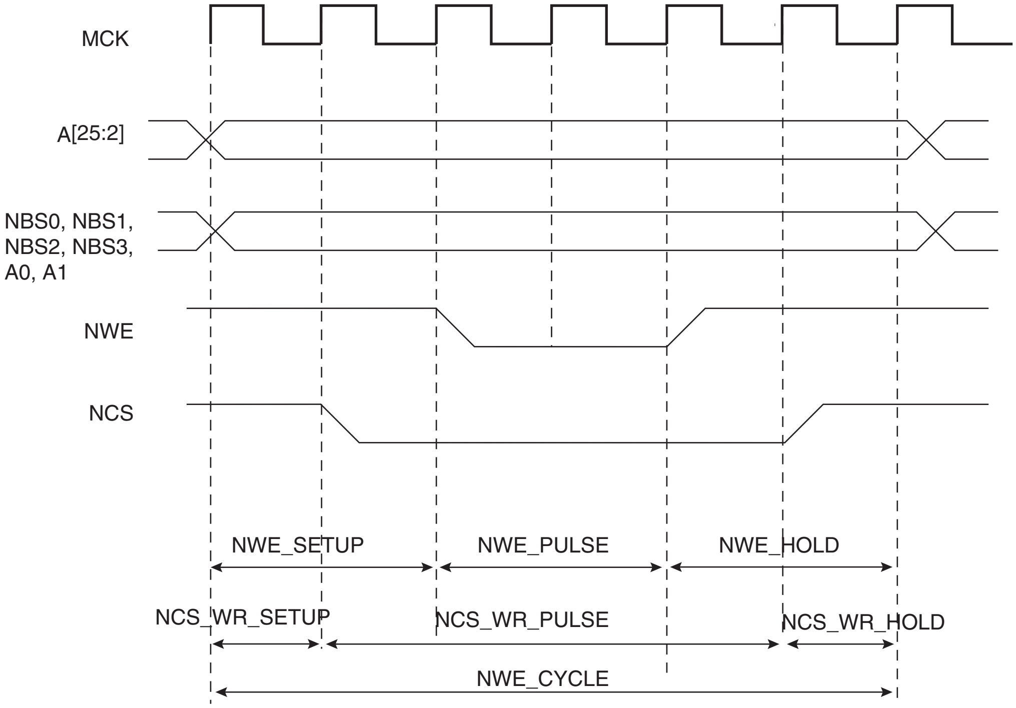
The NWE waveforms apply to all byte-write lines in Byte Write access mode: NWR0 to NWR3.

21.8.3.2 NCS Waveforms

The NCS signal waveforms in write operation are not the same that those applied in read operations, but are separately defined:

1. NCS_WR_SETUP: the NCS setup time is defined as the setup time of address before the NCS falling edge.
2. NCS_WR_PULSE: the NCS pulse length is the time between NCS falling edge and NCS rising edge;
3. NCS_WR_HOLD: the NCS hold time is defined as the hold time of address after the NCS rising edge.

Figure 21-12. Write Cycle



21.8.3.3 Write Cycle

The write_cycle time is defined as the total duration of the write cycle, that is, from the time where address is set on the address bus to the point where address may change. The total write cycle time is equal to:

NWE_CYCLE = NWE_SETUP + NWE_PULSE + NWE_HOLD

= NCS_WR_SETUP + NCS_WR_PULSE + NCS_WR_HOLD

All NWE and NCS (write) timings are defined separately for each chip select as an integer number of Master Clock cycles. To ensure that the NWE and NCS timings are coherent, the user must define the total write cycle instead of the hold timing. This implicitly defines the NWE hold time and NCS (write) hold times as:

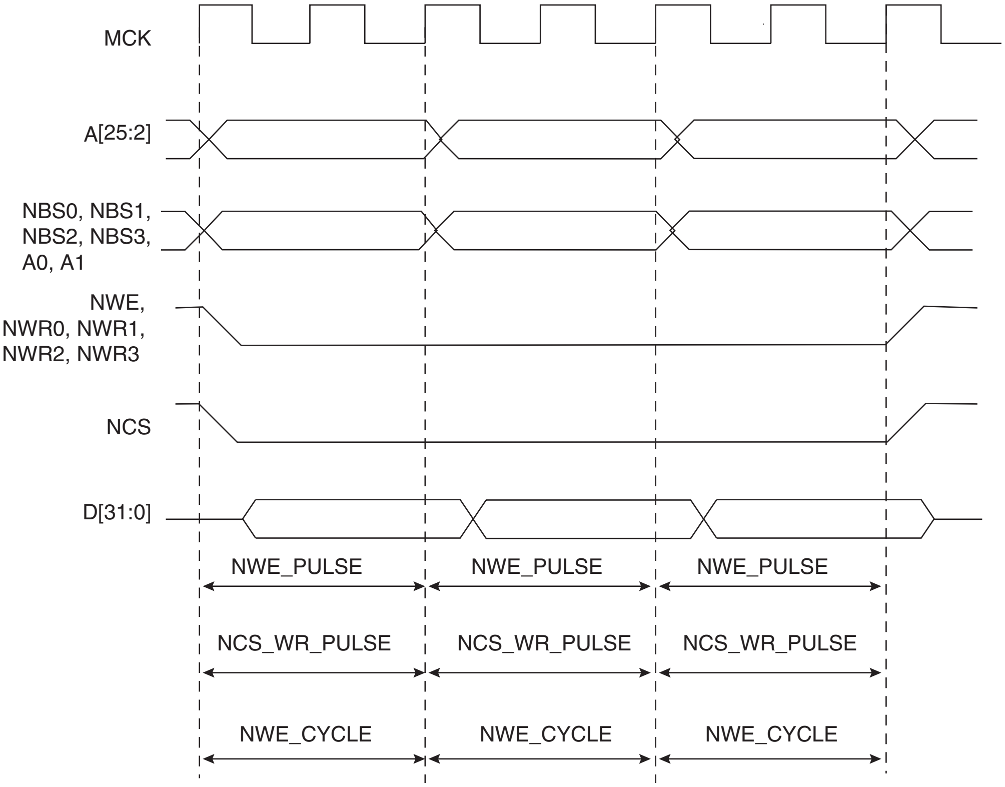
NWE_HOLD = NWE_CYCLE - NWE_SETUP - NWE_PULSE

NCS_WR_HOLD = NWE_CYCLE - NCS_WR_SETUP - NCS_WR_PULSE

21.8.3.4 Null Delay Setup and Hold

If null setup parameters are programmed for NWE and/or NCS, NWE and/or NCS remain active continuously in case of consecutive write cycles in the same memory (see Figure 21-13). However, for devices that perform write operations on the rising edge of NWE or NCS, such as SRAM, either a setup or a hold must be programmed.

Figure 21-13. Null Setup and Hold Values of NCS and NWE in Write Cycle


MCK
A[25:2]
NBS0, NBS1, NBS2, NBS3, A0, A1
NWE, NWR0, NWR1, NWR2, NWR3
NCS
D[31:0]
NWE_PULSE
NWE_PULSE
NWE_PULSE
NCS_WR_PULSE
NCS_WR_PULSE
NCS_WR_PULSE
NWE_CYCLE
NWE_CYCLE
NWE_CYCLE

21.8.3.5 Null Pulse

Programming null pulse is not permitted. Pulse must be at least set to 1. A null value leads to unpredictable behavior.

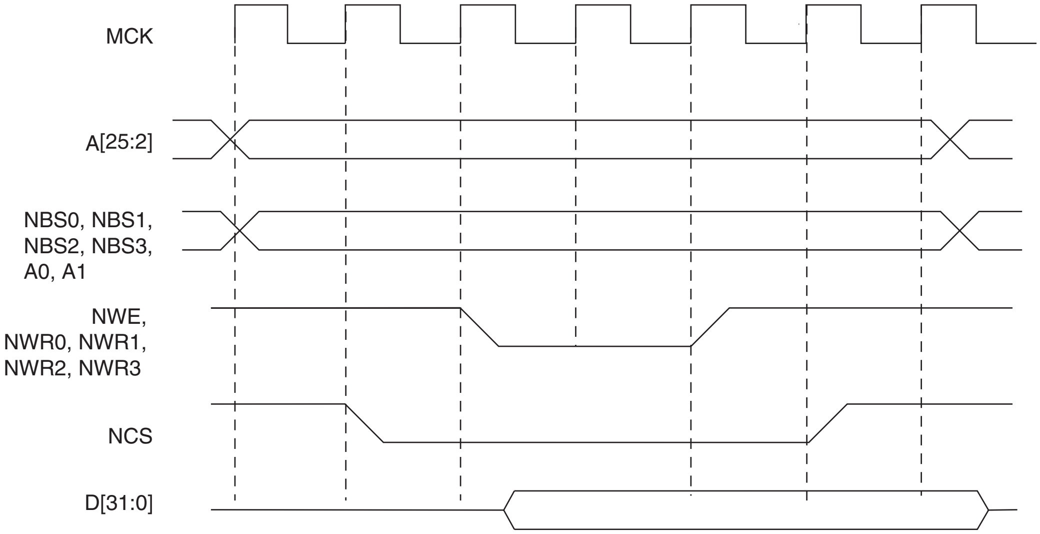
21.8.4 Write Mode

The WRITE_MODE parameter in the SMC_MODE register of the corresponding chip select indicates which signal controls the write operation.

21.8.4.1 Write is Controlled by NWE (WRITE_MODE = 1):

Figure 21-14 shows the waveforms of a write operation with WRITE_MODE set to 1. The data is put on the bus during the pulse and hold steps of the NWE signal. The internal data buffers are turned out after the NWE_SETUP time, and until the end of the write cycle, regardless of the programmed waveform on NCS.

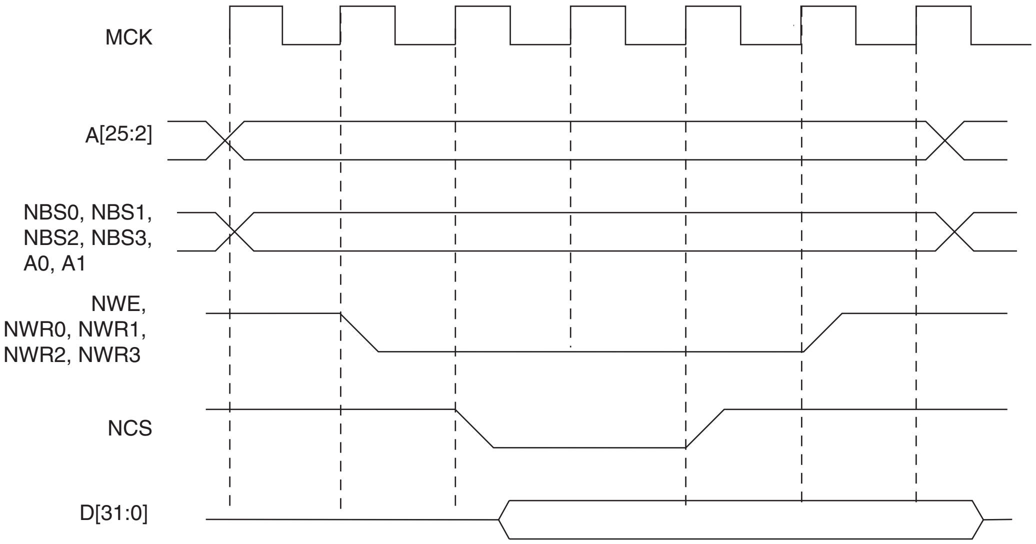
Figure 21-14. WRITE_MODE = 1. The write operation is controlled by NWE



21.8.4.2 Write is Controlled by NCS (WRITE_MODE = 0)

Figure 21-15 shows the waveforms of a write operation with WRITE_MODE set to 0. The data is put on the bus during the pulse and hold steps of the NCS signal. The internal data buffers are turned out after the NCS_WR_SETUP time, and until the end of the write cycle, regardless of the programmed waveform on NWE.

Figure 21-15. WRITE_MODE = 0. The write operation is controlled by NCS



21.8.5 Coding Timing Parameters

All timing parameters are defined for one chip select and are grouped together in one SMC_REGISTER according to their type.

The SMC_SETUP register groups the definition of all setup parameters:

- NRD_SETUP, NCS_RD_SETUP, NWE_SETUP, NCS_WR_SETUP

The SMC_PULSE register groups the definition of all pulse parameters:

- NRD_PULSE, NCS_RD_PULSE, NWE_PULSE, NCS_WR_PULSE

The SMC_CYCLE register groups the definition of all cycle parameters:

- NRD_CYCLE, NWE_CYCLE

Table 21-4 shows how the timing parameters are coded and their permitted range.

Table 21-4. Coding and Range of Timing Parameters

			Permitted Range	
Coded Value	**Number of Bits**	**Effective Value**	**Coded Value**	**Effective Value**
setup [5:0]	6	128 x setup[5] + setup[4:0]	0 ≤≤31	128 ≤≤128+31
pulse [6:0]	7	256 x pulse[6] + pulse[5:0]	0 ≤≤63	256 ≤≤256+63
cycle [8:0]	9	256 x cycle[8:7] + cycle[6:0]	0 ≤≤127	256 ≤≤256+127 512 ≤≤512+127 768 ≤≤768+127

21.8.6 Reset Values of Timing Parameters

Table 21-5 gives the default value of timing parameters at reset.

Table 21-5. Reset Values of Timing Parameters

Register	Reset Value	
SMC_SETUP	SMC_SETUP	All setup timings are set to 1
SMC_PULSE	SMC_PULSE	All pulse timings are set to 1
SMC_CYCLE	SMC_CYCLE	The read and write operation last 3 Master Clock cycles and provide one hold cycle
WRITE_MODE	1	Write is controlled with NWE
READ_MODE	1	Read is controlled with NRD

21.8.7 Usage Restriction

The SMC does not check the validity of the user-programmed parameters. If the sum of SETUP and PULSE parameters is larger than the corresponding CYCLE parameter, this leads to unpredictable behavior of the SMC.

For read operations:

Null but positive setup and hold of address and NRD and/or NCS can not be guaranteed at the memory interface because of the propagation delay of theses signals through external logic and pads. If positive setup and hold values must be verified, then it is strictly recommended to program non-null values so as to cover possible skews between address, NCS and NRD signals.

For write operations:

If a null hold value is programmed on NWE, the SMC can guarantee a positive hold of address, byte select lines, and NCS signal after the rising edge of NWE. This is true for WRITE_MODE = 1 only. See "Early Read Wait State" on page 147.

For read and write operations: a null value for pulse parameters is forbidden and may lead to unpredictable behavior.

In read and write cycles, the setup and hold time parameters are defined in reference to the address bus. For external devices that require setup and hold time between NCS and NRD signals (read), or between NCS and NWE signals (write), these setup and hold times must be converted into setup and hold times in reference to the address bus.

21.9 Automatic Wait States

Under certain circumstances, the SMC automatically inserts idle cycles between accesses to avoid bus contention or operation conflict.

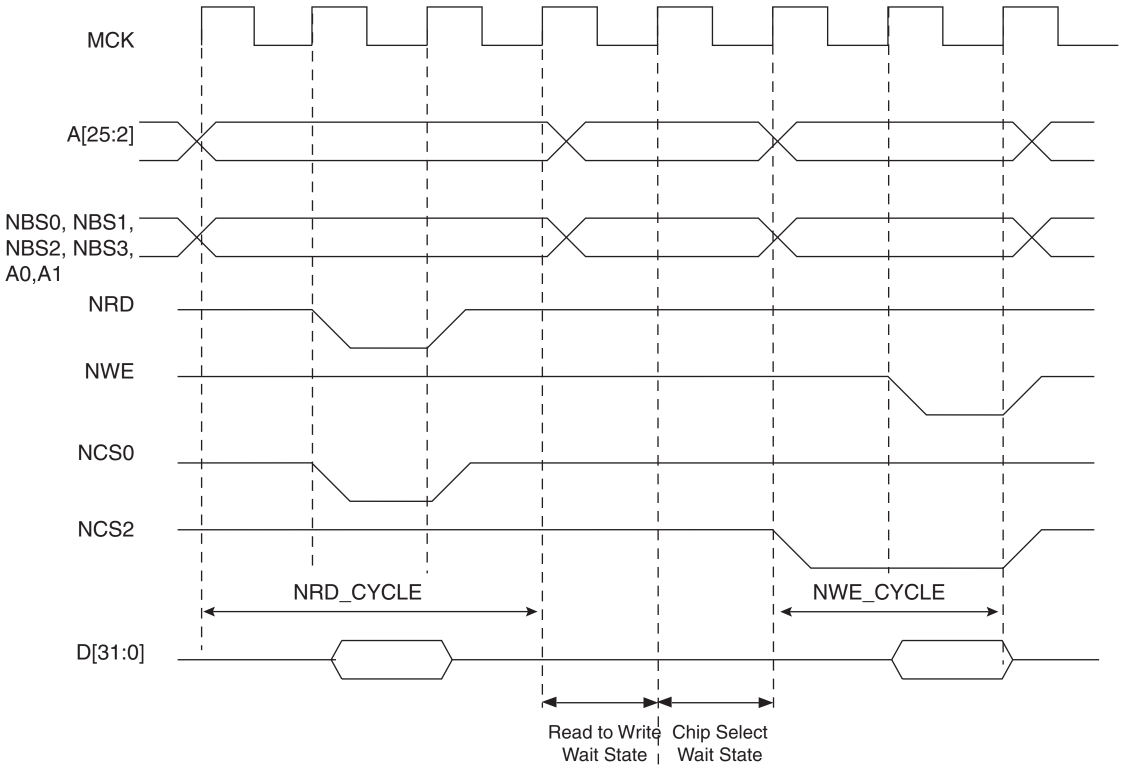
21.9.1 Chip Select Wait States

The SMC always inserts an idle cycle between 2 transfers on separate chip selects. This idle cycle ensures that there is no bus contention between the de-activation of one device and the activation of the next one.

During chip select wait state, all control lines are turned inactive: NBS0 to NBS3, NWR0 to NWR3, NCS[0..7], NRD lines are all set to 1.

Figure 21-16 illustrates a chip select wait state between access on Chip Select 0 and Chip Select 2.

Figure 21-16. Chip Select Wait State between a Read Access on NCS0 and a Write Access on NCS2



21.9.2 Early Read Wait State

In some cases, the SMC inserts a wait state cycle between a write access and a read access to allow time for the write cycle to end before the subsequent read cycle begins. This wait state is not generated in addition to a chip select wait state. The early read cycle thus only occurs between a write and read access to the same memory device (same chip select).

An early read wait state is automatically inserted if at least one of the following conditions is valid:

- if the write controlling signal has no hold time and the read controlling signal has no setup time (Figure 21-17).
- in NCS write controlled mode (WRITE_MODE = 0), if there is no hold timing on the NCS signal and the NCS_RD_SETUP parameter is set to 0, regardless of the read mode (Figure 21-18). The write operation must end with a NCS rising edge. Without an Early Read Wait State, the write operation could not complete properly.
- in NWE controlled mode (WRITE_MODE = 1) and if there is no hold timing (NWE_HOLD = 0), the feedback of the write control signal is used to control address, data, chip select and byte select lines. If the external write control signal is not inactivated as expected due to load capacitances, an Early Read Wait State is inserted and address, data and control signals are maintained one more cycle. See Figure 21-19.

Figure 21-17. Early Read Wait State: Write with No Hold Followed by Read with No Setup

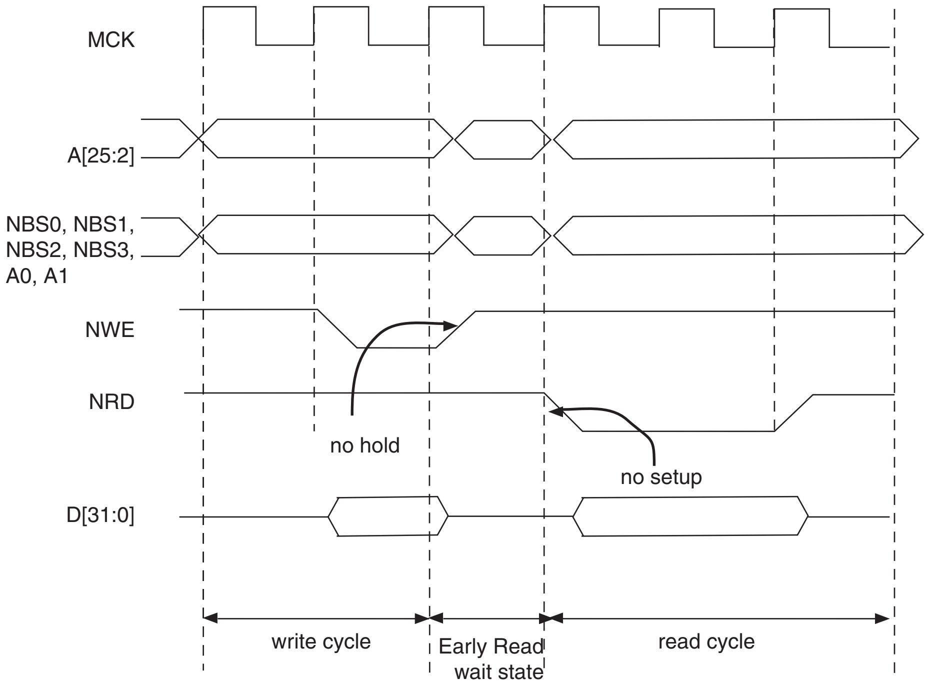

MCK
A[25:2]
NBS0, NBS1, NBS2, NBS3, A0, A1
NWE
NRD
no hold
no setup
D[31:0]
write cycle
Early Read wait state
read cycle

Figure 21-18. Early Read Wait State: NCS Controlled Write with No Hold Followed by a Read with No NCS Setup

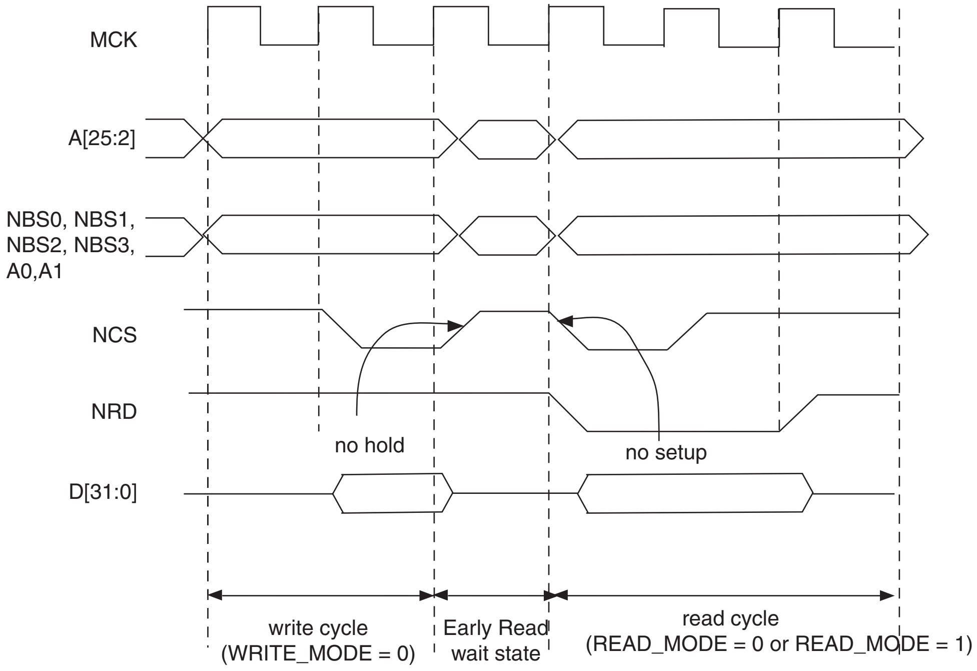
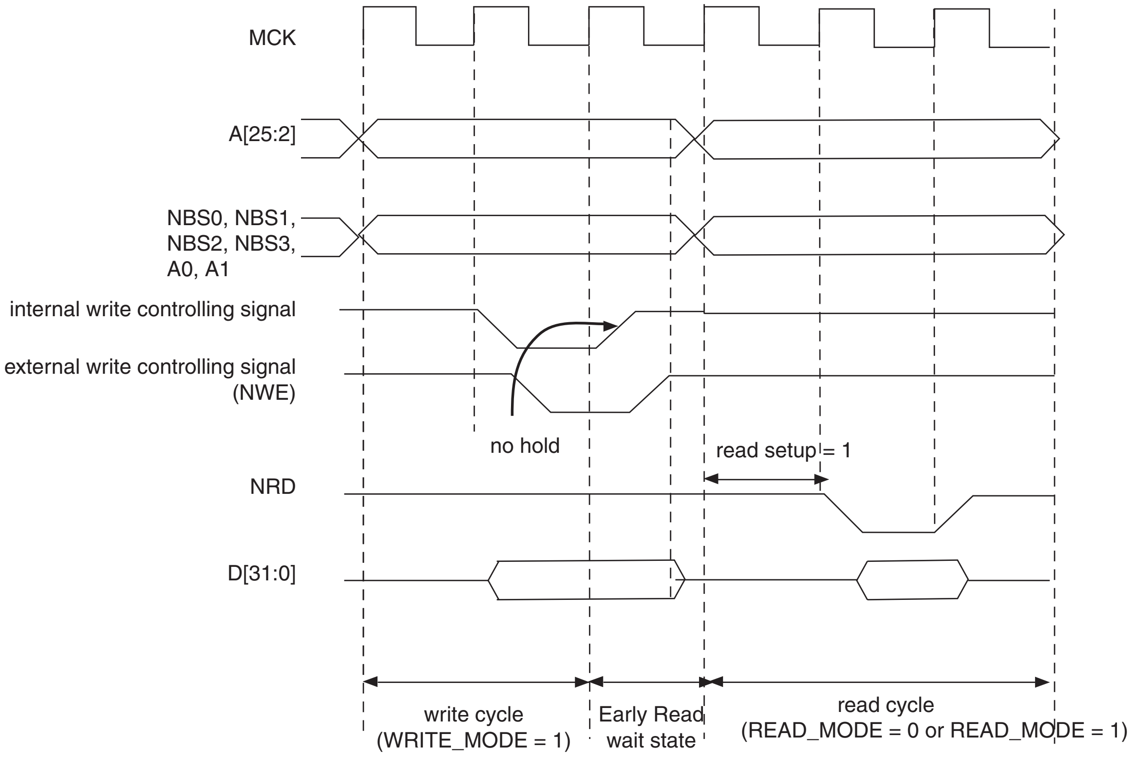

MCK
A[25:2]
NBS0, NBS1, NBS2, NBS3, A0,A1
NCS
NRD
no hold
no setup
D[31:0]
write cycle (WRITE_MODE = 0)
Early Read wait state
read cycle (READ_MODE = 0 or READ_MODE = 1)

Figure 21-19. Early Read Wait State: NWE-controlled Write with No Hold Followed by a Read with one Set-up Cycle


MCK
A[25:2]
NBS0, NBS1,
NBS2, NBS3,
A0, A1
internal write controlling signal
external write controlling signal
(NWE)
no hold
read setup = 1
NRD
D[31:0]
write cycle
(WRITE_MODE = 1)
Early Read
wait state
read cycle
(READ_MODE = 0 or READ_MODE = 1)

21.9.3 Reload User Configuration Wait State

The user may change any of the configuration parameters by writing the SMC user interface.

When detecting that a new user configuration has been written in the user interface, the SMC inserts a wait state before starting the next access. The so called "Reload User Configuration Wait State" is used by the SMC to load the new set of parameters to apply to next accesses.

The Reload Configuration Wait State is not applied in addition to the Chip Select Wait State. If accesses before and after re-programming the user interface are made to different devices (Chip Selects), then one single Chip Select Wait State is applied.

On the other hand, if accesses before and after writing the user interface are made to the same device, a Reload Configuration Wait State is inserted, even if the change does not concern the current Chip Select.

21.9.3.1 User Procedure

To insert a Reload Configuration Wait State, the SMC detects a write access to any SMC_MODE register of the user interface. If the user only modifies timing registers (SMC_SETUP, SMC_PULSE, SMC_CYCLE registers) in the user interface, he must validate the modification by writing the SMC_MODE, even if no change was made on the mode parameters.

21.9.3.2 Slow Clock Mode Transition

A Reload Configuration Wait State is also inserted when the Slow Clock Mode is entered or exited, after the end of the current transfer (see "Slow Clock Mode" on page 160).

21.9.4 Read to Write Wait State

Due to an internal mechanism, a wait cycle is always inserted between consecutive read and write SMC accesses.

This wait cycle is referred to as a read to write wait state in this document.

This wait cycle is applied in addition to chip select and reload user configuration wait states when they are to be inserted. See Figure 21-16 on page 147.

21.10 Data Float Wait States

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

- before starting a read access to a different external memory
- before starting a write access to the same device or to a different external one.

The Data Float Output Time (t_{DF}) for each external memory device is programmed in the TDF_CYCLES field of the SMC_MODE register for the corresponding chip select. The value of TDF_CYCLES indicates the number of data float wait cycles (between 0 and 15) before the external device releases the bus, and represents the time allowed for the data output to go to high impedance after the memory is disabled.

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

The data float wait states management depends on the READ_MODE and the TDF_MODE fields of the SMC_MODE register for the corresponding chip select.

21.10.1 READ_MODE

Setting the READ_MODE to 1 indicates to the SMC that the NRD signal is responsible for turning off the tri-state buffers of the external memory device. The Data Float Period then begins after the rising edge of the NRD signal and lasts TDF_CYCLES MCK cycles.

When the read operation is controlled by the NCS signal (READ_MODE = 0), the TDF field gives the number of MCK cycles during which the data bus remains busy after the rising edge of NCS.

Figure 21-20 illustrates the Data Float Period in NRD-controlled mode (READ_MODE =1), assuming a data float period of 2 cycles (TDF_CYCLES = 2). Figure 21-21 shows the read operation when controlled by NCS (READ_MODE = 0) and the TDF_CYCLES parameter equals 3.

Figure 21-20. TDF Period in NRD Controlled Read Access (TDF = 2)

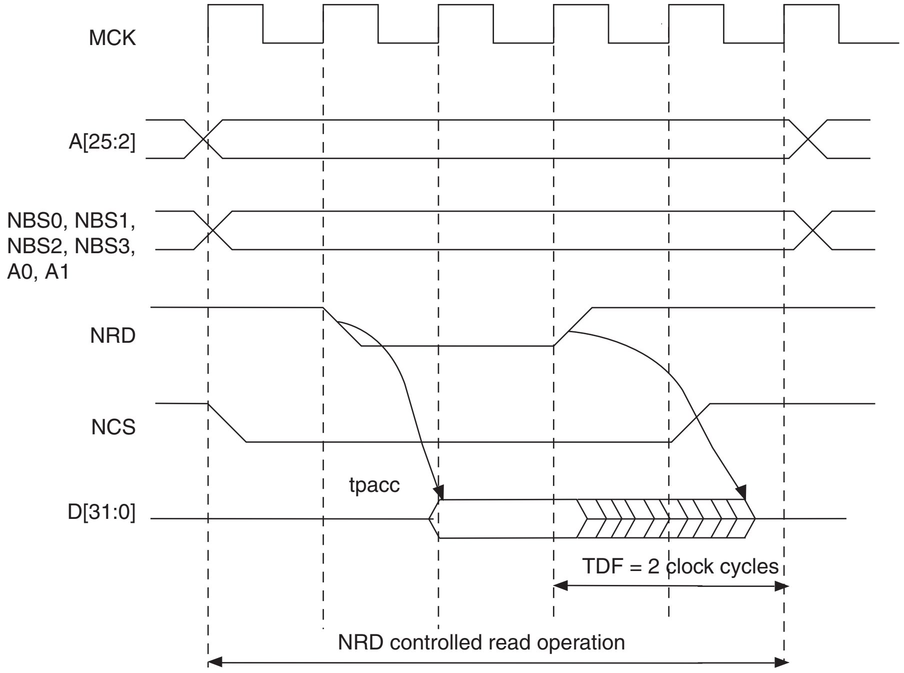
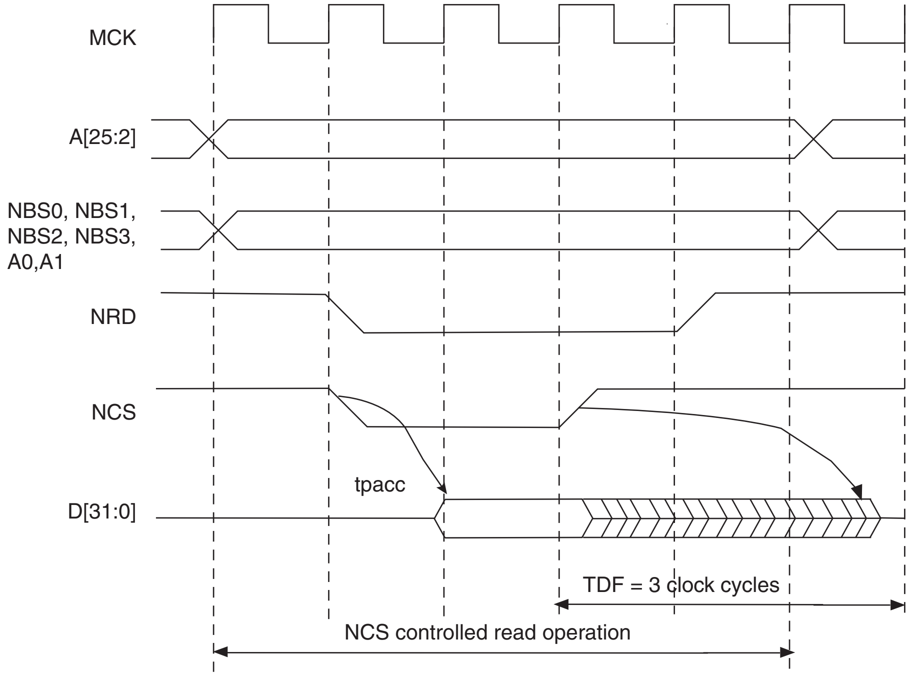

MCK
A[25:2]
NBS0, NBS1,
NBS2, NBS3,
A0, A1
NRD
NCS
tpacc
D[31:0]
TDF = 2 clock cycles
NRD controlled read operation

Figure 21-21. TDF Period in NCS Controlled Read Operation (TDF = 3)


MCK
A[25:2]
NBS0, NBS1,
NBS2, NBS3,
A0,A1
NRD
NCS
tpacc
D[31:0]
TDF = 3 clock cycles
NCS controlled read operation

21.10.2 TDF Optimization Enabled (TDF_MODE = 1)

When the TDF_MODE of the SMC_MODE register is set to 1 (TDF optimization is enabled), the SMC takes advantage of the setup period of the next access to optimize the number of wait states cycle to insert.

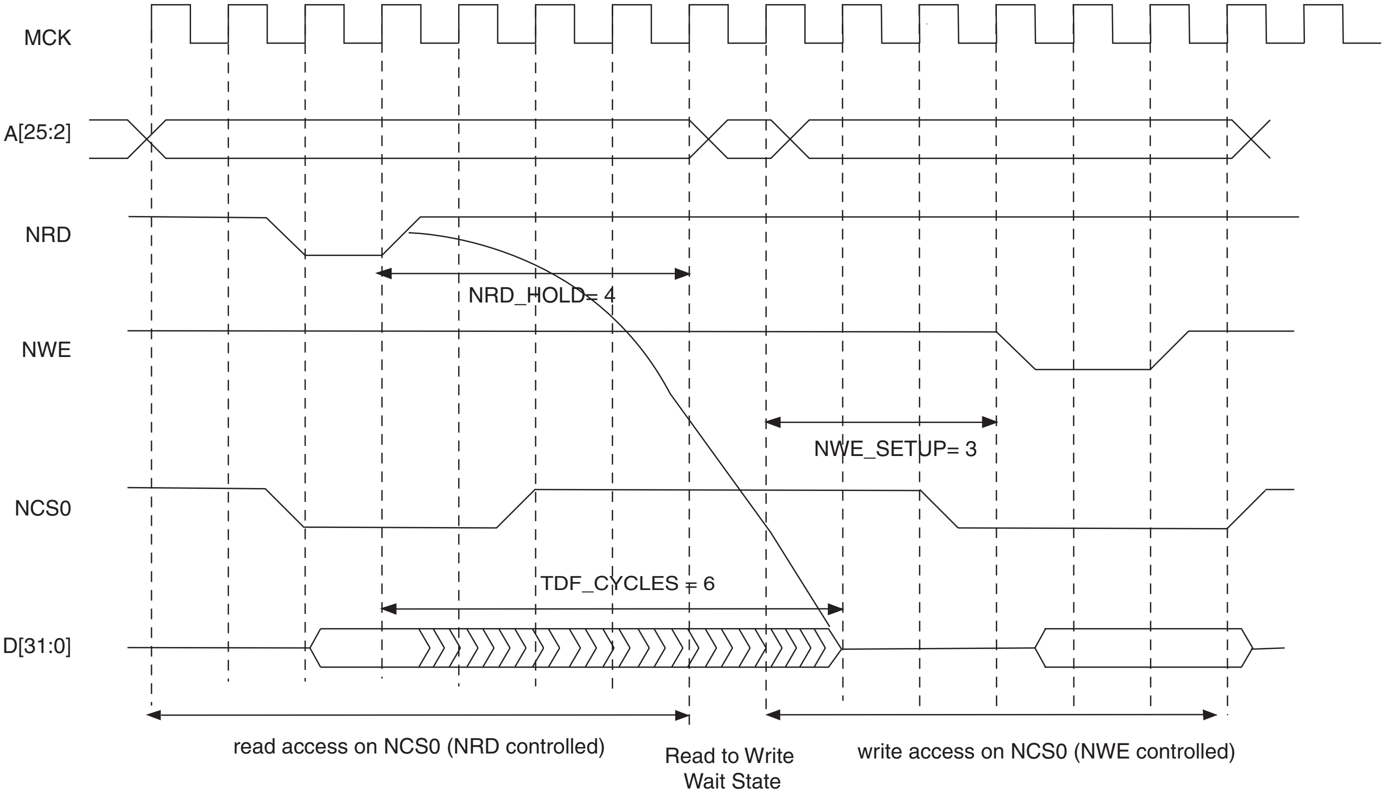
Figure 21-22 shows a read access controlled by NRD, followed by a write access controlled by NWE, on Chip Select 0. Chip Select 0 has been programmed with:

NRD_HOLD = 4; READ_MODE = 1 (NRD controlled)

NWE_SETUP = 3; WRITE_MODE = 1 (NWE controlled)

TDF_CYCLES = 6; TDF_MODE = 1 (optimization enabled).

Figure 21-22. TDF Optimization: No TDF wait states are inserted if the TDF period is over when the next access begins


MCK
A[25:2]
NRD
NRD_HOLD= 4
NWE
NWE_SETUP= 3
NCS0
TDF_CYCLES = 6
D[31:0]
read access on NCS0 (NRD controlled)
Read to Write
Wait State
write access on NCS0 (NWE controlled)

21.10.3 TDF Optimization Disabled (TDF_MODE = 0)

When optimization is disabled, tdf wait states are inserted at the end of the read transfer, so that the data float period is ended when the second access begins. If the hold period of the read1 controlling signal overlaps the data float period, no additional tdf wait states will be inserted.

Figure 21-23, Figure 21-24 and Figure 21-25 illustrate the cases:

- read access followed by a read access on another chip select,
- read access followed by a write access on another chip select,
- read access followed by a write access on the same chip select,

with no TDF optimization.

Figure 21-23. TDF Optimization Disabled (TDF Mode = 0). TDF wait states between 2 read accesses on different chip selects

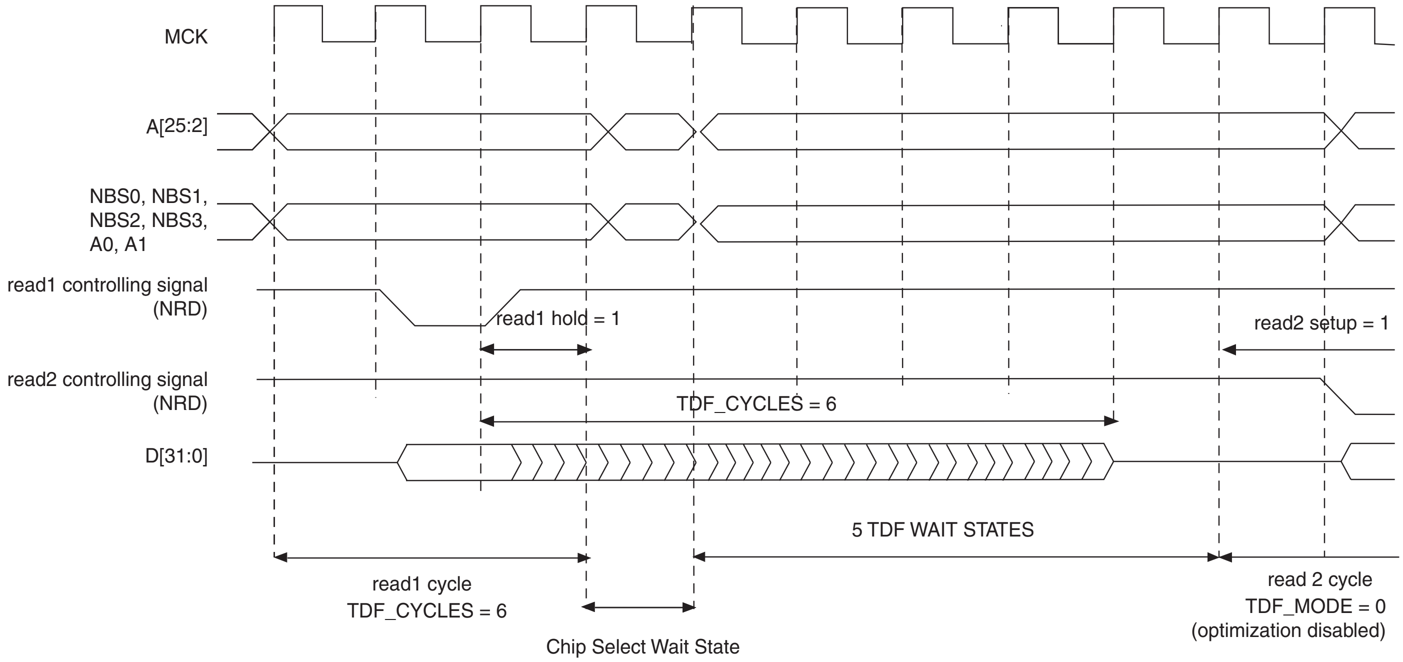

MCK
A[25:2]
NBS0, NBS1, NBS2, NBS3, A0, A1
read1 controlling signal (NRD)
read1 hold = 1
read2 setup = 1
read2 controlling signal (NRD)
TDF_CYCLES = 6
D[31:0]
5 TDF WAIT STATES
read1 cycle
TDF_CYCLES = 6
Chip Select Wait State
read 2 cycle
TDF_MODE = 0
(optimization disabled)

Figure 21-24. TDF Mode = 0: TDF wait states between a read and a write access on different chip selects

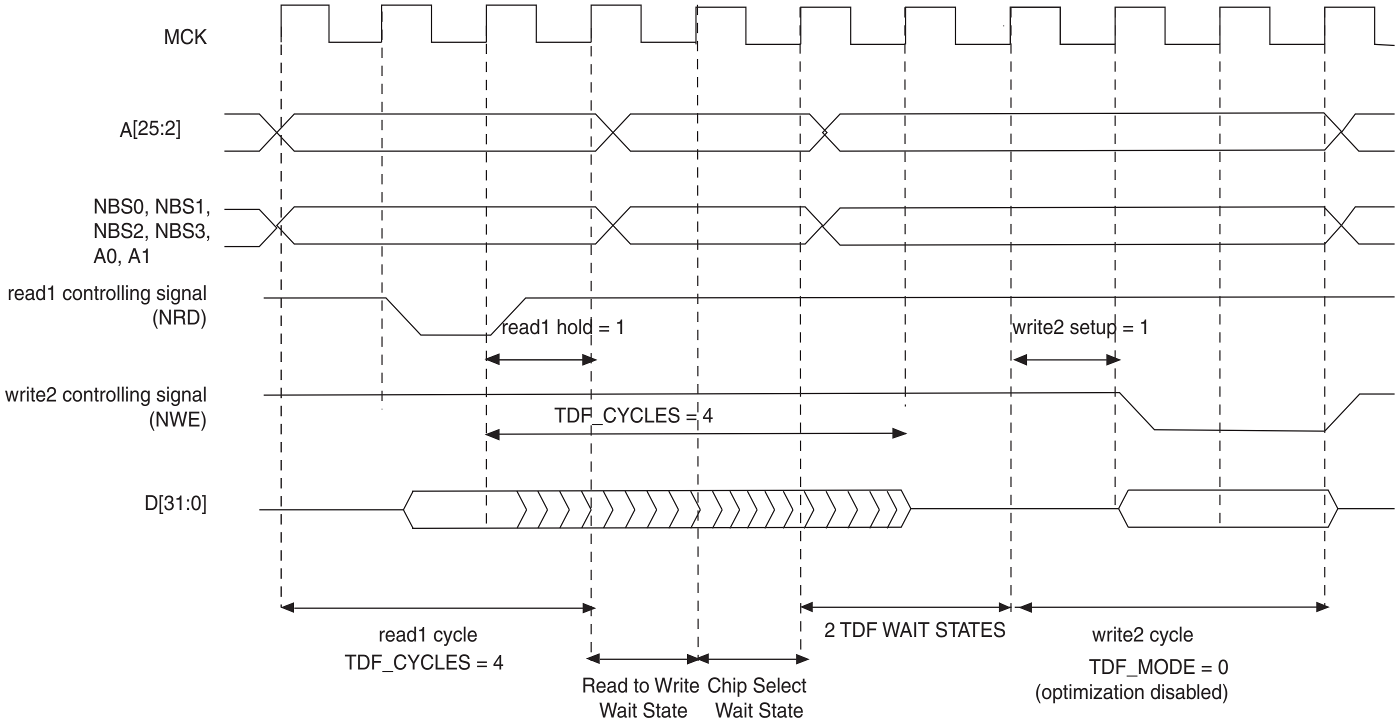
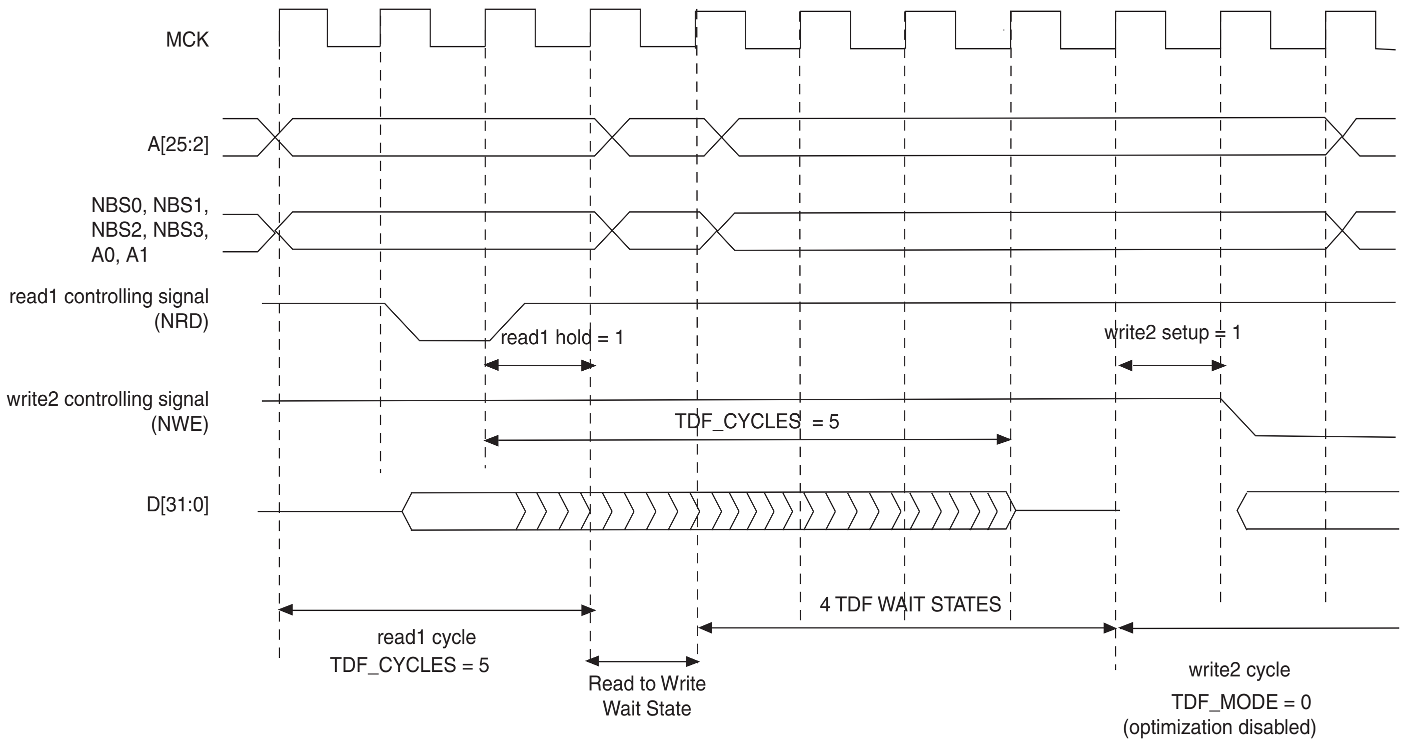

MCK
A[25:2]
NBS0, NBS1, NBS2, NBS3, A0, A1
read1 controlling signal (NRD)
read1 hold = 1
write2 setup = 1
write2 controlling signal (NWE)
TDF_CYCLES = 4
D[31:0]
read1 cycle
TDF_CYCLES = 4
Read to Write Wait State
Chip Select Wait State
2 TDF WAIT STATES
write2 cycle
TDF_MODE = 0
(optimization disabled)

Figure 21-25. TDF Mode = 0: TDF wait states between read and write accesses on the same chip select



MCK
A[25:2]
NBS0, NBS1, NBS2, NBS3, A0, A1
read1 controlling signal (NRD)
read1 hold = 1
write2 setup = 1
write2 controlling signal (NWE)
TDF_CYCLES = 5
D[31:0]
4 TDF WAIT STATES
read1 cycle
TDF_CYCLES = 5
Read to Write Wait State
write2 cycle
TDF_MODE = 0
(optimization disabled)

21.11 External Wait

Any access can be extended by an external device using the NWAIT input signal of the SMC. The EXNW_MODE field of the SMC_MODE register on the corresponding chip select must be set to either to "10" (frozen mode) or "11" (ready mode). When the EXNW_MODE is set to "00" (disabled), the NWAIT signal is simply ignored on the corresponding chip select. The NWAIT signal delays the read or write operation in regards to the read or write controlling signal, depending on the read and write modes of the corresponding chip select.

21.11.1 Restriction

When one of the EXNW_MODE is enabled, **it is mandatory to program at least one hold cycle for the read/write controlling signal. For that reason, the NWAIT signal cannot be used in Page Mode (**"Asynchronous Page Mode" on page 163**), or in Slow Clock Mode (**"Slow Clock Mode" on page 160**).**

The NWAIT signal is assumed to be a response of the external device to the read/write request of the SMC. Then NWAIT is examined by the SMC only in the pulse state of the read or write controlling signal. The assertion of the NWAIT signal outside the expected period has no impact on SMC behavior.

21.11.2 Frozen Mode

When the external device asserts the NWAIT signal (active low), and after internal synchronization of this signal, the SMC state is frozen, i.e., SMC internal counters are frozen, and all control signals remain unchanged. When the resynchronized NWAIT signal is deasserted, the SMC completes the access, resuming the access from the point where it was stopped. See Figure 21-26. This mode must be selected when the external device uses the NWAIT signal to delay the access and to freeze the SMC.

The assertion of the NWAIT signal outside the expected period is ignored as illustrated in Figure 21-27.

Figure 21-26. Write Access with NWAIT Assertion in Frozen Mode (EXNW_MODE = 10)

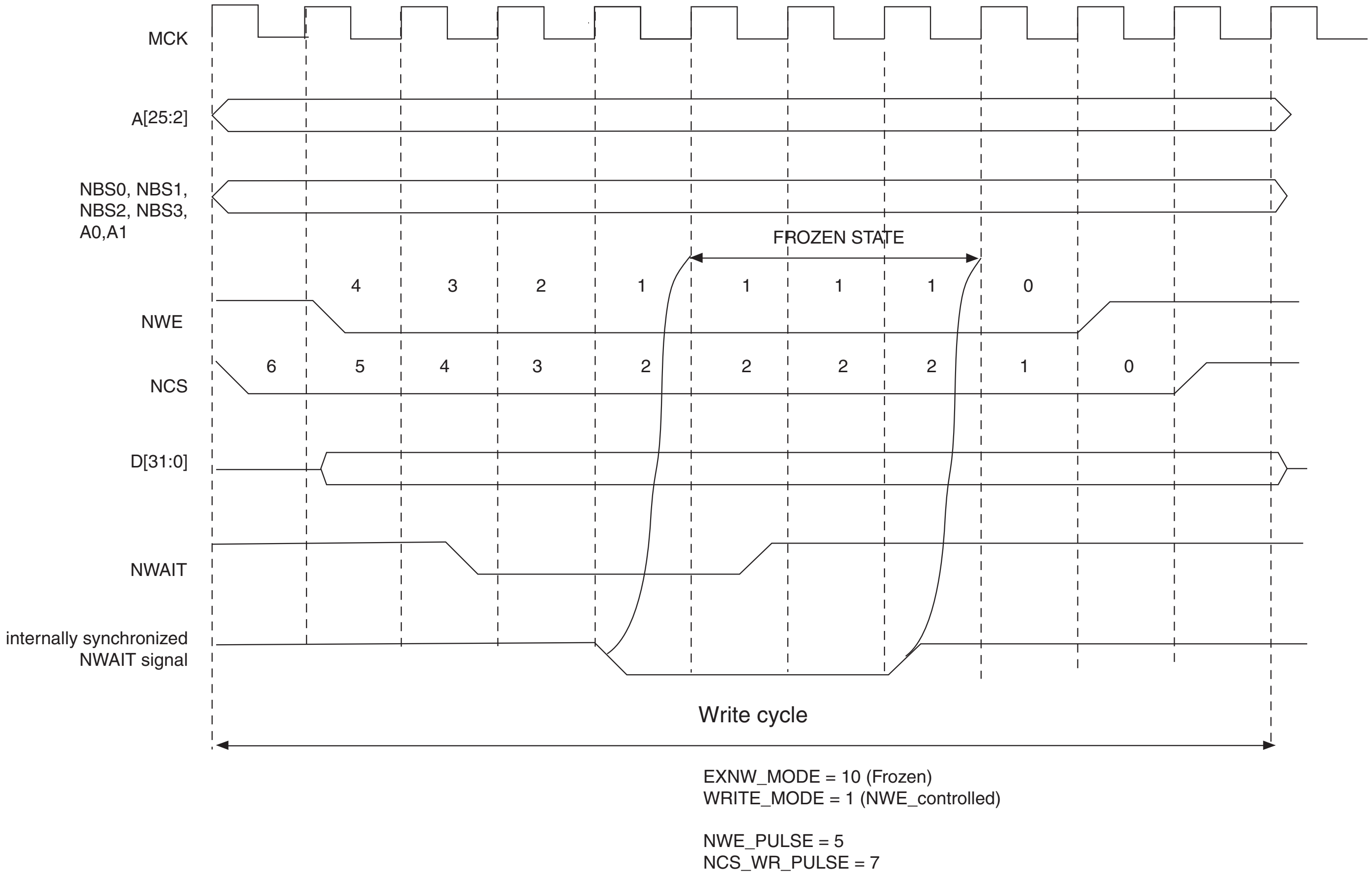
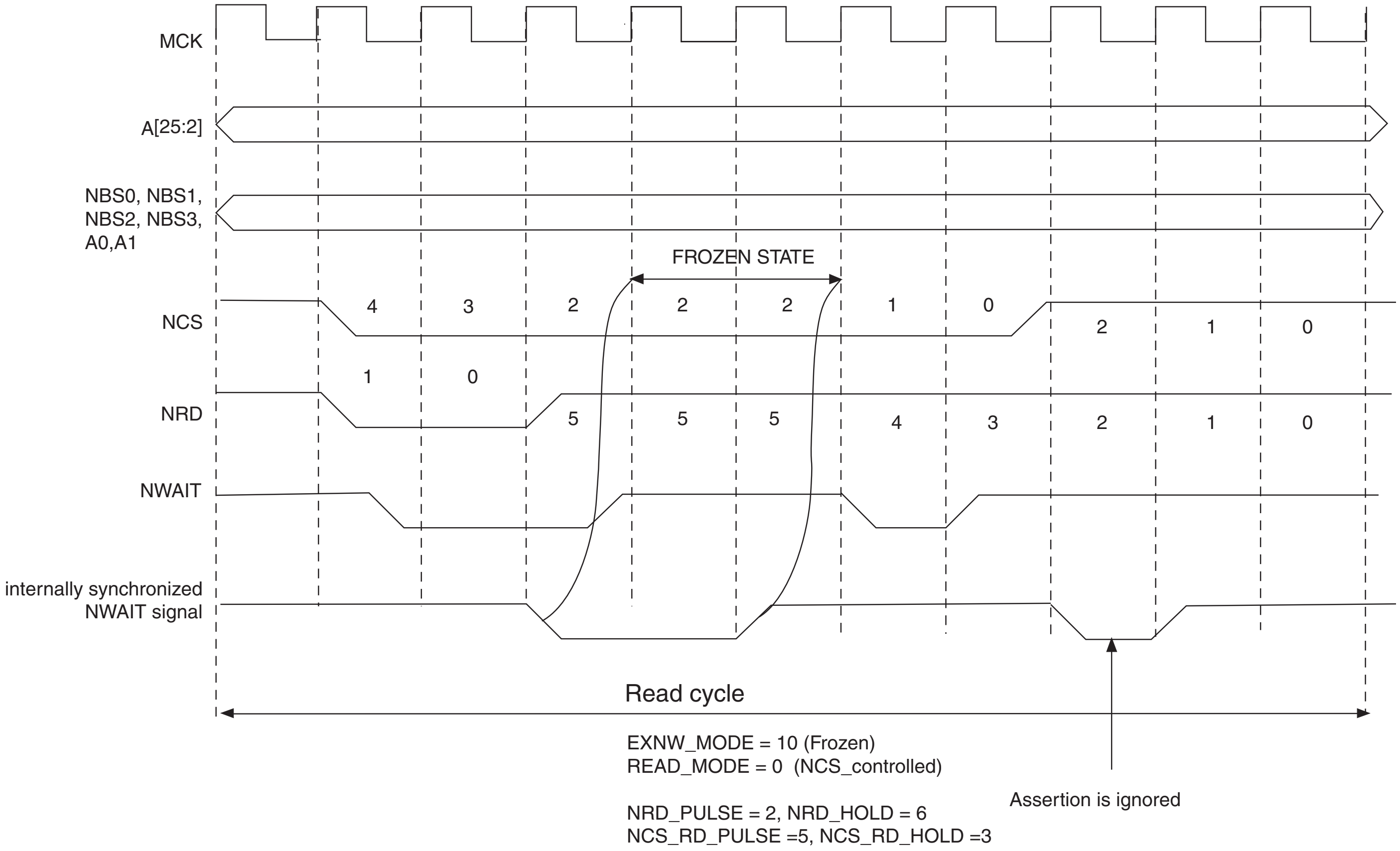

MCK
A[25:2]
NBS0, NBS1, NBS2, NBS3, A0,A1
FROZEN STATE
4
3
2
1
1
1
1
0
NWE
6
5
4
3
2
2
2
2
1
0
NCS
D[31:0]
NWAIT
internally synchronized NWAIT signal
Write cycle
EXNW_MODE = 10 (Frozen)
WRITE_MODE = 1 (NWE_controlled)
NWE_PULSE = 5
NCS_WR_PULSE = 7

Figure 21-27. Read Access with NWAIT Assertion in Frozen Mode (EXNW_MODE = 10)


MCK
A[25:2]
NBS0, NBS1, NBS2, NBS3, A0,A1
FROZEN STATE
NCS
4
3
2
2
2
1
0
2
1
0
NRD
1
0
5
5
5
4
3
2
1
0
NWAIT
internally synchronized NWAIT signal
Read cycle
EXNW_MODE = 10 (Frozen)
READ_MODE = 0 (NCS_controlled)
NRD_PULSE = 2, NRD_HOLD = 6
NCS_RD_PULSE =5, NCS_RD_HOLD =3
Assertion is ignored

21.11.3 Ready Mode

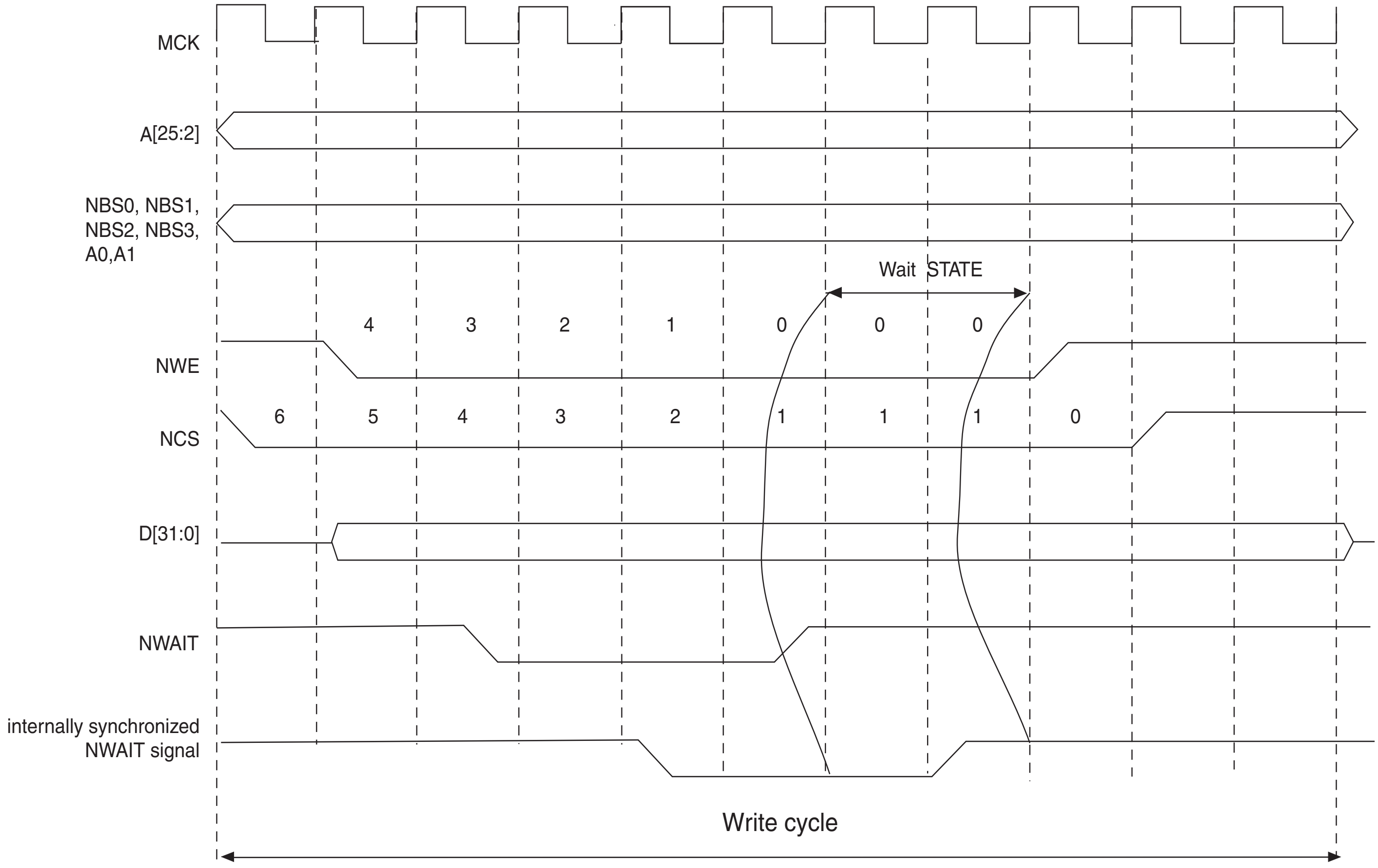
In Ready mode (EXNW_MODE = 11), the SMC behaves differently. Normally, the SMC begins the access by down counting the setup and pulse counters of the read/write controlling signal. In the last cycle of the pulse phase, the resynchronized NWAIT signal is examined.

If asserted, the SMC suspends the access as shown in Figure 21-28 and Figure 21-29. After deassertion, the access is completed: the hold step of the access is performed.

This mode must be selected when the external device uses deassertion of the NWAIT signal to indicate its ability to complete the read or write operation.

If the NWAIT signal is deasserted before the end of the pulse, or asserted after the end of the pulse of the controlling read/write signal, it has no impact on the access length as shown in Figure 21-29.

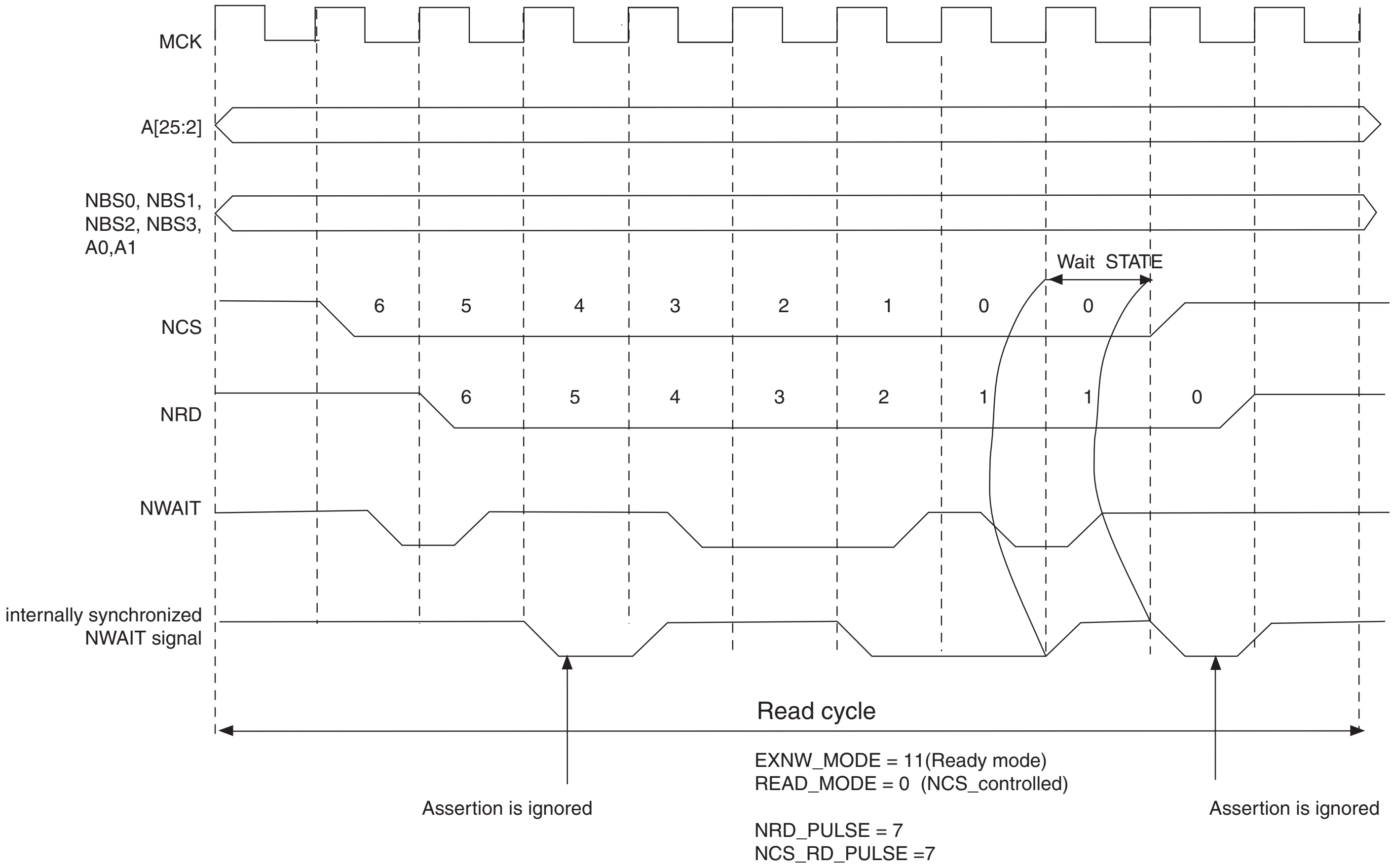
Figure 21-28. NWAIT Assertion in Write Access: Ready Mode (EXNW_MODE = 11)


MCK
A[25:2]
NBS0, NBS1, NBS2, NBS3, A0,A1
Wait STATE
NWE
4 3 2 1 0 0 0
NCS
6 5 4 3 2 1 1 1 0
D[31:0]
NWAIT
internally synchronized NWAIT signal
Write cycle

EXNW_MODE = 11 (Ready mode)
WRITE_MODE = 1 (NWE_controlled)

NWE_PULSE = 5
NCS_WR_PULSE = 7

Figure 21-29. NWAIT Assertion in Read Access: Ready Mode (EXNW_MODE = 11)


MCK
A[25:2]
NBS0, NBS1,
NBS2, NBS3,
A0,A1
Wait STATE
NCS
6
5
4
3
2
1
0
0
NRD
6
5
4
3
2
1
1
0
NWAIT
internally synchronized
NWAIT signal
Read cycle
EXNW_MODE = 11(Ready mode)
READ_MODE = 0 (NCS_controlled)
NRD_PULSE = 7
NCS_RD_PULSE =7
Assertion is ignored
Assertion is ignored

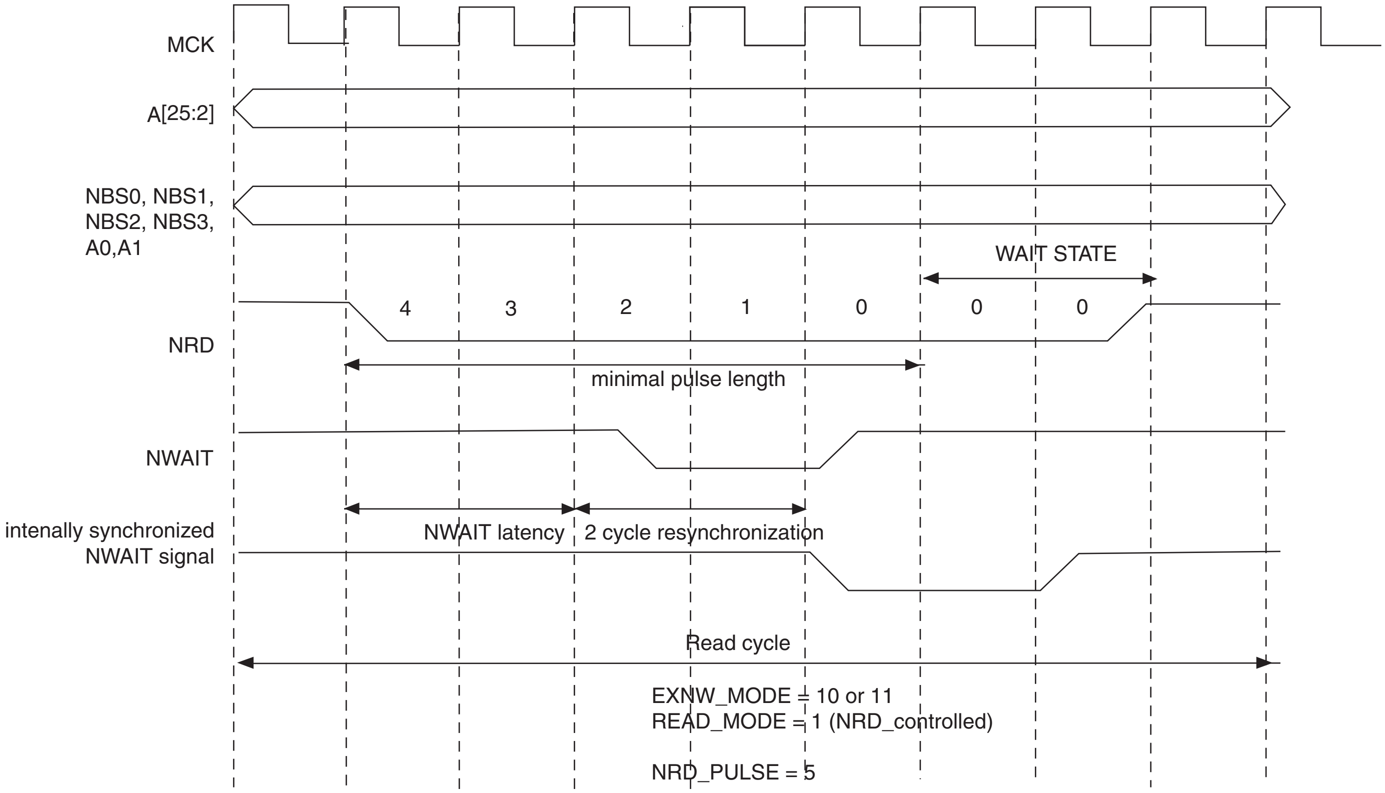
21.11.4 NWAIT Latency and Read/write Timings

There may be a latency between the assertion of the read/write controlling signal and the assertion of the NWAIT signal by the device. The programmed pulse length of the read/write controlling signal must be at least equal to this latency plus the 2 cycles of resynchronization + 1 cycle. Otherwise, the SMC may enter the hold state of the access without detecting the NWAIT signal assertion. This is true in frozen mode as well as in ready mode. This is illustrated on Figure 21-30.

When EXNW_MODE is enabled (ready or frozen), the user must program a pulse length of the read and write controlling signal of at least:

minimal pulse length = NWAIT latency + 2 resynchronization cycles + 1 cycle

Figure 21-30. NWAIT Latency


MCK
A[25:2]
NBS0, NBS1,
NBS2, NBS3,
A0,A1
WAIT STATE
4
3
2
1
0
0
0
NRD
minimal pulse length
NWAIT
intenally synchronized
NWAIT signal
NWAIT latency
2 cycle resynchronization
Read cycle
EXNW_MODE = 10 or 11
READ_MODE = 1 (NRD_controlled)
NRD_PULSE = 5

21.12 Slow Clock Mode

The SMC is able to automatically apply a set of “slow clock mode” read/write waveforms when an internal signal driven by the Power Management Controller is asserted because MCK has been turned to a very slow clock rate (typically 32kHz clock rate). In this mode, the user-programmed waveforms are ignored and the slow clock mode waveforms are applied. This mode is provided so as to avoid reprogramming the User Interface with appropriate waveforms at very slow clock rate. When activated, the slow mode is active on all chip selects.

21.12.1 Slow Clock Mode Waveforms

Figure 21-31 illustrates the read and write operations in slow clock mode. They are valid on all chip selects. Table 21-6 indicates the value of read and write parameters in slow clock mode.

Figure 21-31. Read/write Cycles in Slow Clock Mode

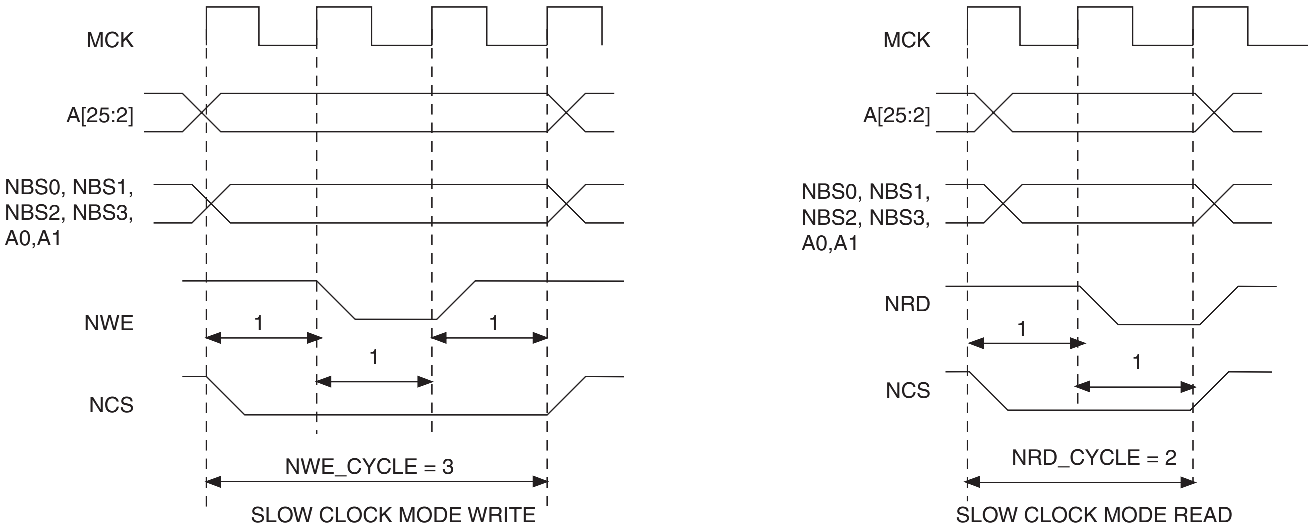

MCK
A[25:2]
NBS0, NBS1,
NBS2, NBS3,
A0,A1
NWE
1
1
1
NCS
NWE_CYCLE = 3
SLOW CLOCK MODE WRITE
MCK
A[25:2]
NBS0, NBS1,
NBS2, NBS3,
A0,A1
NRD
1
1
NCS
NRD_CYCLE = 2
SLOW CLOCK MODE READ

Table 21-6. Read and Write Timing Parameters in Slow Clock Mode

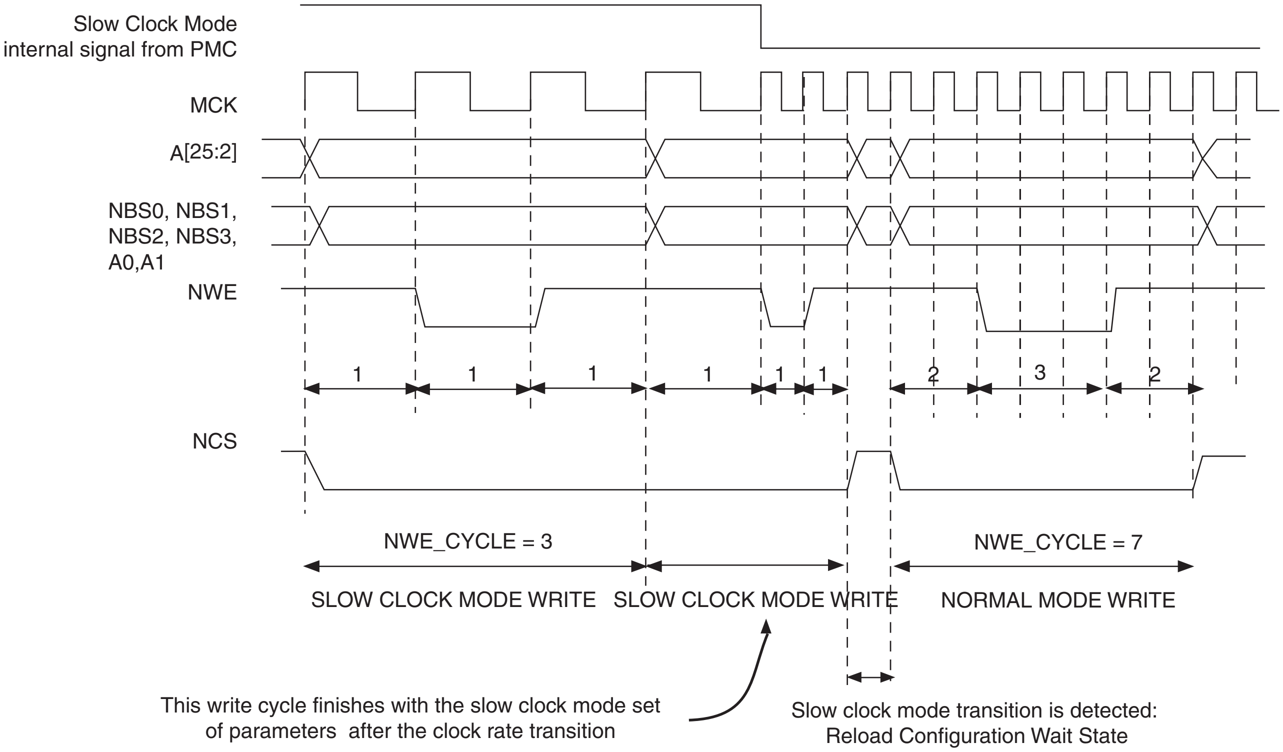
Read Parameters	Duration (cycles)	Write Parameters	Duration (cycles)
NRD_SETUP	1	NWE_SETUP	1
NRD_PULSE	1	NWE_PULSE	1
NCS_RD_SETUP	0	NCS_WR_SETUP	0
NCS_RD_PULSE	2	NCS_WR_PULSE	3
NRD_CYCLE	2	NWE_CYCLE	3

21.12.2 Switching from (to) Slow Clock Mode to (from) Normal Mode

When switching from slow clock mode to the normal mode, the current slow clock mode transfer is completed at high clock rate, with the set of slow clock mode parameters.See Figure 21-32 on page 161. The external device may not be fast enough to support such timings.

Figure 21-33 illustrates the recommended procedure to properly switch from one mode to the other.

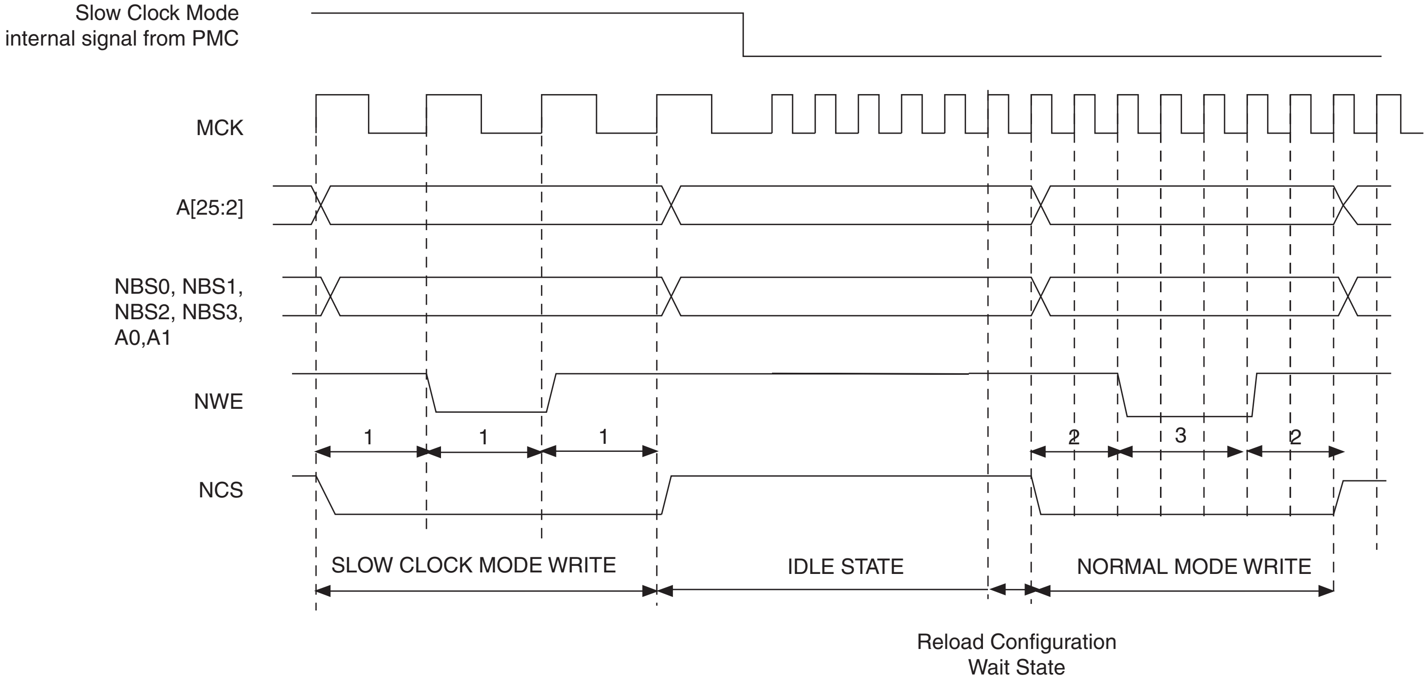
Figure 21-32. Clock Rate Transition Occurs while the SMC is Performing a Write Operation


Slow Clock Mode
internal signal from PMC
MCK
A[25:2]
NBS0, NBS1,
NBS2, NBS3,
A0,A1
NWE
1
1
1
1
1
1
2
3
2
NCS
NWE_CYCLE = 3
NWE_CYCLE = 7
SLOW CLOCK MODE WRITE
SLOW CLOCK MODE WRITE
NORMAL MODE WRITE

This write cycle finishes with the slow clock mode set of parameters after the clock rate transition

Slow clock mode transition is detected: Reload Configuration Wait State

Figure 21-33. Recommended Procedure to Switch from Slow Clock Mode to Normal Mode or from Normal Mode to Slow Clock Mode


Slow Clock Mode
internal signal from PMC
MCK
A[25:2]
NBS0, NBS1,
NBS2, NBS3,
A0,A1
NWE
1
1
1
2
3
2
NCS
SLOW CLOCK MODE WRITE
IDLE STATE
NORMAL MODE WRITE
Reload Configuration
Wait State

21.13 Asynchronous Page Mode

The SMC supports asynchronous burst reads in page mode, providing that the page mode is enabled in the SMC_MODE register (PMEN field). The page size must be configured in the SMC_MODE register (PS field) to 4, 8, 16 or 32 bytes.

The page defines a set of consecutive bytes into memory. A 4-byte page (resp. 8-, 16-, 32-byte page) is always aligned to 4-byte boundaries (resp. 8-, 16-, 32-byte boundaries) of memory. The MSB of data address defines the address of the page in memory, the LSB of address define the address of the data in the page as detailed in Table 21-7.

With page mode memory devices, the first access to one page (t_{pa}) takes longer than the subsequent accesses to the page (t_{sa}) as shown in Figure 21-34. When in page mode, the SMC enables the user to define different read timings for the first access within one page, and next accesses within the page.

Table 21-7. Page Address and Data Address within a Page

Page Size	**Page Address**(1)	Data Address in the Page(2)
4 bytes	A[25:2]	A[1:0]
8 bytes	A[25:3]	A[2:0]
16 bytes	A[25:4]	A[3:0]
32 bytes	A[25:5]	A[4:0]

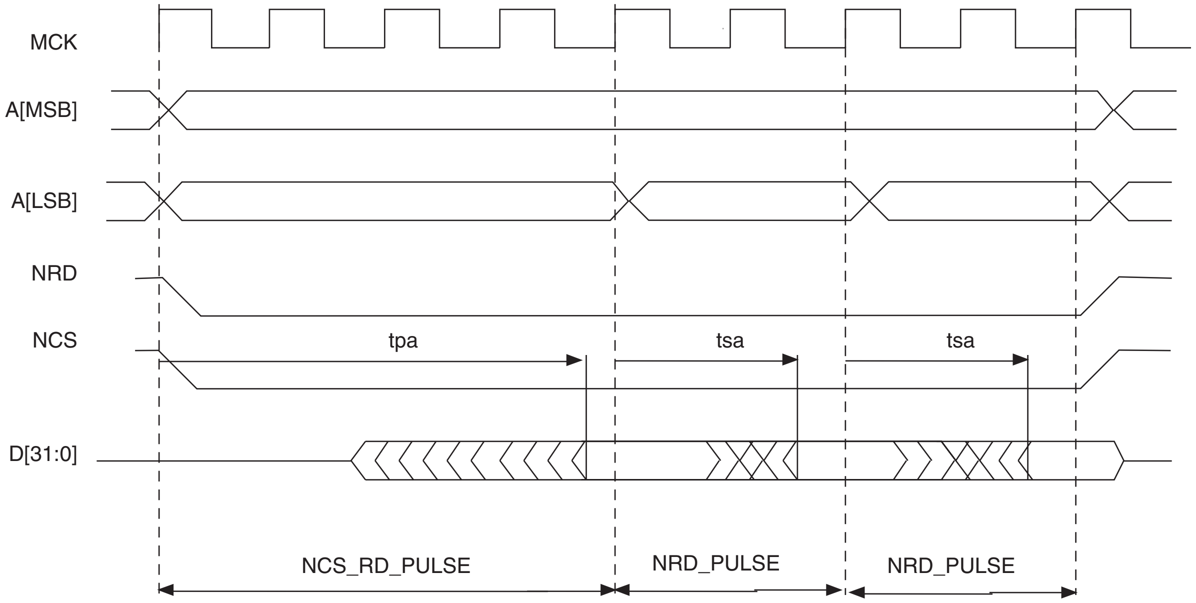
Notes: 1. A denotes the address bus of the memory device

2. For 16-bit devices, the bit 0 of address is ignored. For 32-bit devices, bits [1:0] are ignored.

21.13.1 Protocol and Timings in Page Mode

Figure 21-34 shows the NRD and NCS timings in page mode access.

Figure 21-34. Page Mode Read Protocol (Address MSB and LSB are defined in Table 21-7)



The NRD and NCS signals are held low during all read transfers, whatever the programmed values of the setup and hold timings in the User Interface may be. Moreover, the NRD and NCS timings are identical. The pulse length of the first access to the page is defined with the NCS_RD_PULSE field of the SMC_PULSE register. The pulse length of subsequent accesses within the page are defined using the NRD_PULSE parameter.

In page mode, the programming of the read timings is described in Table 21-8:

Table 21-8. Programming of Read Timings in Page Mode

Parameter	Value	Definition
READ_MODE	‘x’	No impact
NCS_RD_SETUP	‘x’	No impact
NCS_RD_PULSE	t_{pa}	Access time of first access to the page
NRD_SETUP	‘x’	No impact
NRD_PULSE	t_{sa}	Access time of subsequent accesses in the page
NRD_CYCLE	‘x’	No impact

The SMC does not check the coherency of timings. It will always apply the NCS_RD_PULSE timings as page access timing (t_{pa}) and the NRD_PULSE for accesses to the page (t_{sa}), even if the programmed value for t_{pa} is shorter than the programmed value for t_{sa}.

21.13.2 Byte Access Type in Page Mode

The Byte Access Type configuration remains active in page mode. For 16-bit or 32-bit page mode devices that require byte selection signals, configure the BAT field of the SMC_REGISTER to 0 (byte select access type).

21.13.3 Page Mode Restriction

The page mode is not compatible with the use of the NWAIT signal. Using the page mode and the NWAIT signal may lead to unpredictable behavior.

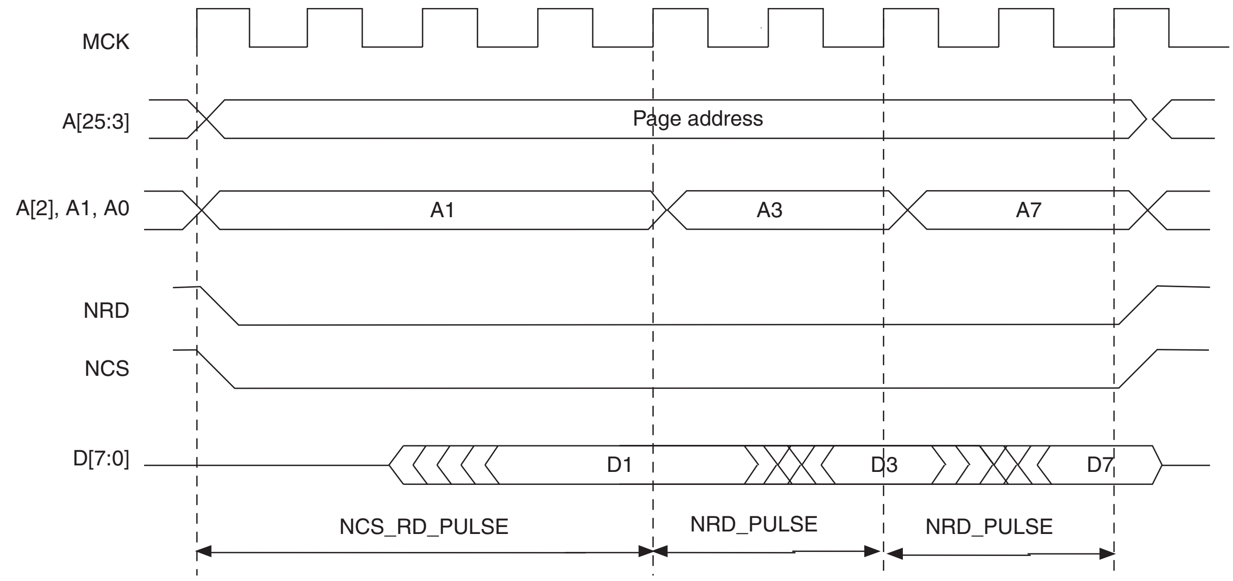
21.13.4 Sequential and Non-sequential Accesses

If the chip select and the MSB of addresses as defined in Table 21-7 are identical, then the current access lies in the same page as the previous one, and no page break occurs.

Using this information, all data within the same page, sequential or not sequential, are accessed with a minimum access time (t_{sa}). Figure 21-35 illustrates access to an 8-bit memory device in page mode, with 8-byte pages. Access to D1 causes a page access with a long access time (t_{pa}). Accesses to D3 and D7, though they are not sequential accesses, only require a short access time (t_{sa}).

If the MSB of addresses are different, the SMC performs the access of a new page. In the same way, if the chip select is different from the previous access, a page break occurs. If two sequential accesses are made to the page mode memory, but separated by an other internal or external peripheral access, a page break occurs on the second access because the chip select of the device was deasserted between both accesses.

Figure 21-35. Access to Non-sequential Data within the Same Page


MCK
A[25:3]
Page address
A[2], A1, A0
A1
A3
A7
NRD
NCS
D[7:0]
D1
D3
D7
NCS_RD_PULSE
NRD_PULSE
NRD_PULSE

21.14 Static Memory Controller (SMC) User Interface

The SMC is programmed using the registers listed in Table 21-9. For each chip select, a set of 4 registers is used to program the parameters of the external device connected on it. In Table 21-9, "CS_number" denotes the chip select number. 16 bytes (0x10) are required per chip select.

The user must complete writing the configuration by writing any one of the SMC_MODE registers.

Table 21-9. SMC Register Mapping

Offset	Register	Name	Access	Reset State
0x10 x CS_number + 0x00	SMC Setup Register	SMC_SETUP	Read/Write	SMC_SETUP
0x10 x CS_number + 0x04	SMC Pulse Register	SMC_PULSE	Read/Write	SMC_PULSE
0x10 x CS_number + 0x08	SMC Cycle Register	SMC_CYCLE	Read/Write	SMC_CYCLE
0x10 x CS_number + 0x0C	SMC Mode Register	SMC_MODE	Read/Write	SMC_MODE

21.14.1 SMC Setup Register

Register Name: SMC_SETUP[0 ..7]

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	NCS_RD_SETUP					

23	22	21	20	19	18	17	16
–	–	NRD_SETUP					

15	14	13	12	11	10	9	8
–	–	NCS_WR_SETUP					

7	6	5	4	3	2	1	0
–	–	NWE_SETUP					

- **NWE_SETUP: NWE Setup Length**

The NWE signal setup length is defined as:

NWE setup length = (128* NWE_SETUP[5] + NWE_SETUP[4:0]) clock cycles

- **NCS_WR_SETUP: NCS Setup Length in WRITE Access**

In write access, the NCS signal setup length is defined as:

NCS setup length = (128* NCS_WR_SETUP[5] + NCS_WR_SETUP[4:0]) clock cycles

- **NRD_SETUP: NRD Setup Length**

The NRD signal setup length is defined in clock cycles as:

NRD setup length = (128* NRD_SETUP[5] + NRD_SETUP[4:0]) clock cycles

- **NCS_RD_SETUP: NCS Setup Length in READ Access**

In read access, the NCS signal setup length is defined as:

NCS setup length = (128* NCS_RD_SETUP[5] + NCS_RD_SETUP[4:0]) clock cycles

21.14.2 SMC Pulse Register

Register Name: SMC_PULSE[0..7]

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	NCS_RD_PULSE						

23	22	21	20	19	18	17	16
–	NRD_PULSE						

15	14	13	12	11	10	9	8
–	NCS_WR_PULSE						

7	6	5	4	3	2	1	0
–	NWE_PULSE						

- **NWE_PULSE: NWE Pulse Length**

The NWE signal pulse length is defined as:

NWE pulse length = (256* NWE_PULSE[6] + NWE_PULSE[5:0]) clock cycles

The NWE pulse length must be at least 1 clock cycle.

- **NCS_WR_PULSE: NCS Pulse Length in WRITE Access**

In write access, the NCS signal pulse length is defined as:

NCS pulse length = (256* NCS_WR_PULSE[6] + NCS_WR_PULSE[5:0]) clock cycles

The NCS pulse length must be at least 1 clock cycle.

- **NRD_PULSE: NRD Pulse Length**

In standard read access, the NRD signal pulse length is defined in clock cycles as:

NRD pulse length = (256* NRD_PULSE[6] + NRD_PULSE[5:0]) clock cycles

The NRD pulse length must be at least 1 clock cycle.

In page mode read access, the NRD_PULSE parameter defines the duration of the subsequent accesses in the page.

- **NCS_RD_PULSE: NCS Pulse Length in READ Access**

In standard read access, the NCS signal pulse length is defined as:

NCS pulse length = (256* NCS_RD_PULSE[6] + NCS_RD_PULSE[5:0]) clock cycles

The NCS pulse length must be at least 1 clock cycle.

In page mode read access, the NCS_RD_PULSE parameter defines the duration of the first access to one page.

21.14.3 SMC Cycle Register

Register Name: SMC_CYCLE[0..7]

Access Type: Read/Write

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

23	22	21	20	19	18	17	16
NRD_CYCLE							

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	NWE_CYCLE

7	6	5	4	3	2	1	0
NWE_CYCLE							

- **NWE_CYCLE: Total Write Cycle Length**

The total write cycle length is the total duration in clock cycles of the write cycle. It is equal to the sum of the setup, pulse and hold steps of the NWE and NCS signals. It is defined as:

Write cycle length = (NWE_CYCLE[8:7]*256 + NWE_CYCLE[6:0]) clock cycles

- **NRD_CYCLE: Total Read Cycle Length**

The total read cycle length is the total duration in clock cycles of the read cycle. It is equal to the sum of the setup, pulse and hold steps of the NRD and NCS signals. It is defined as:

Read cycle length = (NRD_CYCLE[8:7]*256 + NRD_CYCLE[6:0]) clock cycles

21.14.4 SMC MODE Register

Register Name: SMC_MODE[0..7]

Access Type: Read/Write

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

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

15	14	13	12	11	10	9	8
–	–	DBW		–	–	–	BAT

7	6	5	4	3	2	1	0
–	–	EXNW_MODE		–	–	WRITE_MODE	READ_MODE

- **READ_MODE:**

1: The read operation is controlled by the NRD signal.

  - If TDF cycles are programmed, the external bus is marked busy after the rising edge of NRD.
  - If TDF optimization is enabled (TDF_MODE =1), TDF wait states are inserted after the setup of NRD.

0: The read operation is controlled by the NCS signal.

  - If TDF cycles are programmed, the external bus is marked busy after the rising edge of NCS.
  - If TDF optimization is enabled (TDF_MODE =1), TDF wait states are inserted after the setup of NCS.

- **WRITE_MODE**

1: The write operation is controlled by the NWE signal.

  - If TDF optimization is enabled (TDF_MODE =1), TDF wait states will be inserted after the setup of NWE.

0: The write operation is controlled by the NCS signal.

  - If TDF optimization is enabled (TDF_MODE =1), TDF wait states will be inserted after the setup of NCS.

- **EXNW_MODE: NWAIT Mode**

The NWAIT signal is used to extend the current read or write signal. It is only taken into account during the pulse phase of the read and write controlling signal. When the use of NWAIT is enabled, at least one cycle hold duration must be programmed for the read and write controlling signal.

EXNW_MODE		NWAIT Mode
0	0	Disabled
0	1	Reserved
1	0	Frozen Mode
1	1	Ready Mode

- Disabled Mode: The NWAIT input signal is ignored on the corresponding Chip Select.
- Frozen Mode: If asserted, the NWAIT signal freezes the current read or write cycle. After deassertion, the read/write cycle is resumed from the point where it was stopped.
- Ready Mode: The NWAIT signal indicates the availability of the external device at the end of the pulse of the controlling read or write signal, to complete the access. If high, the access normally completes. If low, the access is extended until NWAIT returns high.

- **BAT: Byte Access Type**

This field is used only if DBW defines a 16- or 32-bit data bus.

- 1: Byte write access type:
  - – Write operation is controlled using NCS, NWR0, NWR1, NWR2, NWR3.
  - – Read operation is controlled using NCS and NRD.
- 0: Byte select access type:
  - – Write operation is controlled using NCS, NWE, NBS0, NBS1, NBS2 and NBS3
  - – Read operation is controlled using NCS, NRD, NBS0, NBS1, NBS2 and NBS3

- **DBW: Data Bus Width**

DBW		Data Bus Width
0	0	8-bit bus
0	1	16-bit bus
1	0	32-bit bus
1	1	Reserved

- **TDF_CYCLES: Data Float Time**

This field gives the integer number of clock cycles required by the external device to release the data after the rising edge of the read controlling signal. The SMC always provide one full cycle of bus turnaround after the TDF_CYCLES period. The external bus cannot be used by another chip select during TDF_CYCLES + 1 cycles. From 0 up to 15 TDF_CYCLES can be set.

- **TDF_MODE: TDF Optimization**

1: TDF optimization is enabled.

– The number of TDF wait states is optimized using the setup period of the next read/write access.

0: TDF optimization is disabled.

– The number of TDF wait states is inserted before the next access begins.

- **PMEN: Page Mode Enabled**

1: Asynchronous burst read in page mode is applied on the corresponding chip select.

0: Standard read is applied.

- **PS: Page Size**

If page mode is enabled, this field indicates the size of the page in bytes.

PS		Page Size
0	0	4-byte page
0	1	8-byte page
1	0	16-byte page
1	1	32-byte page

22. SDRAM Controller (HSDRAMC)

22.1 Description

The SDRAM Controller (SDRAMC) extends the memory capabilities of a chip by providing the interface to an external 16-bit or 32-bit SDRAM device. The page size supports ranges from 2048 to 8192 and the number of columns from 256 to 2048. It supports byte (8-bit), half-word (16-bit) and word (32-bit) accesses.

The SDRAM Controller supports a read or write burst length of one location. It keeps track of the active row in each bank, thus maximizing SDRAM performance, e.g., the application may be placed in one bank and data in the other banks. So as to optimize performance, it is advisable to avoid accessing different rows in the same bank.

The SDRAM controller supports a CAS latency of 1, 2 or 3 and optimizes the read access depending on the frequency.

The different modes available - self-refresh, power-down and deep power-down modes - minimize power consumption on the SDRAM device.

22.2 I/O Lines Description

Table 22-1. I/O Line Description

Name	Description	Type	Active Level
SDCK	SDRAM Clock	Output	
SDCKE	SDRAM Clock Enable	Output	High
SDCS	SDRAM Controller Chip Select	Output	Low
BA[1:0]	Bank Select Signals	Output	
RAS	Row Signal	Output	Low
CAS	Column Signal	Output	Low
SDWE	SDRAM Write Enable	Output	Low
NBS[3:0]	Data Mask Enable Signals	Output	Low
SDRAMC_A[12:0]	Address Bus	Output	
D[31:0]	Data Bus	I/O	

22.3 Application Example

22.4 Software Interface

The SDRAM address space is organized into banks, rows, and columns. The SDRAM controller allows mapping different memory types according to the values set in the SDRAMC configuration register.

The SDRAM Controller's function is to make the SDRAM device access protocol transparent to the user. Table 22-2 to Table 22-7 illustrate the SDRAM device memory mapping seen by the user in correlation with the device structure. Various configurations are illustrated.

22.4.1 32-bit Memory Data Bus Width

Table 22-2. SDRAM Configuration Mapping: 2K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Bk[1:0]		Row[10:0]											Column[7:0]								M[1:0]	
				Bk[1:0]		Row[10:0]											Column[8:0]									M[1:0]	
			Bk[1:0]		Row[10:0]											Column[9:0]										M[1:0]	
		Bk[1:0]		Row[10:0]											Column[10:0]											M[1:0]	

Table 22-3. SDRAM Configuration Mapping: 4K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				Bk[1:0]		Row[11:0]												Column[7:0]								M[1:0]	
			Bk[1:0]		Row[11:0]												Column[8:0]									M[1:0]	
		Bk[1:0]		Row[11:0]												Column[9:0]										M[1:0]	
	Bk[1:0]		Row[11:0]												Column[10:0]											M[1:0]	

Table 22-4. SDRAM Configuration Mapping: 8K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			Bk[1:0]		Row[12:0]													Column[7:0]								M[1:0]	
		Bk[1:0]		Row[12:0]													Column[8:0]									M[1:0]	
	Bk[1:0]		Row[12:0]													Column[9:0]										M[1:0]	
Bk[1:0]		Row[12:0]													Column[10:0]											M[1:0]	

Notes:
1. M[1:0] is the byte address inside a 32-bit word.
2. Bk[1] = BA1, Bk[0] = BA0.

22.4.2 16-bit Memory Data Bus Width

Table 22-5. SDRAM Configuration Mapping: 2K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						Bk[1:0]		Row[10:0]											Column[7:0]								M0
					Bk[1:0]		Row[10:0]											Column[8:0]									M0
				Bk[1:0]		Row[10:0]											Column[9:0]										M0
			Bk[1:0]		Row[10:0]											Column[10:0]											M0

Table 22-6. SDRAM Configuration Mapping: 4K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Bk[1:0]		Row[11:0]												Column[7:0]								M0
				Bk[1:0]		Row[11:0]												Column[8:0]									M0
			Bk[1:0]		Row[11:0]												Column[9:0]										M0
		Bk[1:0]		Row[11:0]												Column[10:0]											M0

Table 22-7. SDRAM Configuration Mapping: 8K Rows, 256/512/1024/2048 Columns

CPU Address Line																											
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				Bk[1:0]		Row[12:0]													Column[7:0]								M0
			Bk[1:0]		Row[12:0]													Column[8:0]									M0
		Bk[1:0]		Row[12:0]													Column[9:0]										M0
	Bk[1:0]		Row[12:0]													Column[10:0]											M0

Notes:
1. M0 is the byte address inside a 16-bit half-word.
2. Bk[1] = BA1, Bk[0] = BA0.

22.5 Product Dependencies

22.5.1 SDRAM Device Initialization

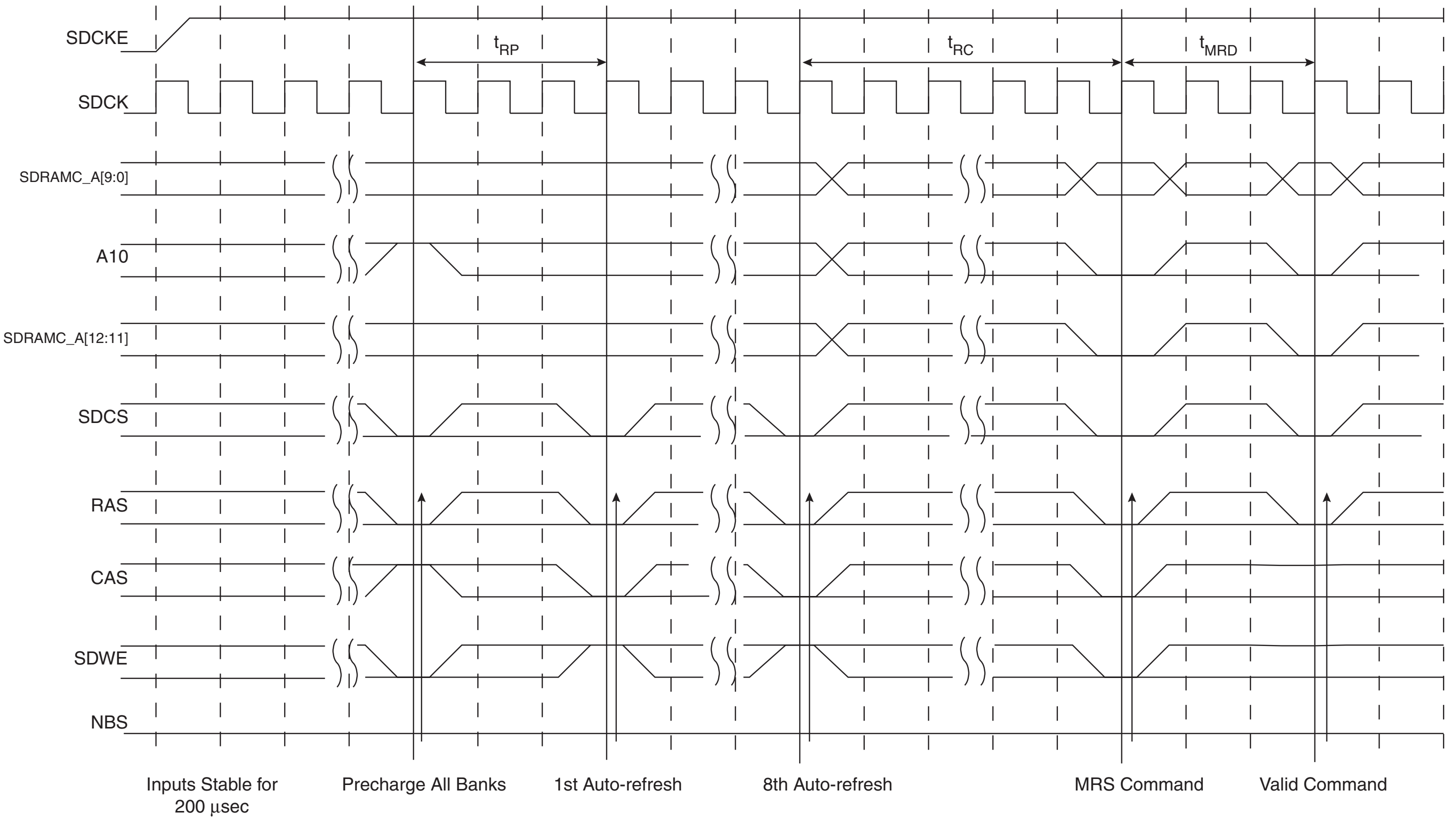
The initialization sequence is generated by software. The SDRAM devices are initialized by the following sequence:

1. SDRAM features must be set in the configuration register: asynchronous timings (TRC, TRAS, etc.), number of columns, rows, CAS latency, and the data bus width.
2. For mobile SDRAM, temperature-compensated self refresh (TCSR), drive strength (DS) and partial array self refresh (PASR) must be set in the Low Power Register.
3. The SDRAM memory type must be set in the Memory Device Register.
4. A minimum pause of 200 µs is provided to precede any signal toggle.
5. (1)A NOP command is issued to the SDRAM devices. The application must set Mode to 1 in the Mode Register and perform a write access to any SDRAM address.
6. An All Banks Precharge command is issued to the SDRAM devices. The application must set Mode to 2 in the Mode Register and perform a write access to any SDRAM address.
7. Eight auto-refresh (CBR) cycles are provided. The application must set the Mode to 4 in the Mode Register and perform a write access to any SDRAM location eight times.
8. A Mode Register set (MRS) cycle is issued to program the parameters of the SDRAM devices, in particular CAS latency and burst length. The application must set Mode to 3 in the Mode Register and perform a write access to the SDRAM. The write address must be chosen so that BA[1:0] are set to 0. For example, with a 16-bit 128 MB SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be done at the address 0x20000000.
9. For mobile SDRAM initialization, an Extended Mode Register set (EMRS) cycle is issued to program the SDRAM parameters (TCSR, PASR, DS). The application must set Mode to 5 in the Mode Register and perform a write access to the SDRAM. The write address must be chosen so that BA[1] or BA[0] are set to 1. For example, with a 16-bit 128 MB SDRAM, (12 rows, 9 columns, 4 banks) bank address the SDRAM write access should be done at the address 0x20800000 or 0x20400000.
10. The application must go into Normal Mode, setting Mode to 0 in the Mode Register and performing a write access at any location in the SDRAM.
11. Write the refresh rate into the count field in the SDRAMC Refresh Timer register. (Refresh rate = delay between refresh cycles). The SDRAM device requires a refresh every 15.625 µs or 7.81 µs. With a 100 MHz frequency, the Refresh Timer Counter Register must be set with the value 1562(15.652 µs x 100 MHz) or 781(7.81 µs x 100 MHz).

After initialization, the SDRAM devices are fully functional.

Note: 1. It is strongly recommended to respect the instructions stated in Step 5 of the initialization process in order to be certain that the subsequent commands issued by the SDRAMC will be taken into account.

Figure 22-1. SDRAM Device Initialization Sequence


SDCKE
SDCK
SDRAMC_A[9:0]
A10
SDRAMC_A[12:11]
SDCS
RAS
CAS
SDWE
NBS
t_{RP}
t_{RC}
t_{MRD}
Inputs Stable for 200 µsec
Precharge All Banks
1st Auto-refresh
8th Auto-refresh
MRS Command
Valid Command

22.5.2 I/O Lines

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

22.5.3 Interrupt

The SDRAM Controller interrupt (Refresh Error notification) is connected to the Memory Controller. This interrupt may be ORed with other System Peripheral interrupt lines and is finally provided as the System Interrupt Source (Source 1) to the AIC (Advanced Interrupt Controller).

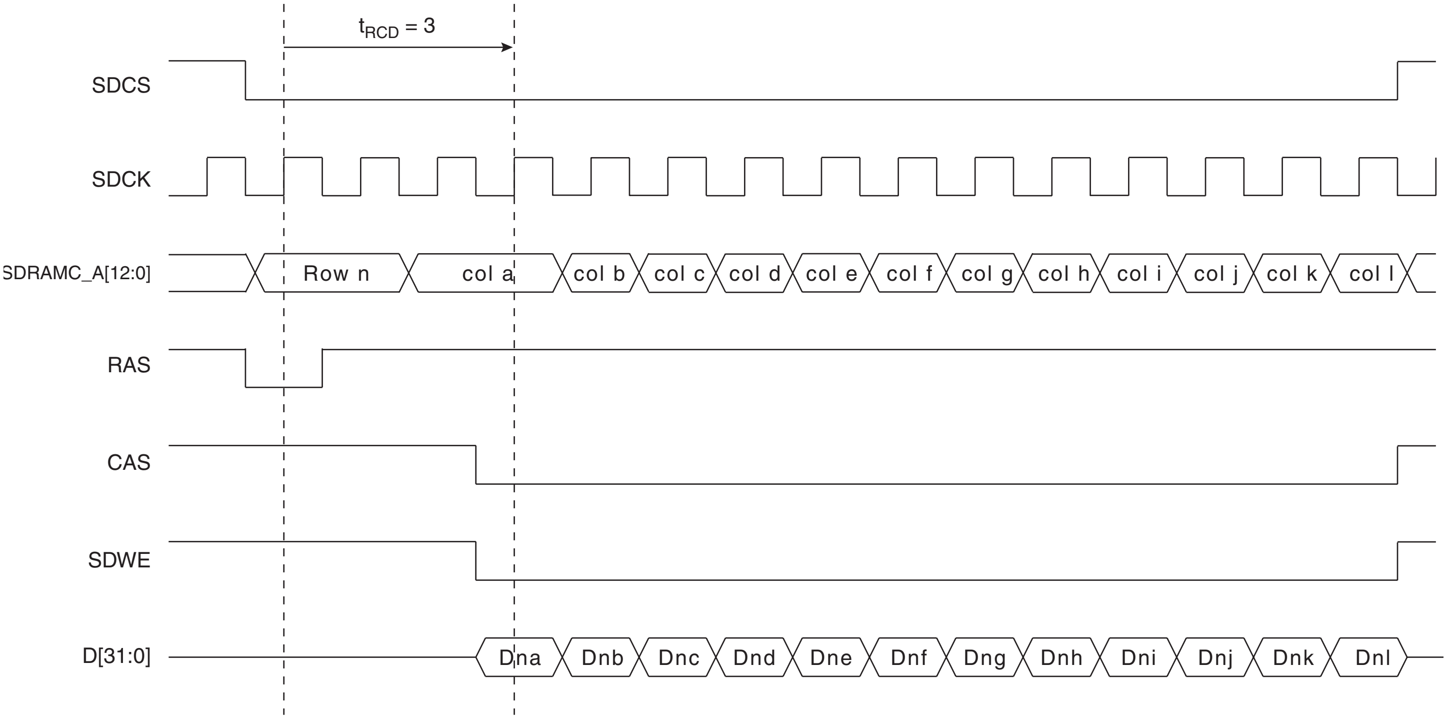
Using the SDRAM Controller interrupt requires the AIC to be programmed first.

22.6 Functional Description

22.6.1 SDRAM Controller Write Cycle

The SDRAM Controller allows burst access or single access. In both cases, the SDRAM controller keeps track of the active row in each bank, thus maximizing performance. To initiate a burst access, the SDRAM Controller uses the transfer type signal provided by the master requesting the access. If the next access is a sequential write access, writing to the SDRAM device is carried out. If the next access is a write-sequential access, but the current access is to a boundary page, or if the next access is in another row, then the SDRAM Controller generates a precharge command, activates the new row and initiates a write command. To comply with SDRAM timing parameters, additional clock cycles are inserted between precharge/active (t_{RP}) commands and active/write (t_{RCD}) commands. For definition of these timing parameters, refer to the "SDRAMC Configuration Register" on page 187. This is described in Figure 22-2 below.

Figure 22-2. Write Burst, 32-bit SDRAM Access



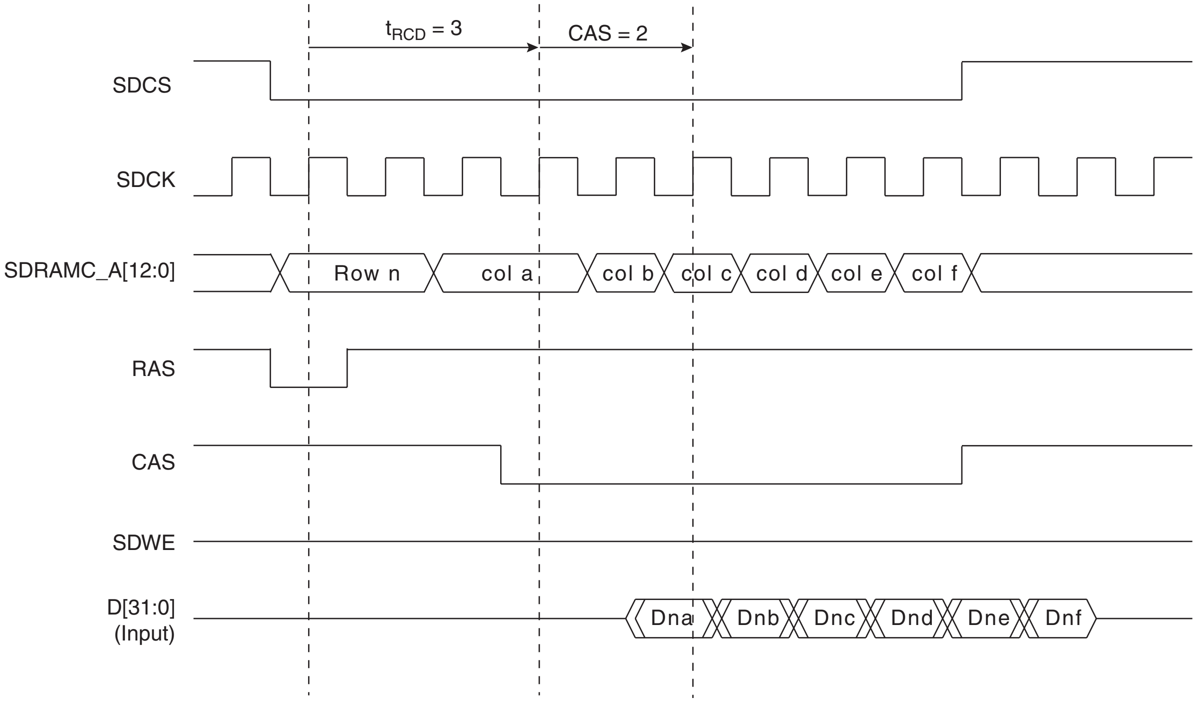
22.6.2 SDRAM Controller Read Cycle

The SDRAM Controller allows burst access, incremental burst of unspecified length or single access. In all cases, the SDRAM Controller keeps track of the active row in each bank, thus maximizing performance of the SDRAM. If row and bank addresses do not match the previous row/bank address, then the SDRAM controller automatically generates a precharge command, activates the new row and starts the read command. To comply with the SDRAM timing parameters, additional clock cycles on SDCK are inserted between precharge and active commands (t_{RP}) and between active and read command (t_{RCD}). These two parameters are set in the configuration register of the SDRAM Controller. After a read command, additional wait states are generated to comply with the CAS latency (1, 2 or 3 clock delays specified in the configuration register).

For a single access or an incremented burst of unspecified length, the SDRAM Controller anticipates the next access. While the last value of the column is returned by the SDRAM Controller on the bus, the SDRAM Controller anticipates the read to the next column and thus anticipates the CAS latency. This reduces the effect of the CAS latency on the internal bus.

For burst access of specified length (4, 8, 16 words), access is not anticipated. This case leads to the best performance. If the burst is broken (border, busy mode, etc.), the next access is handled as an incrementing burst of unspecified length.

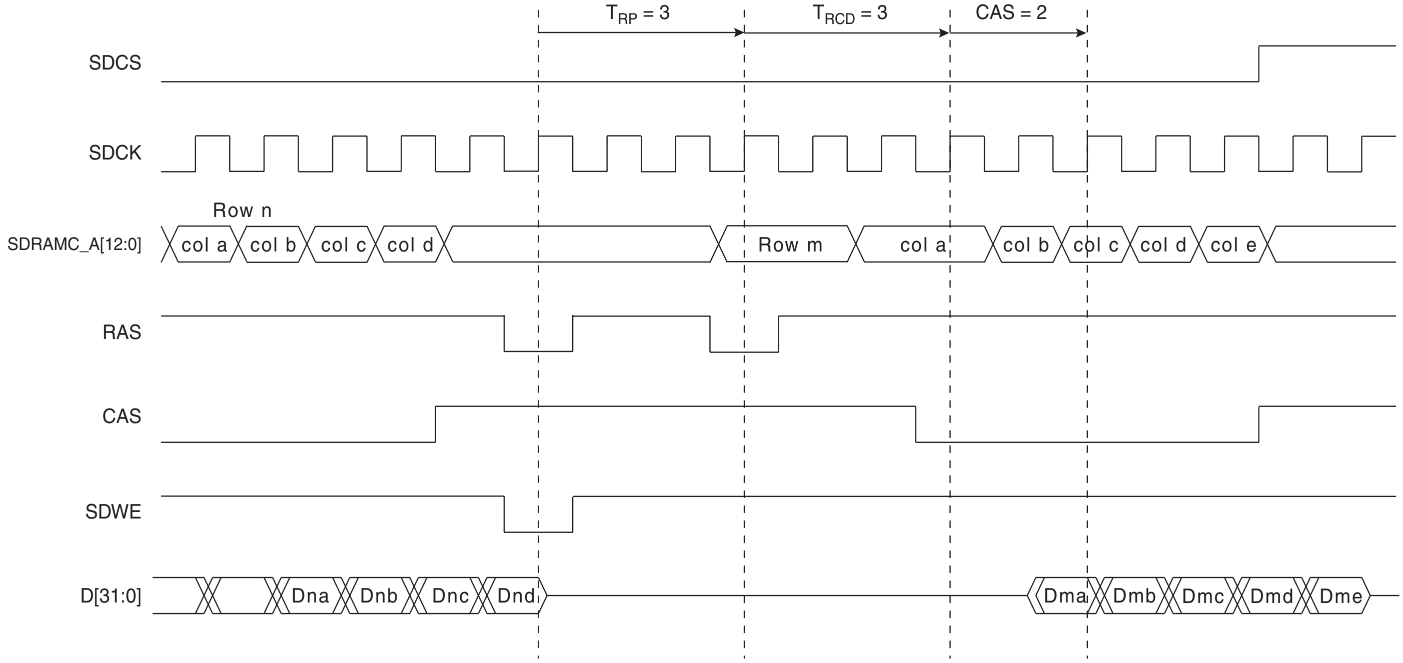
Figure 22-3. Read Burst, 32-bit SDRAM Access


t_{RCD} = 3
CAS = 2
SDCS
SDCK
SDRAMC_A[12:0]
Row n
col a
col b
col c
col d
col e
col f
RAS
CAS
SDWE
D[31:0]
(Input)
Dna
Dnb
Dnc
Dnd
Dne
Dnf

22.6.3 Border Management

When the memory row boundary has been reached, an automatic page break is inserted. In this case, the SDRAM controller generates a precharge command, activates the new row and initiates a read or write command. To comply with SDRAM timing parameters, an additional clock cycle is inserted between the precharge/active (t_{RP}) command and the active/read (t_{RCD}) command. This is described in Figure 22-4 below.

Figure 22-4. Read Burst with Boundary Row Access


$T_{RP} = 3$
$T_{RCD} = 3$
CAS = 2
SDCS
SDCK
Row n
SDRAMC_A[12:0]
col a
col b
col c
col d
Row m
col a
col b
col c
col d
col e
RAS
CAS
SDWE
D[31:0]
Dna
Dnb
Dnc
Dnd
Dma
Dmb
Dmc
Dmd
Dme

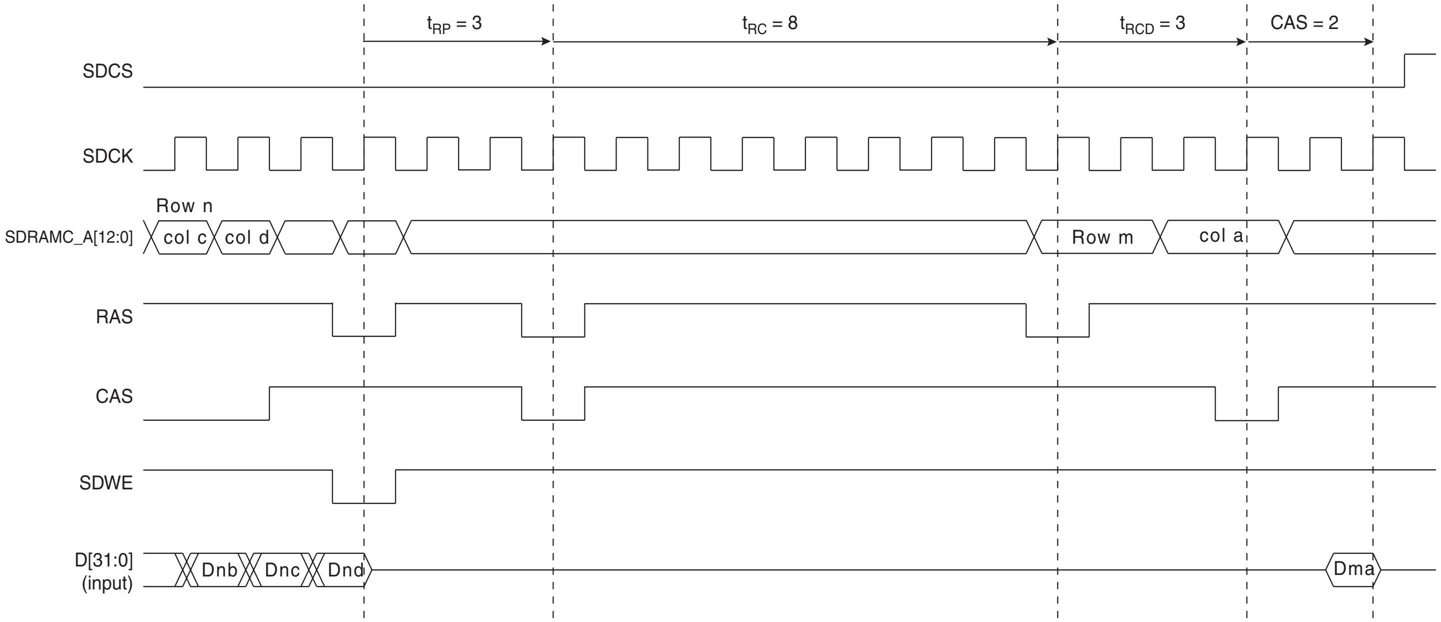
22.6.4 SDRAM Controller Refresh Cycles

An auto-refresh command is used to refresh the SDRAM device. Refresh addresses are generated internally by the SDRAM device and incremented after each auto-refresh automatically. The SDRAM Controller generates these auto-refresh commands periodically. An internal timer is loaded with the value in the register SDRAMC_TR that indicates the number of clock cycles between refresh cycles.

A refresh error interrupt is generated when the previous auto-refresh command did not perform. It is acknowledged by reading the Interrupt Status Register (SDRAMC_ISR).

When the SDRAM Controller initiates a refresh of the SDRAM device, internal memory accesses are not delayed. However, if the CPU tries to access the SDRAM, the slave indicates that the device is busy and the master is held by a wait signal. See Figure 22-5.

Figure 22-5. Refresh Cycle Followed by a Read Access



22.6.5 Power Management

Three low-power modes are available:

- Self-refresh Mode: The SDRAM executes its own Auto-refresh cycle without control of the SDRAM Controller. Current drained by the SDRAM is very low.
- Power-down Mode: Auto-refresh cycles are controlled by the SDRAM Controller. Between auto-refresh cycles, the SDRAM is in power-down. Current drained in Power-down mode is higher than in Self-refresh Mode.
- Deep Power-down Mode: (Only available with Mobile SDRAM) The SDRAM contents are lost, but the SDRAM does not drain any current.

The SDRAM Controller activates one low-power mode as soon as the SDRAM device is not selected. It is possible to delay the entry in self-refresh and power-down mode after the last access by programming a timeout value in the Low Power Register.

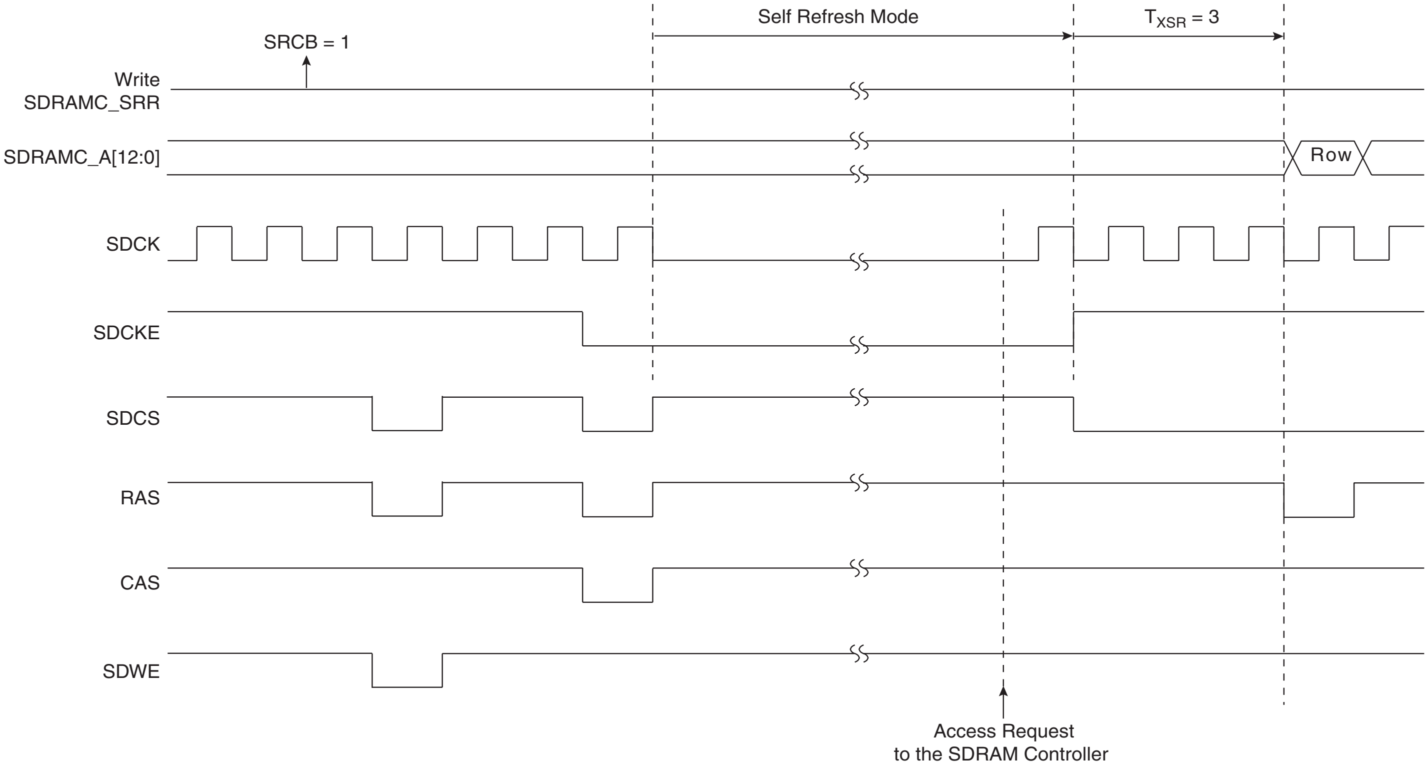
22.6.5.1 Self-refresh Mode

This mode is selected by programming the LPCB field to 1 in the SDRAMC Low Power Register. In self-refresh mode, the SDRAM device retains data without external clocking and provides its own internal clocking, thus performing its own auto-refresh cycles. All the inputs to the SDRAM device become “don’t care” except SDCKE, which remains low. As soon as the SDRAM device is selected, the SDRAM Controller provides a sequence of commands and exits self-refresh mode.

Some low-power SDRAMs (e.g., mobile SDRAM) can refresh only one quarter or a half quarter or all banks of the SDRAM array. This feature reduces the self-refresh current. To configure this feature, Temperature Compensated Self Refresh (TCSR), Partial Array Self Refresh (PASR) and Drive Strength (DS) parameters must be set in the Low Power Register and transmitted to the low-power SDRAM during initialization.

The SDRAM device must remain in self-refresh mode for a minimum period of t_{RAS} and may remain in self-refresh mode for an indefinite period. This is described in Figure 22-6.

Figure 22-6. Self-refresh Mode Behavior

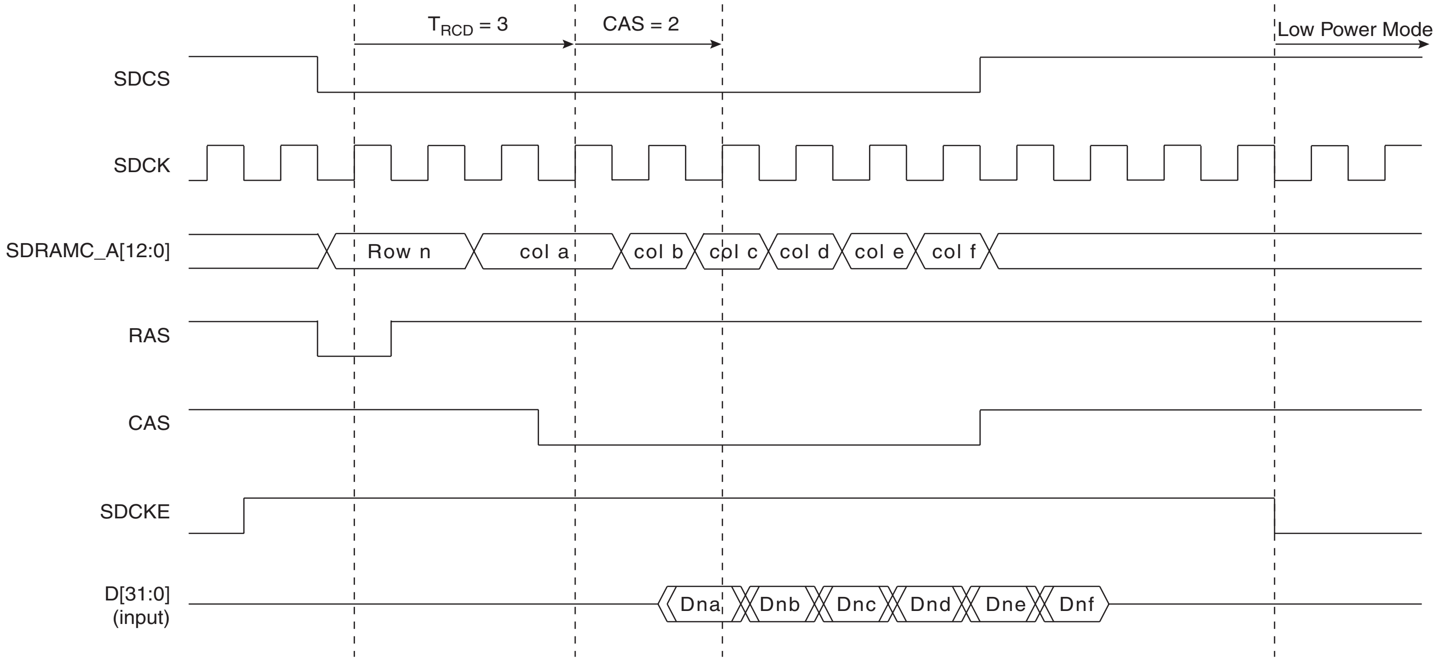

Self Refresh Mode
$T_{XSR} = 3$
SRCB = 1
Write SDRAMC_SRR
SDRAMC_A[12:0]
Row
SDCK
SDCKE
SDCS
RAS
CAS
SDWE
Access Request to the SDRAM Controller

22.6.5.2 Low-power Mode

This mode is selected by programming the LPCB field to 2 in the SDRAMC Low Power Register. Power consumption is greater than in self-refresh mode. All the input and output buffers of the SDRAM device are deactivated except SDCKE, which remains low. In contrast to self-refresh mode, the SDRAM device cannot remain in low-power mode longer than the refresh period (64 ms for a whole device refresh operation). As no auto-refresh operations are performed by the SDRAM itself, the SDRAM Controller carries out the refresh operation. The exit procedure is faster than in self-refresh mode.

This is described in Figure 22-7.

Figure 22-7. Low-power Mode Behavior


T_{RCD} = 3
CAS = 2
Low Power Mode
SDCS
SDCK
SDRAMC_A[12:0]
Row n
col a
col b
col c
col d
col e
col f
RAS
CAS
SDCKE
D[31:0]
(input)
Dna
Dnb
Dnc
Dnd
Dne
Dnf

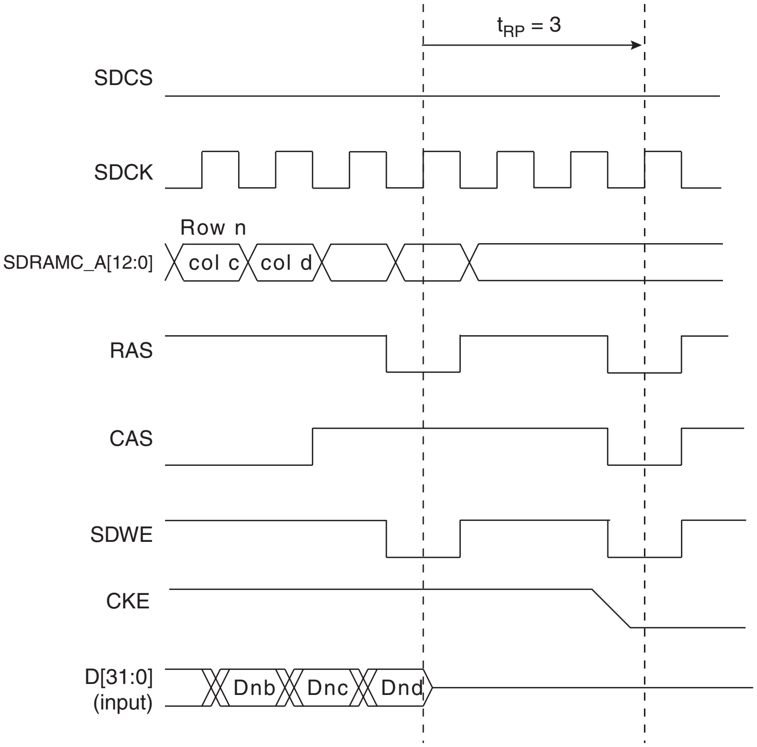
22.6.5.3 Deep Power-down Mode

This mode is selected by programming the LPCB field to 3 in the SDRAMC Low Power Register. When this mode is activated, all internal voltage generators inside the SDRAM are stopped and all data is lost.

When this mode is enabled, the application must not access to the SDRAM until a new initialization sequence is done (See "SDRAM Device Initialization" on page 176).

This is described in Figure 22-8.

Figure 22-8. Deep Power-down Mode Behavior


$t_{RP} = 3$
SDCS
SDCK
Row n
SDRAMC_A[12:0]
col c
col d
RAS
CAS
SDWE
CKE
D[31:0]
(input)
Dnb
Dnc
Dnd

22.7 SDRAM Controller User Interface

Table 22-8. SDRAM Controller Memory Map

Offset	Register	Name	Access	Reset State
0x00	SDRAMC Mode Register	SDRAMC_MR	Read/Write	0x00000000
0x04	SDRAMC Refresh Timer Register	SDRAMC_TR	Read/Write	0x00000000
0x08	SDRAMC Configuration Register	SDRAMC_CR	Read/Write	0x852372C0
0x0C	SDRAMC High Speed Register	SDRAMC_HSR	Read/Write	0x00
0x10	SDRAMC Low Power Register	SDRAMC_LPR	Read/Write	0x0
0x14	SDRAMC Interrupt Enable Register	SDRAMC_IER	Write-only	–
0x18	SDRAMC Interrupt Disable Register	SDRAMC_IDR	Write-only	–
0x1C	SDRAMC Interrupt Mask Register	SDRAMC_IMR	Read-only	0x0
0x20	SDRAMC Interrupt Status Register	SDRAMC_ISR	Read-only	0x0
0x24	SDRAMC Memory Device Register	SDRAMC_MDR	Read	0x0
0x28 - 0xFC	Reserved	–	–	–

22.7.1 SDRAMC Mode Register

Register Name: SDRAMC_MR

Access Type: Read/Write

Reset Value: 0x00000000

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
–	–	–	–	–	MODE		

- **MODE: SDRAMC Command Mode**

This field defines the command issued by the SDRAM Controller when the SDRAM device is accessed.

Table 22-9.

MODE			Description
0	0	0	Normal mode. Any access to the SDRAM is decoded normally.
0	0	1	The SDRAM Controller issues a NOP command when the SDRAM device is accessed regardless of the cycle.
0	1	0	The SDRAM Controller issues an “All Banks Precharge” command when the SDRAM device is accessed regardless of the cycle.
0	1	1	The SDRAM Controller issues a “Load Mode Register” command when the SDRAM device is accessed regardless of the cycle. The address offset with respect to the SDRAM device base address is used to program the Mode Register. For instance, when this mode is activated, an access to the “SDRAM_Base + offset” address generates a “Load Mode Register” command with the value “offset” written to the SDRAM device Mode Register.
1	0	0	The SDRAM Controller issues an “Auto-Refresh” Command when the SDRAM device is accessed regardless of the cycle. Previously, an “All Banks Precharge” command must be issued.
1	0	1	The SDRAM Controller issues an extended load mode register command when the SDRAM device is accessed regardless of the cycle. The address offset with respect to the SDRAM device base address is used to program the Mode Register. For instance, when this mode is activated, an access to the “SDRAM_Base + offset” address generates an “Extended Load Mode Register” command with the value “offset” written to the SDRAM device Mode Register.
1	1	0	Deep power-down mode. Enters deep power-down mode.

22.7.2 SDRAMC Refresh Timer Register

Register Name: SDRAMC_TR

Access Type: Read/Write

Reset Value: 0x00000000

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

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

15	14	13	12	11	10	9	8
–	–	–	–	COUNT			

7	6	5	4	3	2	1	0
COUNT							

- **COUNT: SDRAMC Refresh Timer Count**

This 12-bit field is loaded into a timer that generates the refresh pulse. Each time the refresh pulse is generated, a refresh burst is initiated. The value to be loaded depends on the SDRAMC clock frequency (MCK: Master Clock), the refresh rate of the SDRAM device and the refresh burst length where 15.6 µs per row is a typical value for a burst of length one.

To refresh the SDRAM device, this 12-bit field must be written. If this condition is not satisfied, no refresh command is issued and no refresh of the SDRAM device is carried out.

22.7.3 SDRAMC Configuration Register

Register Name: SDRAMC_CR

Access Type: Read/Write

Reset Value: 0x852372C0

31	30	29	28	27	26	25	24
TXSR				TRAS			

23	22	21	20	19	18	17	16
TRCD				TRP			

15	14	13	12	11	10	9	8
TRC				TWR			

7	6	5	4	3	2	1	0
DBW	CAS		NB	NR		NC	

- **NC: Number of Column Bits**

Reset value is 8 column bits.

NC		Column Bits
0	0	8
0	1	9
1	0	10
1	1	11

- **NR: Number of Row Bits**

Reset value is 11 row bits.

NR		Row Bits
0	0	11
0	1	12
1	0	13
1	1	Reserved

- **NB: Number of Banks**

Reset value is two banks.

NB	Number of Banks
0	2
1	4

- **CAS: CAS Latency**

Reset value is two cycles.

In the SDRAMC, only a CAS latency of one, two and three cycles are managed. In any case, another value must be programmed.

CAS		CAS Latency (Cycles)
0	0	Reserved
0	1	1
1	0	2
1	1	3

- **DBW: Data Bus Width**

Reset value is 16 bits

0: Data bus width is 32 bits.

1: Data bus width is 16 bits.

- **TWR: Write Recovery Delay**

Reset value is two cycles.

This field defines the Write Recovery Time in number of cycles. Number of cycles is between 0 and 15.

- **TRC: Row Cycle Delay**

Reset value is seven cycles.

This field defines the delay between a Refresh and an Activate Command in number of cycles. Number of cycles is between 0 and 15.

- **TRP: Row Precharge Delay**

Reset value is three cycles.

This field defines the delay between a Precharge Command and another Command in number of cycles. Number of cycles is between 0 and 15.

- **TRCD: Row to Column Delay**

Reset value is two cycles.

This field defines the delay between an Activate Command and a Read/Write Command in number of cycles. Number of cycles is between 0 and 15.

- **TRAS: Active to Precharge Delay**

Reset value is five cycles.

This field defines the delay between an Activate Command and a Precharge Command in number of cycles. Number of cycles is between 0 and 15.

- **TXSR: Exit Self Refresh to Active Delay**

Reset value is eight cycles.

This field defines the delay between SCKE set high and an Activate Command in number of cycles. Number of cycles is between 0 and 15.

22.7.4 SDRAMC High Speed Register

Register Name:	SDRAMC_HSR
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**
–	–	–	–	–	–	–	DA

- **DA: Decode Cycle Enable**

A decode cycle can be added on the addresses as soon as a non-sequential access is performed on the AHB bus.

The addition of the decode cycle allows the SDRAMC to gain time to access the SDRAM memory.

0: Decode cycle is disabled.

1: Decode cycle is enabled.

22.7.5 SDRAMC Low Power Register

Register Name: SDRAMC_LPR

Access Type: Read/Write

Reset Value: 0x0

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

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

15	14	13	12	11	10	9	8
–	–	TIMEOUT		DS		TCSR	

7	6	5	4	3	2	1	0
–	PASR			–	–	LPCB	

- **LPCB: Low-power Configuration Bits**

00	Low Power Feature is inhibited: no Power-down, Self-refresh or Deep Power-down command is issued to the SDRAM device.
01	The SDRAM Controller issues a Self-refresh command to the SDRAM device, the SDCLK clock is deactivated and the SDCKE signal is set low. The SDRAM device leaves the Self Refresh Mode when accessed and enters it after the access.
10	The SDRAM Controller issues a Power-down Command to the SDRAM device after each access, the SDCKE signal is set to low. The SDRAM device leaves the Power-down Mode when accessed and enters it after the access.
11	The SDRAM Controller issues a Deep Power-down command to the SDRAM device. This mode is unique to low-power SDRAM.

- **PASR: Partial Array Self-refresh (only for low-power SDRAM)**

PASR parameter is transmitted to the SDRAM during initialization to specify whether only one quarter, one half or all banks of the SDRAM array are enabled. Disabled banks are not refreshed in self-refresh mode. This parameter must be set according to the SDRAM device specification.

- **TCSR: Temperature Compensated Self-Refresh (only for low-power SDRAM)**

TCSR parameter is transmitted to the SDRAM during initialization to set the refresh interval during self-refresh mode depending on the temperature of the low-power SDRAM. This parameter must be set according to the SDRAM device specification.

- **DS: Drive Strength (only for low-power SDRAM)**

DS parameter is transmitted to the SDRAM during initialization to select the SDRAM strength of data output. This parameter must be set according to the SDRAM device specification.

- **TIMEOUT: Time to define when low-power mode is enabled**

00	The SDRAM controller activates the SDRAM low-power mode immediately after the end of the last transfer.
01	The SDRAM controller activates the SDRAM low-power mode 64 clock cycles after the end of the last transfer.
10	The SDRAM controller activates the SDRAM low-power mode 128 clock cycles after the end of the last transfer.
11	Reserved.

22.7.6 SDRAMC Interrupt Enable Register

Register Name:	SDRAMC_IER
Access Type:	Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	RES

- **RES: Refresh Error Status**

0: No effect.

1: Enables the refresh error interrupt.

22.7.7 SDRAMC Interrupt Disable Register

Register Name:	SDRAMC_IDR
Access Type:	Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	RES

- **RES: Refresh Error Status**

0: No effect.

1: Disables the refresh error interrupt.

22.7.8 SDRAMC Interrupt Mask Register

Register Name: SDRAMC_IMR

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	RES

- **RES: Refresh Error Status**

0: The refresh error interrupt is disabled.

1: The refresh error interrupt is enabled.

22.7.9 SDRAMC Interrupt Status Register

Register Name: SDRAMC_ISR

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	RES

- **RES: Refresh Error Status**

0: No refresh error has been detected since the register was last read.

1: A refresh error has been detected since the register was last read.

22.7.10 SDRAMC Memory Device Register

Register Name: SDRAMC_MDR

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
–	–	–	–	–	–	MD	

- **MD: Memory Device Type**

00	SDRAM
01	Low-power SDRAM
10	Reserved
11	Reserved.

23. Peripheral DMA Controller (PDC)

23.1 Description

The Peripheral DMA Controller (PDC) transfers data between on-chip serial peripherals and the on- and/or off-chip memories. The link between the PDC and a serial peripheral is operated by the AHB to ABP bridge.

The PDC contains 22 channels. The full-duplex peripherals feature 19 mono directional channels used in pairs (transmit only or receive only) except the ADC Controller uses only one RX channel. The half-duplex peripherals feature 3 bi-directional channels.

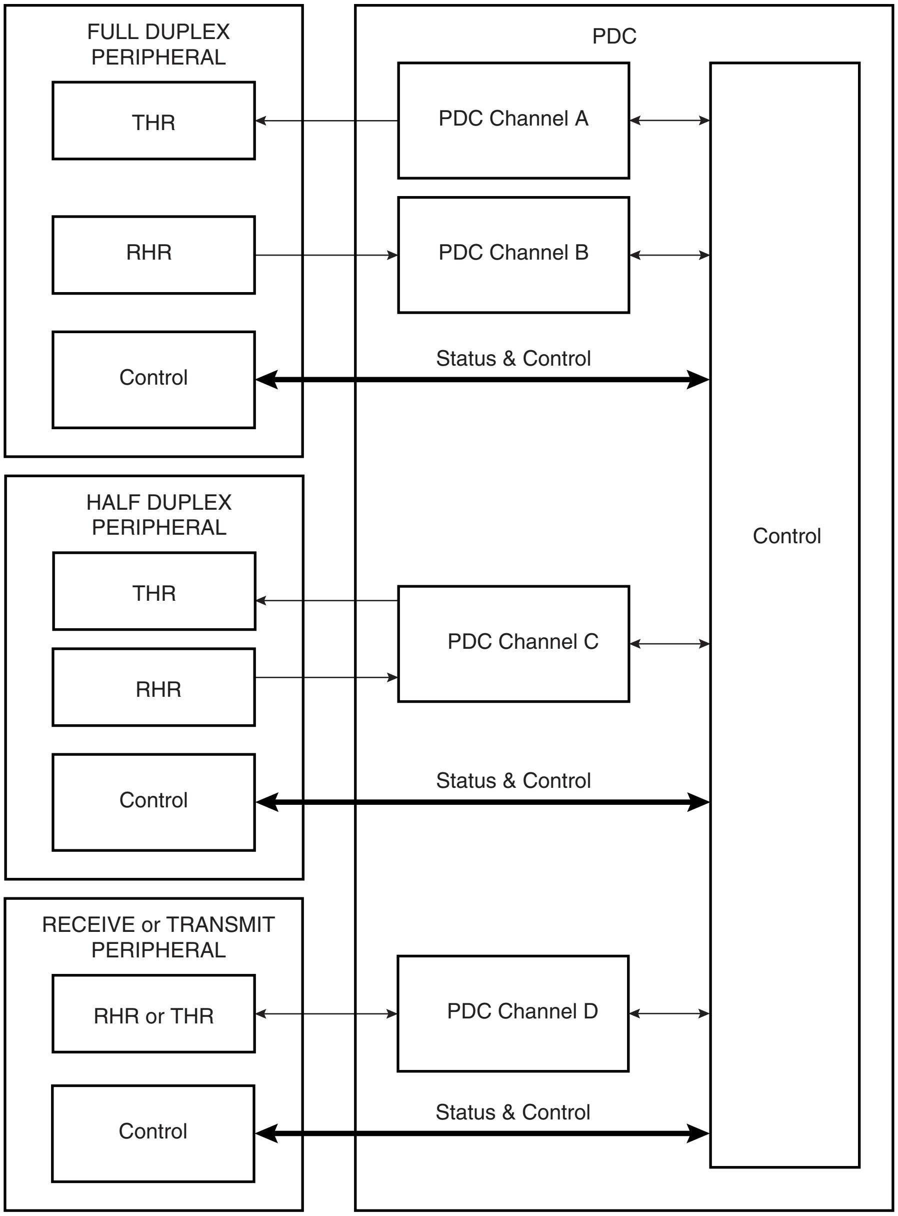
The user interface of each PDC channel is integrated into the user interface of the peripheral it serves. The user interface of mono directional channels (receive only or transmit only), contains two 32-bit memory pointers and two 16-bit counters, one set (pointer, counter) for current transfer and one set (pointer, counter) for next transfer. The bi-directional channel user interface contains four 32-bit memory pointers and four 16-bit counters. Each set (pointer, counter) is used by current transmit, next transmit, current receive and next receive.

Using the PDC removes processor overhead by reducing its intervention during the transfer. This significantly reduces the number of clock cycles required for a data transfer, which improves microcontroller performance.

To launch a transfer, the peripheral triggers its associated PDC channels by using transmit and receive signals. When the programmed data is transferred, an end of transfer interrupt is generated by the peripheral itself.

23.2 Block Diagram

Figure 23-1. Block Diagram


FULL DUPLEX PERIPHERAL
THR
RHR
Control
PDC
PDC Channel A
PDC Channel B
Status & Control
Control
HALF DUPLEX PERIPHERAL
THR
RHR
Control
PDC Channel C
Status & Control
RECEIVE or TRANSMIT PERIPHERAL
RHR or THR
Control
PDC Channel D
Status & Control

23.3 Functional Description

23.3.1 Configuration

The PDC channel user interface enables the user to configure and control data transfers for each channel. The user interface of each PDC channel is integrated into the associated peripheral user interface.

The user interface of a serial peripheral, whether it is full or half duplex, contains four 32-bit pointers (RPR, RNPR, TPR, TNPR) and four 16-bit counter registers (RCR, RNCR, TCR, TNCR). However, the transmit and receive parts of each type are programmed differently: the transmit and receive parts of a full duplex peripheral can be programmed at the same time, whereas only one part (transmit or receive) of a half duplex peripheral can be programmed at a time.

32-bit pointers define the access location in memory for current and next transfer, whether it is for read (transmit) or write (receive). 16-bit counters define the size of current and next transfers. It is possible, at any moment, to read the number of transfers left for each channel.

The PDC has dedicated status registers which indicate if the transfer is enabled or disabled for each channel. The status for each channel is located in the associated peripheral status register. Transfers can be enabled and/or disabled by setting TXTEN/TXTDIS and RXTEN/RXTDIS in the peripheral's Transfer Control Register.

At the end of a transfer, the PDC channel sends status flags to its associated peripheral. These flags are visible in the peripheral status register (ENDRX, ENDTX, RXBUFF, and TXBUFE). Refer to Section 23.3.3 and to the associated peripheral user interface.

23.3.2 Memory Pointers

Each full duplex peripheral is connected to the PDC by a receive channel and a transmit channel. Both channels have 32-bit memory pointers that point respectively to a receive area and to a transmit area in on- and/or off-chip memory.

Each half duplex peripheral is connected to the PDC by a bidirectional channel. This channel has two 32-bit memory pointers, one for current transfer and the other for next transfer. These pointers point to transmit or receive data depending on the operating mode of the peripheral.

Depending on the type of transfer (byte, half-word or word), the memory pointer is incremented respectively by 1, 2 or 4 bytes.

If a memory pointer address changes in the middle of a transfer, the PDC channel continues operating using the new address.

23.3.3 Transfer Counters

Each channel has two 16-bit counters, one for current transfer and the other one for next transfer. These counters define the size of data to be transferred by the channel. The current transfer counter is decremented first as the data addressed by current memory pointer starts to be transferred. When the current transfer counter reaches zero, the channel checks its next transfer counter. If the value of next counter is zero, the channel stops transferring data and sets the appropriate flag. But if the next counter value is greater then zero, the values of the next pointer/next counter are copied into the current pointer/current counter and the channel resumes the transfer whereas next pointer/next counter get zero/zero as values. At the end of this transfer the PDC channel sets the appropriate flags in the Peripheral Status Register.

The following list gives an overview of how status register flags behave depending on the counters' values:

- ENDRX flag is set when the PERIPH_RCR register reaches zero.
- RXBUFF flag is set when both PERIPH_RCR and PERIPH_RNCR reach zero.
- ENDTX flag is set when the PERIPH_TCR register reaches zero.
- TXBUFE flag is set when both PERIPH_TCR and PERIPH_TNCR reach zero.

These status flags are described in the Peripheral Status Register.

23.3.4 Data Transfers

The serial peripheral triggers its associated PDC channels' transfers using transmit enable (TXEN) and receive enable (RXEN) flags in the transfer control register integrated in the peripheral's user interface.

When the peripheral receives an external data, it sends a Receive Ready signal to its PDC receive channel which then requests access to the Matrix. When access is granted, the PDC receive channel starts reading the peripheral Receive Holding Register (RHR). The read data are stored in an internal buffer and then written to memory.

When the peripheral is about to send data, it sends a Transmit Ready to its PDC transmit channel which then requests access to the Matrix. When access is granted, the PDC transmit channel reads data from memory and puts them to Transmit Holding Register (THR) of its associated peripheral. The same peripheral sends data according to its mechanism.

23.3.5 PDC Flags and Peripheral Status Register

Each peripheral connected to the PDC sends out receive ready and transmit ready flags and the PDC sends back flags to the peripheral. All these flags are only visible in the Peripheral Status Register.

Depending on the type of peripheral, half or full duplex, the flags belong to either one single channel or two different channels.

23.3.5.1 Receive Transfer End

This flag is set when PERIPH_RCR register reaches zero and the last data has been transferred to memory.

It is reset by writing a non zero value in PERIPH_RCR or PERIPH_RNCR.

23.3.5.2 Transmit Transfer End

This flag is set when PERIPH_TCR register reaches zero and the last data has been written into peripheral THR.

It is reset by writing a non zero value in PERIPH_TCR or PERIPH_TNCR.

23.3.5.3 Receive Buffer Full

This flag is set when PERIPH_RCR register reaches zero with PERIPH_RNCR also set to zero and the last data has been transferred to memory.

It is reset by writing a non zero value in PERIPH_TCR or PERIPH_TNCR.

23.3.5.4 Transmit Buffer Empty

This flag is set when PERIPH_TCR register reaches zero with PERIPH_TNCR also set to zero and the last data has been written into peripheral THR.

It is reset by writing a non zero value in PERIPH_TCR or PERIPH_TNCR.

23.4 Peripheral DMA Controller (PDC) User Interface

Table 23-1. Memory Map

Offset	Register	Name	Access	Reset State
0x100	Receive Pointer Register	PERIPH[1]_RPR	Read/Write	0
0x104	Receive Counter Register	PERIPH_RCR	Read/Write	0
0x108	Transmit Pointer Register	PERIPH_TPR	Read/Write	0
0x10C	Transmit Counter Register	PERIPH_TCR	Read/Write	0
0x110	Receive Next Pointer Register	PERIPH_RNPR	Read/Write	0
0x114	Receive Next Counter Register	PERIPH_RNCR	Read/Write	0
0x118	Transmit Next Pointer Register	PERIPH_TNPR	Read/Write	0
0x11C	Transmit Next Counter Register	PERIPH_TNCR	Read/Write	0
0x120	Transfer Control Register	PERIPH_PTCR	Write	0
0x124	Transfer Status Register	PERIPH_PTSR	Read	0

Note: 1. PERIPH: Ten registers are mapped in the peripheral memory space at the same offset. These can be defined by the user according to the function and the peripheral desired (DBGU, USART, SPI, etc.)

23.4.1 Receive Pointer Register

Register Name: PERIPH_RPR

Access Type: Read/Write

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

- **RXPTR: Receive Pointer Register**

RXPTR must be set to receive buffer address.

When a half duplex peripheral is connected to the PDC, RXPTR = TXPTR.

23.4.2 Receive Counter Register

Register Name: PERIPH_RCR

Access Type: Read/Write

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

- **RXCTR: Receive Counter Register**

RXCTR must be set to receive buffer size.

When a half duplex peripheral is connected to the PDC, RXCTR = TXCTR.

0 = Stops peripheral data transfer to the receiver

1 - 65535 = Starts peripheral data transfer if corresponding channel is active

23.4.3 Transmit Pointer Register

Register Name: PERIPH_TPR

Access Type: Read/Write

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

- **TXPTR: Transmit Counter Register**

TXPTR must be set to transmit buffer address.

When a half duplex peripheral is connected to the PDC, RXPTR = TXPTR.

23.4.4 Transmit Counter Register

Register Name: PERIPH_TCR

Access Type: Read/Write

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

- **TXCTR: Transmit Counter Register**

TXCTR must be set to transmit buffer size.

When a half duplex peripheral is connected to the PDC, RXCTR = TXCTR.

0 = Stops peripheral data transfer to the transmitter

1- 65535 = Starts peripheral data transfer if corresponding channel is active

23.4.5 Receive Next Pointer Register

Register Name: PERIPH_RNPR

Access Type: Read/Write

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

- **RXNPTR: Receive Next Pointer**

RXNPTR contains next receive buffer address.

When a half duplex peripheral is connected to the PDC, RXNPTR = TXNPTR.

23.4.6 Receive Next Counter Register

Register Name: PERIPH_RNCR

Access Type: Read/Write

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

- **RXNCTR: Receive Next Counter**

RXNCTR contains next receive buffer size.

When a half duplex peripheral is connected to the PDC, RXNCTR = TXNCTR.

23.4.7 Transmit Next Pointer Register

Register Name: PERIPH_TNPR

Access Type: Read/Write

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

- **TXNPTR: Transmit Next Pointer**

TXNPTR contains next transmit buffer address.

When a half duplex peripheral is connected to the PDC, RXNPTR = TXNPTR.

23.4.8 Transmit Next Counter Register

Register Name: PERIPH_TNCR

Access Type: Read/Write

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

- **TXNCTR: Transmit Counter Next**

TXNCTR contains next transmit buffer size.

When a half duplex peripheral is connected to the PDC, RXNCTR = TXNCTR.

23.4.9 Transfer Control Register

Register Name: PERIPH_PTCR

Access Type: 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**
–	–	–	–	–	–	TXTDIS	TXTEN
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	RXTDIS	RXTEN

- **RXTEN: Receiver Transfer Enable**

0 = No effect.

1 = Enables PDC receiver channel requests if RXTDIS is not set.

When a half duplex peripheral is connected to the PDC, enabling the receiver channel requests automatically disables the transmitter channel requests. It is forbidden to set both TXTEN and RXTEN for a half duplex peripheral.

- **RXTDIS: Receiver Transfer Disable**

0 = No effect.

1 = Disables the PDC receiver channel requests.

When a half duplex peripheral is connected to the PDC, disabling the receiver channel requests also disables the transmitter channel requests.

- **TXTEN: Transmitter Transfer Enable**

0 = No effect.

1 = Enables the PDC transmitter channel requests.

When a half duplex peripheral is connected to the PDC, it enables the transmitter channel requests only if RXTEN is not set. It is forbidden to set both TXTEN and RXTEN for a half duplex peripheral.

- **TXTDIS: Transmitter Transfer Disable**

0 = No effect.

1 = Disables the PDC transmitter channel requests.

When a half duplex peripheral is connected to the PDC, disabling the transmitter channel requests disables the receiver channel requests.

23.4.10 Transfer Status Register

Register Name: PERIPH_PTSR

Access Type: Read

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

- **RXTEN: Receiver Transfer Enable**

0 = PDC Receiver channel requests are disabled.

1 = PDC Receiver channel requests are enabled.

- **TXTEN: Transmitter Transfer Enable**

0 = PDC Transmitter channel requests are disabled.

1 = PDC Transmitter channel requests are enabled.

24. Advanced Power Management Controller

24.1 Clock Generator

24.1.1 Description

The Clock Generator is made up of 2 PLL, a Main Oscillator, as well as an RC Oscillator and a 32,768 Hz low-power Oscillator.

It provides the following clocks:

- SLCK, the Slow Clock, which is the only permanent clock within the system
- MAINCK is the output of the Main Oscillator

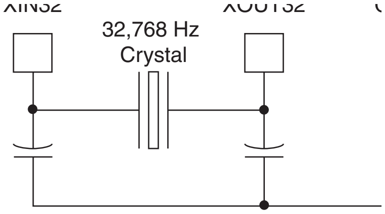
The Clock Generator User Interface is embedded within the Power Management Controller one and is described in Section 24.2.10. However, the Clock Generator registers are named CKGR_.

- PLLACK is the output of the Divider and PLL A block
- PLLBCK is the output of the Divider and PLL B block

24.1.2 Slow Clock Crystal Oscillator

The Clock Generator integrates a 32,768 Hz low-power oscillator. The XIN32 and XOUT32 pins must be connected to a 32,768 Hz crystal. Two external capacitors must be wired as shown in Figure 24-1.

Figure 24-1. Typical Slow Clock Crystal Oscillator Connection



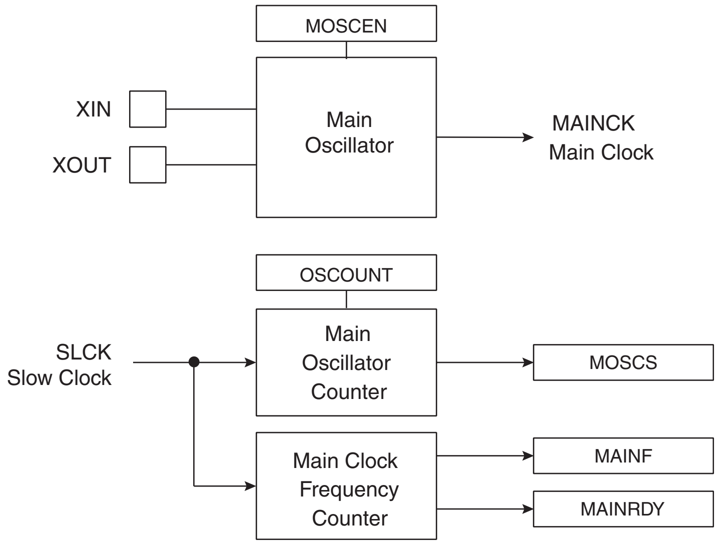
24.1.3 Slow Clock RC Oscillator

The user has to take into account the possible drifts of the RC Oscillator. More details are given in the section “DC Characteristics” of the product datasheet.

24.1.4 Main Oscillator

Figure 24-2 shows the Main Oscillator block diagram.

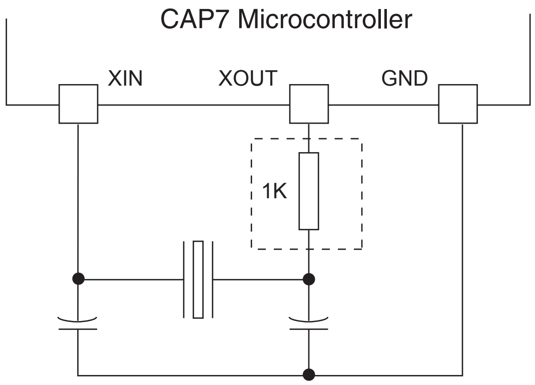
Figure 24-2. Main Oscillator Block Diagram


MOSCEN
XIN
XOUT
Main Oscillator
MAINCK
Main Clock
OSCOUNT
SLCK
Slow Clock
Main Oscillator Counter
MOSCS
Main Clock Frequency Counter
MAINF
MAINRDY

24.1.4.1 *Main Oscillator Connections*

The Clock Generator integrates a Main Oscillator that is designed for a 8 to 16 MHz fundamental crystal. The typical crystal connection is illustrated in Figure 24-3. For further details on the electrical characteristics of the Main Oscillator, see the section "DC Characteristics" of the product datasheet.

Figure 24-3. Typical Crystal Connection


CAP7 Microcontroller
XIN
XOUT
GND
1K

24.1.4.2 *Main Oscillator Startup Time*

The startup time of the Main Oscillator is given in the DC Characteristics section of the product datasheet. The startup time depends on the crystal frequency and decreases when the frequency rises.

24.1.4.3 *Main Oscillator Control*

To minimize the power required to start up the system, the main oscillator is disabled after reset and slow clock is selected.

The software enables or disables the main oscillator so as to reduce power consumption by clearing the MOSCEN bit in the Main Oscillator Register (CKGR_MOR).

When disabling the main oscillator by clearing the MOSCEN bit in CKGR_MOR, the MOSCS bit in PMC_SR is automatically cleared, indicating the main clock is off.

When enabling the main oscillator, the user must initiate the main oscillator counter with a value corresponding to the startup time of the oscillator. This startup time depends on the crystal frequency connected to the main oscillator.

When the MOSCEN bit and the OSCOUNT are written in CKGR_MOR to enable the main oscillator, the MOSCS bit in PMC_SR (Status Register) is cleared and the counter starts counting down on the slow clock divided by 8 from the OSCOUNT value. Since the OSCOUNT value is coded with 8 bits, the maximum startup time is about 62 ms.

When the counter reaches 0, the MOSCS bit is set, indicating that the main clock is valid. Setting the MOSCS bit in PMC_IMR can trigger an interrupt to the processor.

24.1.4.4 Main Clock Frequency Counter

The Main Oscillator features a Main Clock frequency counter that provides the quartz frequency connected to the Main Oscillator. Generally, this value is known by the system designer; however, it is useful for the boot program to configure the device with the correct clock speed, independently of the application.

The Main Clock frequency counter starts incrementing at the Main Clock speed after the next rising edge of the Slow Clock as soon as the Main Oscillator is stable, i.e., as soon as the MOSCS bit is set. Then, at the 16th falling edge of Slow Clock, the MAINRDY bit in CKGR_MCFR (Main Clock Frequency Register) is set and the counter stops counting. Its value can be read in the MAINF field of CKGR_MCFR and gives the number of Main Clock cycles during 16 periods of Slow Clock, so that the frequency of the crystal connected on the Main Oscillator can be determined.

24.1.4.5 Main Oscillator Bypass

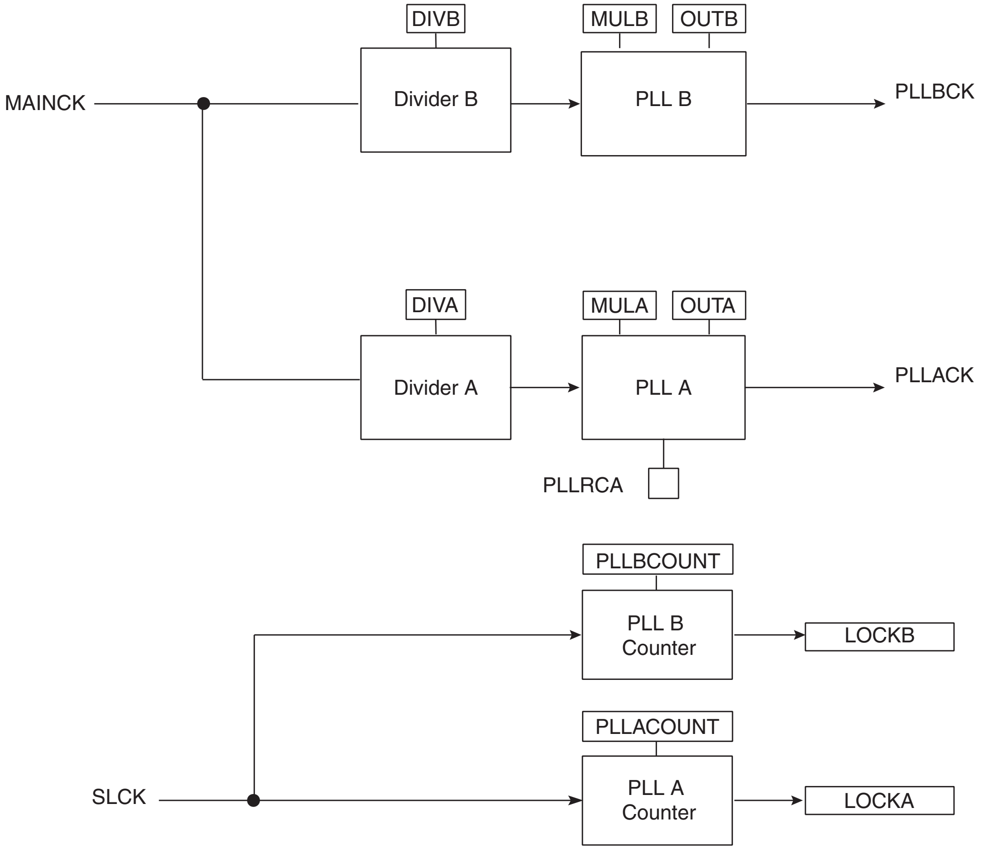
The user can input a clock on the device instead of connecting a crystal. In this case, the user has to provide the external clock signal on the XIN pin. The input characteristics of the XIN pin under these conditions are given in the product electrical characteristics section. The programmer has to be sure to set the OSCBYPASS bit to 1 and the MOSCEN bit to 0 in the Main OSC register (CKGR_MOR) for the external clock to operate properly.

24.1.5 Divider and PLL Block

The PLL embeds an input divider to increase the accuracy of the resulting clock signals. However, the user must respect the PLL minimum input frequency when programming the divider.

Figure 24-4 shows the block diagram of the divider and PLL blocks.

Figure 24-4. Divider and PLL Block Diagram

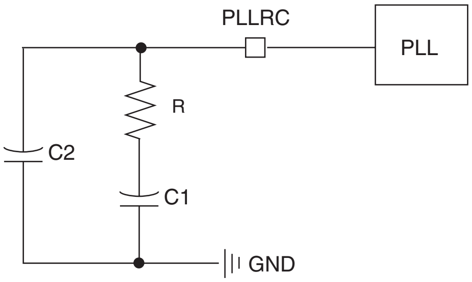

DIVB
MULB
OUTB
MAINCK
Divider B
PLL B
PLLBCK
DIVA
MULA
OUTA
Divider A
PLL A
PLLACK
PLLRCA
PLLBCOUNT
PLL B Counter
LOCKB
PLLACOUNT
PLL A Counter
LOCKA
SLCK

24.1.5.1 PLL Filter

The PLLA requires connection to an external second-order filter through the PLLRCA pin. Figure 24-5 shows a schematic of this filter.

PLLB has its own internal filter which is tuned for optimum operation with a 12 MHz input clock frequency and an output frequency of 96 MHz for generating the USB clock. Use of any other frequency for the input clock or output setting for PLLB will likely result in increased jitter and reduced quality of the PLLB output clock.

Figure 24-5. PLL Capacitors and Resistors


PLLRC
PLL
R
C2
C1
GND

Values of R, C1 and C2 to be connected to the PLLRC pin must be calculated as a function of the PLL input frequency, the PLL output frequency and the phase margin. A trade-off has to be found between output signal overshoot and startup time.

24.1.5.2 *Divider and Phase Lock Loop Programming*

The divider can be set between 1 and 255 in steps of 1. When a divider field (DIV) is set to 0, the output of the corresponding divider and the PLL output is a continuous signal at level 0. On reset, each DIV field is set to 0, thus the corresponding PLL input clock is set to 0.

The PLL allows multiplication of the divider's outputs. The PLL clock signal has a frequency that depends on the respective source signal frequency and on the parameters DIV and MUL. The factor applied to the source signal frequency is (MUL + 1)/DIV. When MUL is written to 0, the corresponding PLL is disabled and its power consumption is saved. Re-enabling the PLL can be performed by writing a value higher than 0 in the MUL field.

Whenever the PLL is re-enabled or one of its parameters is changed, the LOCK bit (LOCKA or LOCKB) in PMC_SR is automatically cleared. The values written in the PLLCOUNT field (PLLACOUNT or PLLBCOUNT) in CKGR_PLLR (CKGR_PLLAR or CKGR_PLLBR), are loaded in the PLL counter. The PLL counter then decrements at the speed of the Slow Clock until it reaches 0. At this time, the LOCK bit is set in PMC_SR and can trigger an interrupt to the processor. The user has to load the number of Slow Clock cycles required to cover the PLL transient time into the PLLCOUNT field. The transient time depends on the PLL filter. The initial state of the PLL and its target frequency can be calculated using a specific tool provided by Atmel.

For PLLB, the values of DIV and MUL should be set to produce an input clock frequency of 12MHz and an output clock frequency of 96MHz for optimal operation for USB support. Any other settings will likely result in result in reduced quality of the PLLB output clock.

24.2 Power Management Controller (PMC)

24.2.1 Description

The Power Management Controller (PMC) optimizes power consumption by controlling all system and user peripheral clocks. The PMC enables/disables the clock inputs to many of the peripherals and the ARM Processor.

The Power Management Controller provides the following clocks:

- MCK, the Master Clock, programmable from a few hundred Hz to the maximum operating frequency of the device. It is available to the modules running permanently, such as the AIC and the Memory Controller.
- Processor Clock (PCK), switched off when entering processor in Idle Mode.
- Peripheral Clocks, typically MCK, provided to the embedded peripherals (USART, SSC, SPI, TWI, TC, MCI, etc.) and independently controllable. In order to reduce the number of clock names in a product, the Peripheral Clocks are named MCK in the product datasheet.
- HClocks (HCKx), provided to the AHB high speed peripherals and independently controllable.
- UHP Clock (UHPCK), required by USB Host Port operations.
- UDP Clock (UDPCK), required by USB Device Port operations.
- Programmable Clock Outputs can be selected from the clocks provided by the clock generator and driven on the PCKx pins.

24.2.2 Master Clock Controller

The Master Clock Controller provides selection and division of the Master Clock (MCK). MCK is the clock provided to all the peripherals and the memory controller.

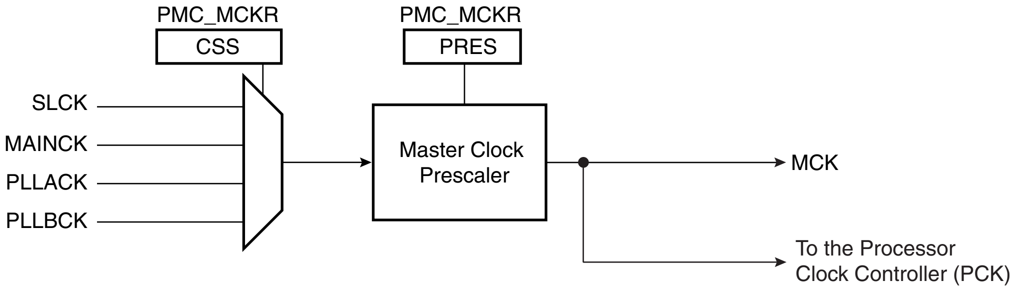
The Master Clock is selected from one of the clocks provided by the Clock Generator. Selecting the Slow Clock provides a Slow Clock signal to the whole device. Selecting the Main Clock saves power consumption of the PLLs.

The Master Clock Controller is made up of a clock selector and a prescaler.

The Master Clock selection is made by writing the CSS field (Clock Source Selection) in PMC_MCKR (Master Clock Register). The prescaler supports the division by a power of 2 of the selected clock between 1 and 64. The PRES field in PMC_MCKR programs the prescaler.

Each time PMC_MCKR is written to define a new Master Clock, the MCKRDY bit is cleared in PMC_SR. It reads 0 until the Master Clock is established. Then, the MCKRDY bit is set and can trigger an interrupt to the processor. This feature is useful when switching from a high-speed clock to a lower one to inform the software when the change is actually done.

Figure 24-6. Master Clock Controller



24.2.3 Processor Clock Controller

The PMC features a Processor Clock Controller (PCK) that implements the Processor Idle Mode. The Processor Clock can be disabled by writing the System Clock Disable Register (PMC_SCDR). The status of this clock (at least for debug purposes) can be read in the System Clock Status Register (PMC_SCSR).

The Processor Clock PCK is enabled after a reset and is automatically re-enabled by any enabled interrupt. The Processor Idle Mode is achieved by disabling the Processor Clock which is automatically re-enabled by any enabled fast or normal interrupt, or by the reset of the product. When the Processor Clock is disabled, the current instruction is finished before the clock is stopped, but this does not prevent data transfers from other masters of the system bus.

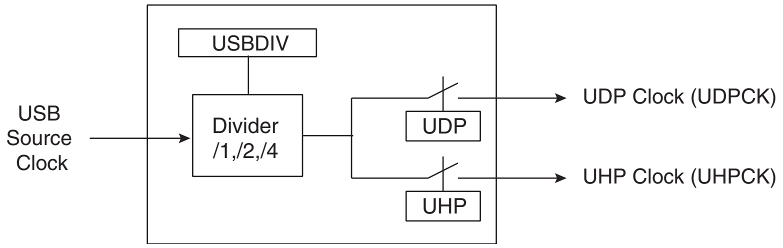
24.2.4 USB Clock Controller

The USB Source Clock is always generated from the PLL B output. If using USB, the user must program the PLLB to generate a 96 MHz signal (with an accuracy of ± 0.25%) and then further divide this clock by 2 to generate a 48 MHz clock by programming the appropriate value into the USBDIV bits in CKGR_PLLBR (see Figure 24-7).

When the PLL B output is stable, i.e., the LOCKB is set:

- The USB host clock can be enabled by setting the UHP bit in PMC_SCER. To save power on this peripheral when it is not used, the user can set the UHP bit in PMC_SCDR. The UHP bit in PMC_SCSR gives the activity of this clock. A USB host port requires both the 12/48 MHz signal and the Master Clock. The Master Clock may be controlled via the Master Clock Controller.
- The USB device clock can be enabled by setting the UDP bit in PMC_SCER. To save power on this peripheral when it is not used, the user can set the UDP bit in PMC_SCDR. The UDP bit in PMC_SCSR gives the activity of this clock. The USB device port require both the 48 MHz signal and the Master Clock. The Master Clock may be controlled via the Master Clock Controller.

Figure 24-7. USB Clock Controller



24.2.5 Peripheral Clock Controller

The Power Management Controller controls the clocks of each embedded peripheral by the way of the Peripheral Clock Controller. The user can individually enable and disable the Master Clock on the peripherals by writing into the Peripheral Clock Enable (PMC_PCER) and Peripheral Clock Disable (PMC_PCDR) registers. The status of the peripheral clock activity can be read in the Peripheral Clock Status Register (PMC_PCSR).

When a peripheral clock is disabled, the clock is immediately stopped. The peripheral clocks are automatically disabled after a reset.

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

The bit number within the Peripheral Clock Control registers (PMC_PCER, PMC_PCDR, and PMC_PCSR) is the Peripheral Identifier defined at the product level. Generally, the bit number corresponds to the interrupt source number assigned to the peripheral.

24.2.6 HClock Controller

The PMC facilitates control of the clocks of each specific AHB peripheral by means of the HClock Controller. The user can individually enable and disable the Hclocks by writing into the registers; Peripheral Clock Enable (PMC_PCER) and Peripheral Clock Disable (PMC_PCDR). The status of HClock activity can be read in the Peripheral Clock Status Register (PMC_PCSR).

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

4 HClocks can be controlled.

24.2.7 Programmable Clock Output Controller

The PMC controls 4 signals to be output on external pins PCKx. Each signal can be independently programmed via the PMC_PCKx registers.

PCKx can be independently selected between the Slow clock, the PLL A output, the PLL B output and the main clock by writing the CSS field in PMC_PCKx. Each output signal can also be divided by a power of 2 between 1 and 64 by writing the PRES (Prescaler) field in PMC_PCKx.

Each output signal can be enabled and disabled by writing 1 in the corresponding bit, PCKx of PMC_SCER and PMC_SCDR, respectively. Status of the active programmable output clocks are given in the PCKx bits of PMC_SCSR (System Clock Status Register).

Moreover, like the PCK, a status bitin PMC_SR indicates that the Programmable Clock is actually what has been programmed in the Programmable Clock registers.

As the Programmable Clock Controller does not manage with glitch prevention when switching clocks, it is strongly recommended to disable the Programmable Clock before any configuration change and to re-enable it after the change is actually performed.

24.2.8 Programming Sequence

1. Enabling the Main Oscillator:

   The main oscillator is enabled by setting the MOSCEN field in the CKGR_MOR register. In some cases it may be advantageous to define a start-up time. This can be achieved by writing a value in the OSCOUNT field in the CKGR_MOR register.

Once this register has been correctly configured, the user must wait for MOSCS field in the PMC_SR register to be set. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to MOSCS has been enabled in the PMC_IER register.

Code Example:

```
write_register(CKGR_MOR,0x00000701)
```

Start Up Time = 8 * OSCOUNT / SLCK = 56 Slow Clock Cycles.

So, the main oscillator will be enabled (MOSCS bit set) after 56 Slow Clock Cycles.

2. Checking the Main Oscillator Frequency (Optional):

   In some situations the user may need an accurate measure of the main oscillator frequency. This measure can be accomplished via the CKGR_MCFR register.

   Once the MAINRDY field is set in CKGR_MCFR register, the user may read the MAINF field in CKGR_MCFR register. This provides the number of main clock cycles within sixteen slow clock cycles.

3. Setting PLL A and divider A:

   All parameters necessary to configure PLL A and divider A are located in the CKGR_PLLAR register.

   It is important to note that Bit 29 must always be set to 1 when programming the CKGR_PLLAR register.

   The DIVA field is used to control the divider A itself. The user can program a value between 0 and 255. Divider A output is divider A input divided by DIVA. By default, DIVA parameter is set to 0 which means that divider A is turned off.

   The OUTA field is used to select the PLL A output frequency range.

   The MULA field is the PLL A multiplier factor. This parameter can be programmed between 0 and 2047. If MULA is set to 0, PLL A will be turned off. Otherwise PLL A output frequency is PLL A input frequency multiplied by (MULA + 1).

   The PLLACOUNT field specifies the number of slow clock cycles before LOCKA bit is set in the PMC_SR register after CKGR_PLLAR register has been written.

   Once CKGR_PLLAR register has been written, the user is obliged to wait for the LOCKA bit to be set in the PMC_SR register. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to LOCKA has been enabled in the PMC_IER register.

   All parameters in CKGR_PLLAR can be programmed in a single write operation. If at some stage one of the following parameters, SRCA, MULA, DIVA is modified,

   LOCKA bit will go low to indicate that PLL A is not ready yet. When PLL A is locked, LOCKA will be set again. User has to wait for LOCKA bit to be set before using the PLL A output clock.

   Code Example:

```
write_register(CKGR_PLLAR,0x20030605)
```

PLL A and divider A are enabled. PLL A input clock is main clock divided by 5. PLL An output clock is PLL A input clock multiplied by 4. Once CKGR_PLLAR has been written, LOCKA bit will be set after six slow clock cycles.

4. Setting PLL B and divider B:

All parameters needed to configure PLL B and divider B are located in the CKGR_PLLBR register.

The DIVB field is used to control divider B itself. A value between 0 and 255 can be programmed. Divider B output is divider B input divided by DIVB parameter. By default DIVB parameter is set to 0 which means that divider B is turned off.

The OUTB field is used to select the PLL B output frequency range.

The MULB field is the PLL B multiplier factor. This parameter can be programmed between 0 and 2047. If MULB is set to 0, PLL B will be turned off, otherwise the PLL B output frequency is PLL B input frequency multiplied by (MULB + 1).

As stated previously in this chapter and due to the tuning of the internal PLLB filter for USB operation, DIVB and MULB should be programmed to generate a 96 MHz PLL clock for optimum performance. Other settings are likely to produce an output clock with reduced quality.

The PLLBCOUNT field specifies the number of slow clock cycles before LOCKB bit is set in the PMC_SR register after CKGR_PLLBR register has been written.

Once the PMC_PLLB register has been written, the user must wait for the LOCKB bit to be set in the PMC_SR register. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to LOCKB has been enabled in the PMC_IER register. All parameters in CKGR_PLLBR can be programmed in a single write operation. If at some stage one of the following parameters, MULB, DIVB is modified, LOCKB bit will go low to indicate that PLL B is not ready yet. When PLL B is locked, LOCKB will be set again. The user is constrained to wait for LOCKB bit to be set before using the PLL A output clock.

The USBDIV field is used to control the additional divider by 1, 2 or 4, which generates the USB clock(s). As mentioned in an earlier paragraph, should be normally set to divide the 96 MHz PLLB output clock by 2 and thereby generate the 48 MHz USB clocks.

Code Example:

```
write_register(CKGR_PLLBR,0x00040805)
```

If PLL B and divider B are enabled, the PLL B input clock is the main clock. PLL B output clock is PLL B input clock multiplied by 5. Once CKGR_PLLBR has been written, LOCKB bit will be set after eight slow clock cycles.

5. Selection of Master Clock and Processor Clock

The Master Clock and the Processor Clock are configurable via the PMC_MCKR register.

The CSS field is used to select the Master Clock divider source. By default, the selected clock source is slow clock.

The PRES field is used to control the Master Clock prescaler. The user can choose between different values (1, 2, 4, 8, 16, 32, 64). Master Clock output is prescaler input divided by PRES parameter. By default, PRES parameter is set to 1 which means that master clock is equal to slow clock.

Once the PMC_MCKR register has been written, the user must wait for the MCKRDY bit to be set in the PMC_SR register. This can be done either by polling the status register or by waiting for the interrupt line to be raised if the associated interrupt to MCKRDY has been enabled in the PMC_IER register.

The PMC_MCKR register must not be programmed in a single write operation. The preferred programming sequence for the PMC_MCKR register is as follows:

- If a new value for CSS field corresponds to PLL Clock,
  - Program the PRES field in the PMC_MCKR register.
  - Wait for the MCKRDY bit to be set in the PMC_SR register.
  - Program the CSS field in the PMC_MCKR register.
  - Wait for the MCKRDY bit to be set in the PMC_SR register.
- If a new value for CSS field corresponds to Main Clock or Slow Clock,
  - Program the CSS field in the PMC_MCKR register.
  - Wait for the MCKRDY bit to be set in the PMC_SR register.
  - Program the PRES field in the PMC_MCKR register.
  - Wait for the MCKRDY bit to be set in the PMC_SR register.

If at some stage one of the following parameters, CSS or PRES, is modified, the MCKRDY bit will go low to indicate that the Master Clock and the Processor Clock are not ready yet. The user must wait for MCKRDY bit to be set again before using the Master and Processor Clocks.

Note: IF PLLx clock was selected as the Master Clock and the user decides to modify it by writing in CKGR_PLLR (CKGR_PLLAR or CKGR_PLLBR), the MCKRDY flag will go low while PLL is unlocked. Once PLL is locked again, LOCK (LOCKA or LOCKB) goes high and MCKRDY is set. While PLLA is unlocked, the Master Clock selection is automatically changed to Slow Clock. While PLLB is unlocked, the Master Clock selection is automatically changed to Main Clock. For further information, see Section 24.2.9.2. "Clock Switching Waveforms" on page 221.

Code Example:

```
write_register(PMC_MCKR,0x00000001)
wait (MCKRDY=1)
write_register(PMC_MCKR,0x00000011)
wait (MCKRDY=1)
```

The Master Clock is main clock divided by 16.

The Processor Clock is the Master Clock.

6. Selection of Programmable clocks

   Programmable clocks are controlled via registers; PMC_SCER, PMC_SCDR and PMC_SCSR.

   Programmable clocks can be enabled and/or disabled via the PMC_SCER and PMC_SCDR registers. Depending on the system used, 4 Programmable clocks can be enabled or disabled. The PMC_SCSR provides a clear indication as to which Programmable clock is enabled. By default all Programmable clocks are disabled.

   PMC_PCKx registers are used to configure Programmable clocks.

   The CSS field is used to select the Programmable clock divider source. Four clock options are available: main clock, slow clock, PLLACK, PLLBCK. By default, the clock source selected is slow clock.

   The PRES field is used to control the Programmable clock prescaler. It is possible to choose between different values (1, 2, 4, 8, 16, 32, 64). Programmable clock output is prescaler input divided by PRES parameter. By default, the PRES parameter is set to 1 which means that master clock is equal to slow clock.

   Once the PMC_PCKx register has been programmed, The corresponding Programmable clock must be enabled and the user is constrained to wait for the PCKRDYx bit to be set in the PMC_SR register. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to PCKRDYx has been enabled in the PMC_IER register. All parameters in PMC_PCKx can be programmed in a single write operation.

   If the CSS and PRES parameters are to be modified, the corresponding Programmable clock must be disabled first. The parameters can then be modified. Once this has been done, the user must re-enable the Programmable clock and wait for the PCKRDYx bit to be set.

   Code Example:

   ```
   write_register(PMC_PCK0,0x00000015)
   ```

   Programmable clock 0 is main clock divided by 32.

7. Enabling Peripheral Clocks

   Once all of the previous steps have been completed, the peripheral clocks can be enabled and/or disabled via registers PMC_PCER and PMC_PCDR.

   Depending on the system used, 24 peripheral clocks can be enabled or disabled. The PMC_PCSR provides a clear view as to which peripheral clock is enabled.

Note: Each enabled peripheral clock corresponds to Master Clock.

   Code Examples:

   ```
   write_register(PMC_PCER,0x00000110)
   ```

Peripheral clocks 4 and 8 are enabled.

```
write_register(PMC_PCDR,0x00000010)
```

Peripheral clock 4 is disabled.

8. Enabling HClocks

   Once all of the previous steps have been completed, the HClocks can be enabled and/or disabled via registers; PMC_PCER and PMC_PCDR.

   Depending on the system used, 4 HClocks can be enabled or disabled.

   The PMC_PCSR register indicates which HClock is enabled.

Note: Each enabled HClock corresponds to Master Clock.

Code Examples:

```
write_register(PMC_PCER,0x24000000)
```

HClocks 0 and 3 are enabled.

24.2.9 Clock Switching Details

24.2.9.1 Master Clock Switching Timings

Table 24-1 and Table 24-2 give the worst case timings required for the Master Clock to switch from one selected clock to another one. This is in the event that the prescaler is de-activated. When the prescaler is activated, an additional time of 64 clock cycles of the new selected clock has to be added.

Table 24-1. Clock Switching Timings (Worst Case)

From **To**	**Main Clock**	**SLCK**	**PLL Clock**
Main Clock	–	4 x SLCK + 2.5 x Main Clock	3 x PLL Clock + 4 x SLCK + 1 x Main Clock
SLCK	0.5 x Main Clock + 4.5 x SLCK	–	3 x PLL Clock + 5 x SLCK
PLL Clock	0.5 x Main Clock + 4 x SLCK + PLLCOUNT x SLCK + 2.5 x PLLx Clock	2.5 x PLL Clock + 5 x SLCK + PLLCOUNT x SLCK	2.5 x PLL Clock + 4 x SLCK + PLLCOUNT x SLCK

Notes: 1. PLL designates either the PLL A or the PLL B Clock.
2. PLLCOUNT designates either PLLACOUNT or PLLBCOUNT.

Table 24-2. Clock Switching Timings Between Two PLLs (Worst Case)

From **To**	**PLLA Clock**	**PLLB Clock**
PLLA Clock	2.5 x PLLA Clock + 4 x SLCK + PLLACOUNT x SLCK	3 x PLLA Clock + 4 x SLCK + 1.5 x PLLA Clock
PLLB Clock	3 x PLLB Clock + 4 x SLCK + 1.5 x PLLB Clock	2.5 x PLLB Clock + 4 x SLCK + PLLBCOUNT x SLCK

24.2.9.2 Clock Switching Waveforms

Figure 24-8. Switch Master Clock from Slow Clock to PLL Clock

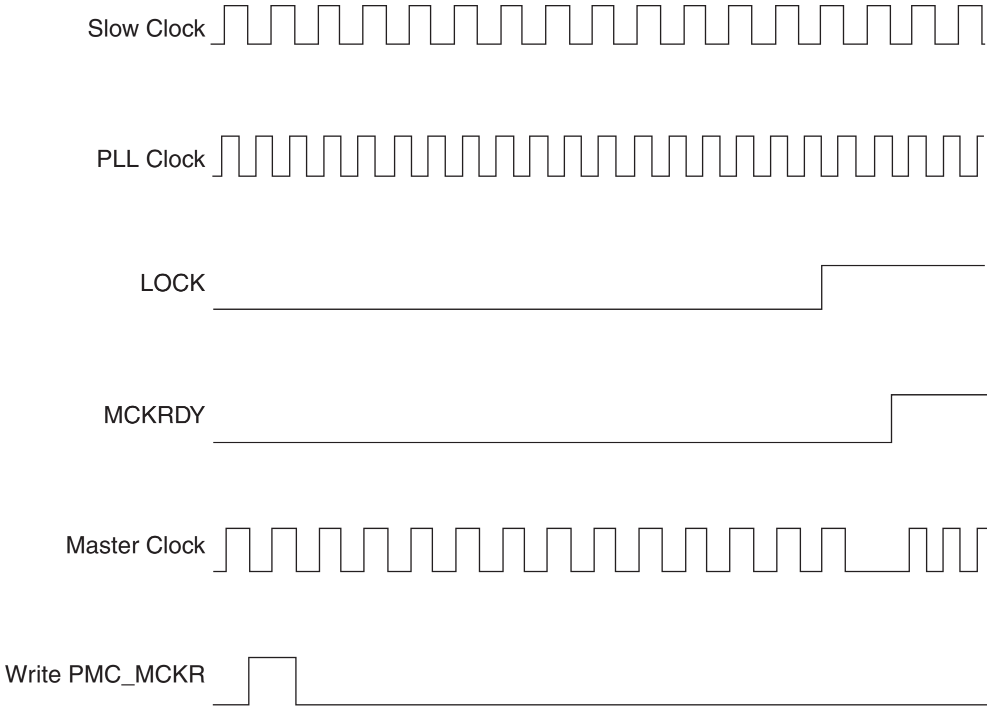

Slow Clock
PLL Clock
LOCK
MCKRDY
Master Clock
Write PMC_MCKR

Figure 24-9. Switch Master Clock from Main Clock to Slow Clock

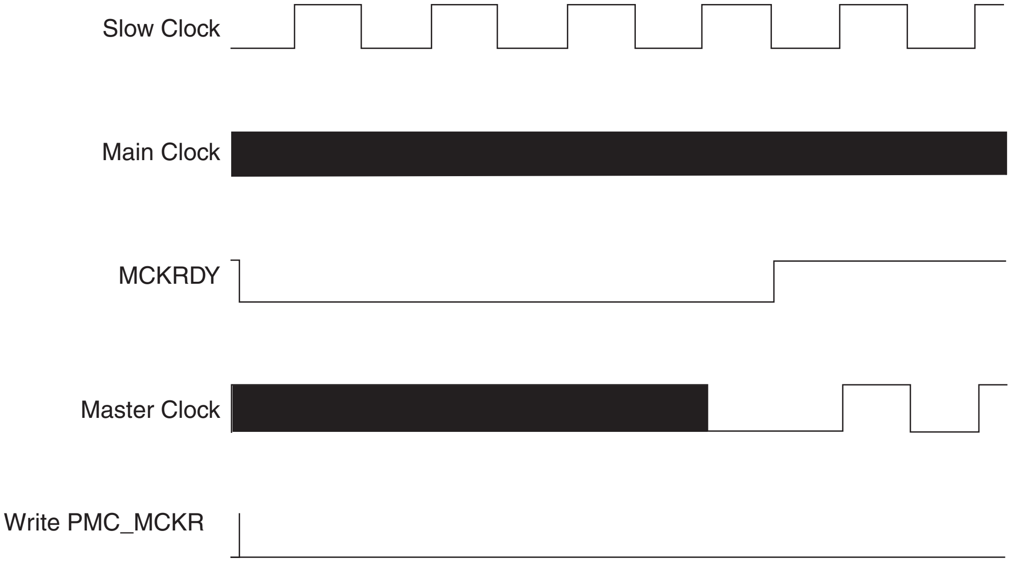

Slow Clock
Main Clock
MCKRDY
Master Clock
Write PMC_MCKR

Figure 24-10. Change PLLA Programming

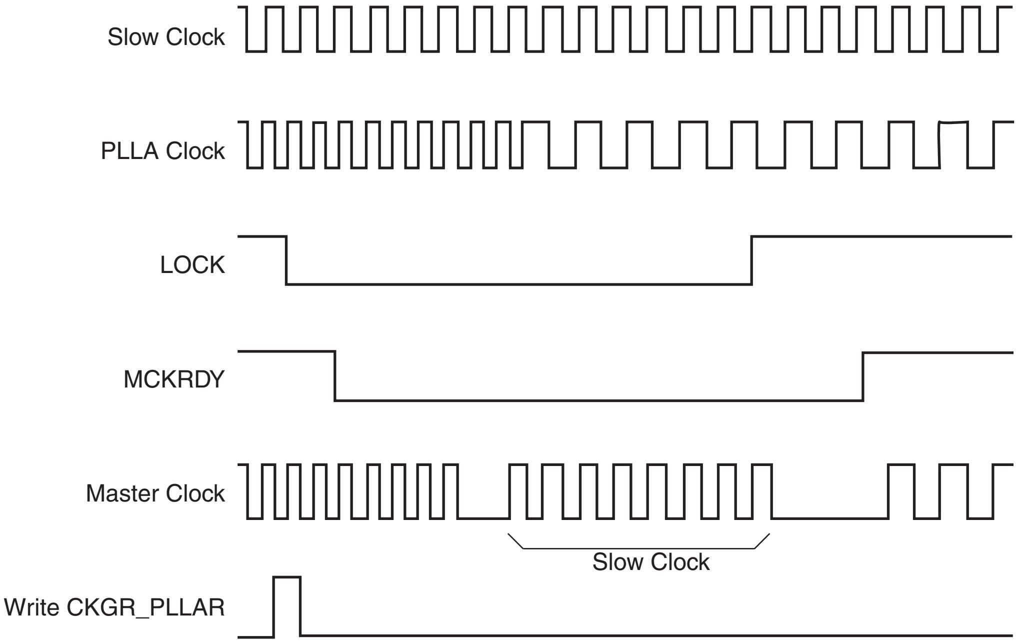

Slow Clock
PLLA Clock
LOCK
MCKRDY
Master Clock
Slow Clock
Write CKGR_PLLAR

Figure 24-11. Change PLLB Programming

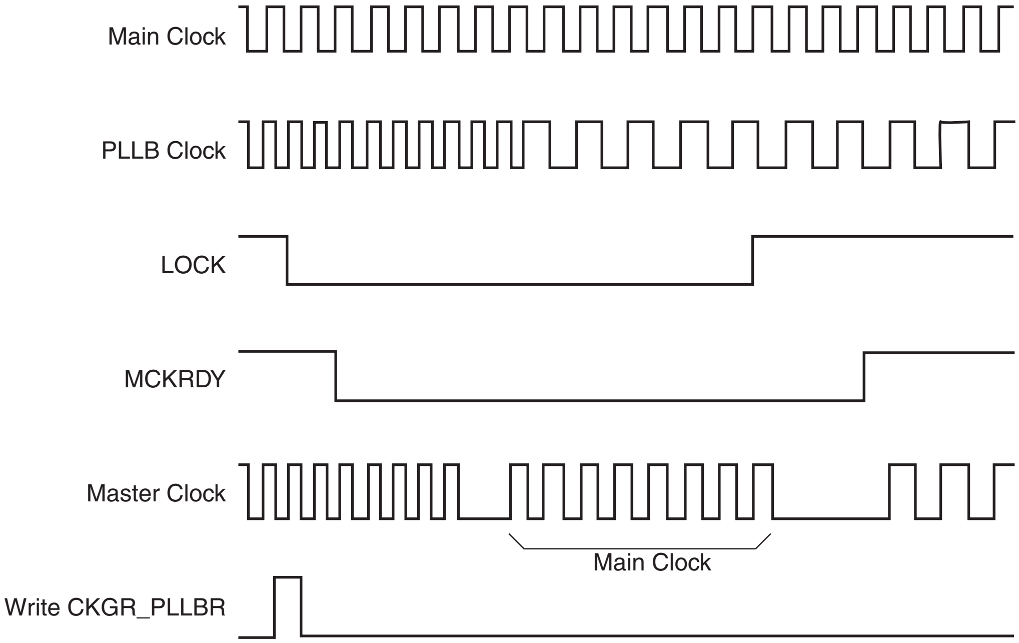
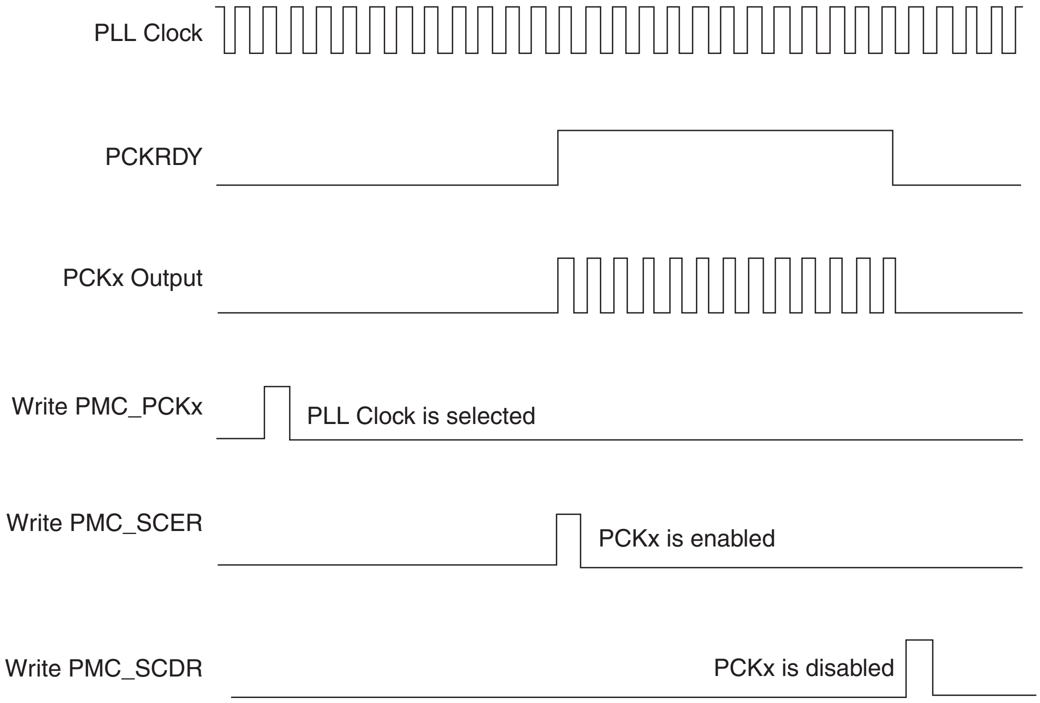

Main Clock
PLLB Clock
LOCK
MCKRDY
Master Clock
Main Clock
Write CKGR_PLLBR

Figure 24-12. Programmable Clock Output Programming


PLL Clock
PCKRDY
PCKx Output
Write PMC_PCKx
PLL Clock is selected
Write PMC_SCER
PCKx is enabled
Write PMC_SCDR
PCKx is disabled

24.2.10 Power Management Controller (PMC) User Interface

Table 24-3. Register Mapping

Offset	Register	Name	Access	Reset Value
0x0000	System Clock Enable Register	PMC_SCER	Write-only	–
0x0004	System Clock Disable Register	PMC_SCDR	Write-only	–
0x0008	System Clock Status Register	PMC _SCSR	Read-only	0x01
0x000C	Reserved	–	–	–
0x0010	Peripheral Clock Enable Register	PMC _PCER	Write-only	–
0x0014	Peripheral Clock Disable Register	PMC_PCDR	Write-only	–
0x0018	Peripheral Clock Status Register	PMC_PCSR	Read-only	0x0
0x001C	Reserved	–	–	–
0x0020	Main Oscillator Register	CKGR_MOR	Read/Write	0x0
0x0024	Main Clock Frequency Register	CKGR_MCFR	Read-only	0x0
0x0028	PLL A Register	CKGR_PLLAR	ReadWrite	0x3F00
0x002C	PLL B Register	CKGR_PLLBR	ReadWrite	0x3F00
0x0030	Master Clock Register	PMC_MCKR	Read/Write	0x0
0x0038	Reserved	–	–	–
0x003C	Reserved	–	–	–
0x0040	Programmable Clock 0 Register	PMC_PCK0	Read/Write	0x0
0x0044	Programmable Clock 1 Register	PMC_PCK1	Read/Write	0x0
...	...	...	...	...
0x0060	Interrupt Enable Register	PMC_IER	Write-only	--
0x0064	Interrupt Disable Register	PMC_IDR	Write-only	--
0x0068	Status Register	PMC_SR	Read-only	0x08
0x006C	Interrupt Mask Register	PMC_IMR	Read-only	0x0
0x0070 - 0x007C	Reserved	–	–	–

24.2.10.1 PMC System Clock Enable Register

Register Name: PMC_SCER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	PCK3	PCK2	PCK1	PCK0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UDP	UHP	–	–	–	–	-	PCK

- **UHP: USB Host Port Clock Enable**

0 = No effect.

1 = Enables the 12 and 48 MHz clock of the USB Host Port.

- **UDP: USB Device Port Clock Enable**

0 = No effect.

1 = Enables the 48 MHz clock of the USB Device Port.

- **PCKx: Programmable Clock x Output Enable**

0 = No effect.

1 = Enables the corresponding Programmable Clock output.

24.2.10.2 PMC System Clock Disable Register

Register Name: PMC_SCDR

Access Type: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	PCK3	PCK2	PCK1	PCK0

7	6	5	4	3	2	1	0
UDP	UHP	–	–	–	–	-	PCK

- **PCK: Processor Clock Disable**

0 = No effect.

1 = Disables the Processor clock. This is used to enter the processor in Idle Mode.

- **UHP: USB Host Port Clock Disable**

0 = No effect.

1 = Disables the 12 and 48 MHz clock of the USB Host Port.

- **UDP: USB Device Port Clock Disable**

0 = No effect.

1 = Disables the 48 MHz clock of the USB Device Port.

- **PCKx: Programmable Clock x Output Disable**

0 = No effect.

1 = Disables the corresponding Programmable Clock output.

24.2.10.3 PMC System Clock Status Register

Register Name: PMC_SCSR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	PCK3	PCK2	PCK1	PCK0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UDP	UHP	–	–	–	–	-	PCK

- **PCK: Processor Clock Status**

0 = The Processor clock is disabled.

1 = The Processor clock is enabled.

- **UHP: USB Host Port Clock Status**

0 = The 12 and 48 MHz clock (UHPCK) of the USB Host Port is disabled.

1 = The 12 and 48 MHz clock (UHPCK) of the USB Host Port is enabled.

- **UDP: USB Device Port Clock Status**

0 = The 48 MHz clock (UDPCK) of the USB Device Port is disabled.

1 = The 48 MHz clock (UDPCK) of the USB Device Port is enabled.

- **PCKx: Programmable Clock x Output Status**

0 = The corresponding Programmable Clock output is disabled.

1 = The corresponding Programmable Clock output is enabled.

24.2.10.4 PMC Peripheral Clock Enable Register

Register Name: PMC_PCER

Access Type: Write-only

31	30	29	28	27	26	25	24
-	-	HCK3(PID29)	HCK2(PID28)	HCK1(PID27)	HCK0(PID26)	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	-	-

- **PIDx: Peripheral Clock x Enable**

0 = No effect.

1 = Enables the corresponding peripheral clock.

Note: PID2 to PID29 refer to identifiers as defined in Section 10.2 Peripheral Identifiers.

Note: Programming the control bits of the Peripheral ID that are not implemented has no effect on the behavior of the PMC.

- **HCKx: HClock x Output Enable**

0 = No effect.

1 = Enables the corresponding HClock output.

Note: HCK0 - HCK3 correspond to PID26 - PID29 and therefore control the AHB clocks for the MP Block Master A, B, C, and D respectively as defined in Section 10.2 Peripheral Identifiers.

24.2.10.5 PMC Peripheral Clock Disable Register

Register Name: PMC_PCDR

Access Type: Write-only

31	30	29	28	27	26	25	24
-	-	HCK3(PID29)	HCK2(PID28)	HCK1(PID27)	HCK0(PID26)	PID25	PID24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PID7	PID6	PID5	PID4	PID3	PID2	-	-

- **PIDx: Peripheral Clock x Disable**

0 = No effect.

1 = Disables the corresponding peripheral clock.

Note: PID2 to PID29 refer to identifiers as defined in Section 10.2 Peripheral Identifiers.

- **HCKx: Hclock x Output Disable**

0 = No effect.

1 = Disables the corresponding HClock output.

Note: HCK0 - HCK3 correspond to PID26 - PID29 and therefore control the AHB clocks for the MP Block Master A, B, C, and D respectively as defined in Section 10.2 Peripheral Identifiers.

24.2.10.6 PMC Peripheral Clock Status Register

Register Name: PMC_PCSR

Access Type: Read-only

31	30	29	28	27	26	25	24
-	-	HCK3(PID29)	HCK2(PID28)	HCK1(PID27)	HCK0(PID26)	PID25	PID24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PID7	PID6	PID5	PID4	PID3	PID2	–	–

- **PIDx: Peripheral Clock x Status**

0 = The corresponding peripheral clock is disabled.

1 = The corresponding peripheral clock is enabled.

Note: PID2 to PID29 refer to identifiers as defined in Section 10.2 Peripheral Identifiers.

- **HCKx: HClock Output x Status**

0 = The corresponding HClock output is disabled.

1 = The corresponding HClock output is enabled.

Note: HCK0 - HCK3 correspond to PID26 - PID29 and therefore control the AHB clocks for the MP Block Master A, B, C, and D respectively as defined in Section 10.2 Peripheral Identifiers.

24.2.10.7 PMC Clock Generator Main Oscillator Register

Register Name: CKGR_MOR

Access Type: Read/Write

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

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

15	14	13	12	11	10	9	8
OSCOUNT							

7	6	5	4	3	2	1	0
–	–	–	–	–	–	OSCBYPASS	MOSCEN

- **MOSCEN: Main Oscillator Enable**

A crystal must be connected between XIN and XOUT.

0 = The Main Oscillator is disabled.

1 = The Main Oscillator is enabled. OSCBYPASS must be set to 0.

When MOSCEN is set, the MOSCS flag is set once the Main Oscillator startup time is achieved.

- **OSCBYPASS: Oscillator Bypass**

0 = No effect.

1 = The Main Oscillator is bypassed. MOSCEN must be set to 0. An external clock must be connected on XIN.

When OSCBYPASS is set, the MOSCS flag in PMC_SR is automatically set.

Clearing MOSCEN and OSCBYPASS bits allows resetting the MOSCS flag.

- **OSCOUNT: Main Oscillator Start-up Time**

Specifies the number of Slow Clock cycles multiplied by 8 for the Main Oscillator start-up time.

24.2.10.8 PMC Clock Generator Main Clock Frequency Register

Register Name: CKGR_MCFR

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
MAINF							

7	6	5	4	3	2	1	0
MAINF							

- **MAINF: Main Clock Frequency**

Gives the number of Main Clock cycles within 16 Slow Clock periods.

- **MAINRDY: Main Clock Ready**

0 = MAINF value is not valid or the Main Oscillator is disabled.

1 = The Main Oscillator has been enabled previously and MAINF value is available.

24.2.10.9 PMC Clock Generator PLL A Register

Register Name: CKGR_PLLAR

Access Type: Read/Write

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

23	22	21	20	19	18	17	16
MULA							

15	14	13	12	11	10	9	8
OUTA		PLLACOUNT					

7	6	5	4	3	2	1	0
DIVA							

Possible limitations on PLL A input frequencies and multiplier factors should be checked before using the PMC.

Warning: Bit 29 must always be set to 1 when programming the CKGR_PLLAR register.

- **DIVA: Divider A**

DIVA	Divider Selected
0	Divider output is 0
1	Divider is bypassed
2 - 255	Divider output is the Main Clock divided by DIVA.

- **PLLACOUNT: PLL A Counter**

Specifies the number of Slow Clock cycles before the LOCKA bit is set in PMC_SR after CKGR_PLLAR is written.

- **OUTA: PLL A Clock Frequency Range**

To optimize clock performance, this field must be programmed as specified in “PLL Characteristics” in the Electrical Characteristics section of the product datasheet.

- **MULA: PLL A Multiplier**

0 = The PLL A is deactivated.

1 up to 2047 = The PLL A Clock frequency is the PLL A input frequency multiplied by MULA + 1.

24.2.10.10 PMC Clock Generator PLL B Register

Register Name: CKGR_PLLBR

Access Type: Read/Write

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

23	22	21	20	19	18	17	16
MULB							

15	14	13	12	11	10	9	8
OUTB		PLLBCOUNT					

7	6	5	4	3	2	1	0
DIVB							

Possible limitations on PLL B input frequencies and multiplier factors should be checked before using the PMC.

- **DIVB: Divider B**

DIVB	Divider Selected
0	Divider output is 0
1	Divider is bypassed
2 - 255	Divider output is the selected clock divided by DIVB.

- **PLLBCOUNT: PLL B Counter**

Specifies the number of slow clock cycles before the LOCKB bit is set in PMC_SR after CKGR_PLLBR is written.

- **OUTB: PLLB Clock Frequency Range**

To optimize clock performance, this field must be programmed as specified in “PLL Characteristics” in the Electrical Characteristics section of the product datasheet.

- **MULB: PLL B Multiplier**

0 = The PLL B is deactivated.

1 up to 2047 = The PLL B Clock frequency is the PLL B input frequency multiplied by MULB + 1.

- **USBDIV: Divider for USB Clock**

USBDIV		Divider for USB Clock(s)
0	0	Divider output is PLL B clock output.
0	1	Divider output is PLL B clock output divided by 2.
1	0	Divider output is PLL B clock output divided by 4.
1	1	Reserved.

24.2.10.11 PMC Master Clock Register

Register Name: PMC_MCKR

Access Type: Read/Write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	-	

7	6	5	4	3	2	1	0
–	–	–	PRES			CSS	

- **CSS: Master Clock Selection**

CSS		Clock Source Selection
0	0	Slow Clock is selected
0	1	Main Clock is selected
1	0	PLL A Clock is selected
1	1	PLL B Clock is selected

- **PRES: Processor Clock Prescaler**

PRES			Processor Clock
0	0	0	Selected clock
0	0	1	Selected clock divided by 2
0	1	0	Selected clock divided by 4
0	1	1	Selected clock divided by 8
1	0	0	Selected clock divided by 16
1	0	1	Selected clock divided by 32
1	1	0	Selected clock divided by 64
1	1	1	Reserved

24.2.10.12 PMC Programmable Clock Register

Register Name: PMC_PCKx

Access Type: Read/Write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	PRES			CSS	

- **CSS: Master Clock Selection**

CSS		Clock Source Selection
0	0	Slow Clock is selected
0	1	Main Clock is selected
1	0	PLL A Clock is selected
1	1	PLL B Clock is selected

- **PRES: Programmable Clock Prescaler**

PRES			Programmable Clock
0	0	0	Selected clock
0	0	1	Selected clock divided by 2
0	1	0	Selected clock divided by 4
0	1	1	Selected clock divided by 8
1	0	0	Selected clock divided by 16
1	0	1	Selected clock divided by 32
1	1	0	Selected clock divided by 64
1	1	1	Reserved

24.2.10.13 PMC Interrupt Enable Register

Register Name: PMC_IER

Access Type: Write-only

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

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

15	14	13	12	11	10	9	8
-	-	-	-	PCKRDY3	PCKRDY2	PCKRDY1	PCKRDY0

7	6	5	4	3	2	1	0
–	-	-	-	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: Main Oscillator Status Interrupt Enable**
- **LOCKA: PLL A Lock Interrupt Enable**
- **LOCKB: PLL B Lock Interrupt Enable**
- **MCKRDY: Master Clock Ready Interrupt Enable**
- **PCKRDYx: Programmable Clock Ready x Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

24.2.10.14 PMC Interrupt Disable Register

Register Name: PMC_IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
-	-	-	-	PCKRDY3	PCKRDY2	PCKRDY1	PCKRDY0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	-	-	-	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: Main Oscillator Status Interrupt Disable**

- **LOCKA: PLL A Lock Interrupt Disable**

- **LOCKB: PLL B Lock Interrupt Disable**

- **MCKRDY: Master Clock Ready Interrupt Disable**

- **PCKRDYx: Programmable Clock Ready x Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

24.2.10.15 PMC Status Register

Register Name: PMC_SR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
-	-	-	-	PCKRDY3	PCKRDY2	PCKRDY1	PCKRDY0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OSC_SEL	-	-	-	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: MOSCS Flag Status**

0 = Main oscillator is not stabilized.

1 = Main oscillator is stabilized.

- **LOCKA: PLL A Lock Status**

0 = PLL A is not locked

1 = PLL A is locked.

- **LOCKB: PLL B Lock Status**

0 = PLL B is not locked.

1 = PLL B is locked.

- **MCKRDY: Master Clock Status**

0 = Master Clock is not ready.

1 = Master Clock is ready.

- **OSC_SEL: Slow Clock Oscillator Selection**

0 = Internal slow clock RC oscillator.

1 = External slow clock 32 kHz oscillator.

- **PCKRDYx: Programmable Clock Ready Status**

0 = Programmable Clock x is not ready.

1 = Programmable Clock x is ready.

24.2.10.16 PMC Interrupt Mask Register

Register Name: PMC_IMR

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
-	-	-	-	PCKRDY3	PCKRDY2	PCKRDY1	PCKRDY0

7	6	5	4	3	2	1	0
–	-	-	-	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: Main Oscillator Status Interrupt Mask**

- **LOCKA: PLL A Lock Interrupt Mask**

- **LOCKB: PLL B Lock Interrupt Mask**

- **MCKRDY: Master Clock Ready Interrupt Mask**

- **PCKRDYx: Programmable Clock Ready x Interrupt Mask**

0 = The corresponding interrupt is enabled.

1 = The corresponding interrupt is disabled.

25. Advanced Interrupt Controller (AIC)

25.1 Description

The Advanced Interrupt Controller (AIC) is an 8-level priority, individually maskable, vectored interrupt controller, providing handling of up to thirty-two interrupt sources. It is designed to substantially reduce the software and real-time overhead in handling internal and external interrupts.

The AIC drives the nFIQ (fast interrupt request) and the nIRQ (standard interrupt request) inputs of an ARM processor. Inputs of the AIC are either internal peripheral interrupts or external interrupts coming from the product's pins.

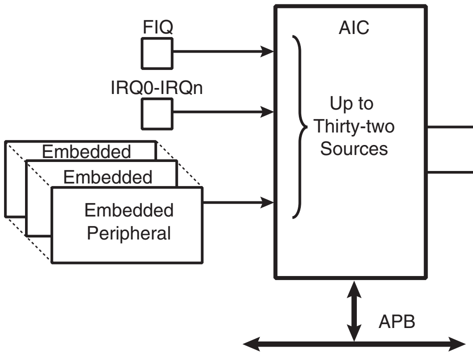
The 8-level Priority Controller allows the user to define the priority for each interrupt source, thus permitting higher priority interrupts to be serviced even if a lower priority interrupt is being treated.

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

The fast forcing feature redirects any internal or external interrupt source to provide a fast interrupt rather than a normal interrupt.

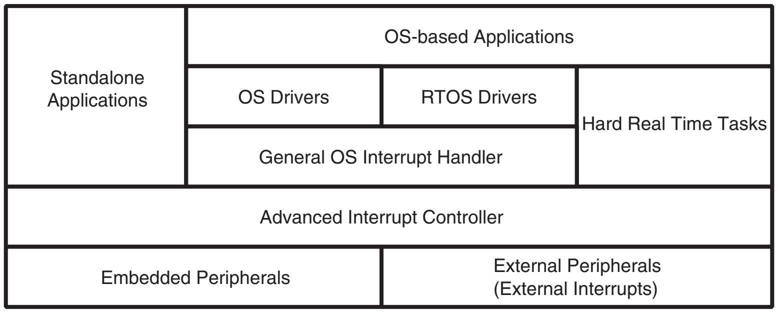
25.2 Block Diagram

Figure 25-1. Block Diagram


FIQ
IRQ0-IRQn
Embedded
Embedded
Embedded Peripheral
AIC
Up to Thirty-two Sources
ARM Processor
nFIQ
nIRQ
APB

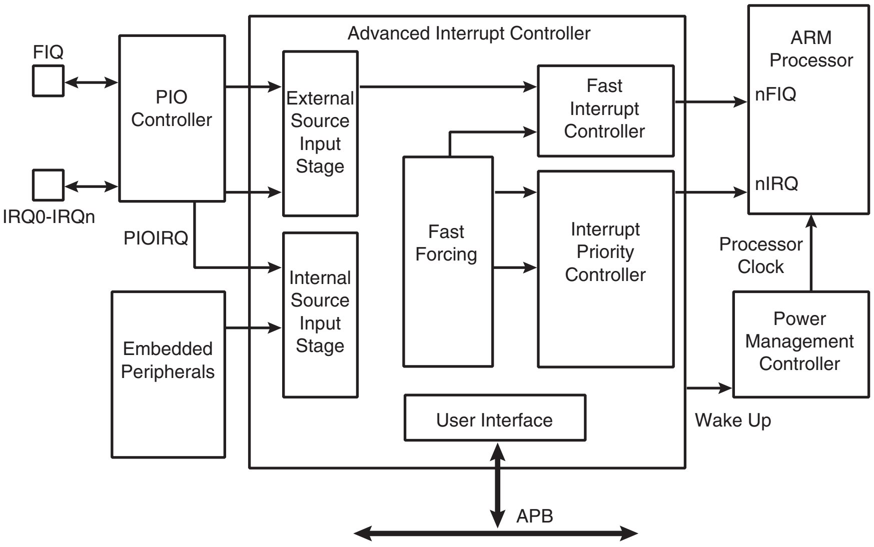
25.2.1 Application Block Diagram

Figure 25-2. Description of the Application Block


Standalone Applications
OS-based Applications
OS Drivers
RTOS Drivers
Hard Real Time Tasks
General OS Interrupt Handler
Advanced Interrupt Controller
Embedded Peripherals
External Peripherals (External Interrupts)

25.2.2 AIC Detailed Block Diagram

Figure 25-3. AIC Detailed Block Diagram


FIQ
IRQ0-IRQn
PIO Controller
PIOIRQ
Embedded Peripherals
Advanced Interrupt Controller
External Source Input Stage
Internal Source Input Stage
Fast Forcing
Fast Interrupt Controller
Interrupt Priority Controller
User Interface
ARM Processor
nFIQ
nIRQ
Processor Clock
Power Management Controller
Wake Up
APB

25.3 I/O Line Description

Table 25-1. I/O Line Description

Pin Name	Pin Description	Type
FIQ	Fast Interrupt	Input
IRQ0 - IRQn	Interrupt 0 - Interrupt n	Input

25.4 Product Dependencies

25.4.1 I/O Lines

The interrupt signals FIQ and IRQ0 to IRQn are normally multiplexed through the PIO controllers. Depending on the features of the PIO controller used in the product, the pins must be programmed in accordance with their assigned interrupt function. This is not applicable when the PIO controller used in the product is transparent on the input path.

25.4.2 Power Management

The Advanced Interrupt Controller is continuously clocked. The Power Management Controller has no effect on the Advanced Interrupt Controller behavior.

The assertion of the Advanced Interrupt Controller outputs, either nIRQ or nFIQ, wakes up the ARM processor while it is in Idle Mode. The General Interrupt Mask feature enables the AIC to wake up the processor without asserting the interrupt line of the processor, thus providing synchronization of the processor on an event.

25.4.3 Interrupt Sources

The Interrupt Source 0 is always located at FIQ. If the product does not feature an FIQ pin, the Interrupt Source 0 cannot be used.

The Interrupt Source 1 is always located at System Interrupt. This is the result of the OR-wiring of the system peripheral interrupt lines, such as the System Timer, the Real Time Clock, the Power Management Controller and the Memory Controller. When a system interrupt occurs, the service routine must first distinguish the cause of the interrupt. This is performed by reading successively the status registers of the above mentioned system peripherals.

The interrupt sources 2 to 31 can either be connected to the interrupt outputs of an embedded user peripheral or to external interrupt lines. The external interrupt lines can be connected directly, or through the PIO Controller.

The PIO Controllers are considered as user peripherals in the scope of interrupt handling. Accordingly, the PIO Controller interrupt lines are connected to the Interrupt Sources 2 to 31.

The peripheral identification defined at the product level corresponds to the interrupt source number (as well as the bit number controlling the clock of the peripheral). Consequently, to simplify the description of the functional operations and the user interface, the interrupt sources are named FIQ, SYS, and PID2 to PID31.

25.5 Functional Description

25.5.1 Interrupt Source Control

25.5.1.1 Interrupt Source Mode

The Advanced Interrupt Controller independently programs each interrupt source. The SRCTYPE field of the corresponding AIC_SMR (Source Mode Register) selects the interrupt condition of each source.

The internal interrupt sources wired on the interrupt outputs of the embedded peripherals can be programmed either in level-sensitive mode or in edge-triggered mode. The active level of the internal interrupts is not important for the user.

The external interrupt sources can be programmed either in high level-sensitive or low level-sensitive modes, or in positive edge-triggered or negative edge-triggered modes.

25.5.1.2 Interrupt Source Enabling

Each interrupt source, including the FIQ in source 0, can be enabled or disabled by using the command registers; AIC_IECR (Interrupt Enable Command Register) and AIC_IDCR (Interrupt Disable Command Register). This set of registers conducts enabling or disabling in one instruction. The interrupt mask can be read in the AIC_IMR register. A disabled interrupt does not affect servicing of other interrupts.

25.5.1.3 Interrupt Clearing and Setting

All interrupt sources programmed to be edge-triggered (including the FIQ in source 0) can be individually set or cleared by writing respectively the AIC_ISCR and AIC_ICCR registers. Clearing or setting interrupt sources programmed in level-sensitive mode has no effect.

The clear operation is perfunctory, as the software must perform an action to reinitialize the “memorization” circuitry activated when the source is programmed in edge-triggered mode. However, the set operation is available for auto-test or software debug purposes. It can also be used to execute an AIC-implementation of a software interrupt.

The AIC features an automatic clear of the current interrupt when the AIC_IVR (Interrupt Vector Register) is read. Only the interrupt source being detected by the AIC as the current interrupt is affected by this operation. (See “Priority Controller” on page 247.) The automatic clear reduces the operations required by the interrupt service routine entry code to reading the AIC_IVR. Note that the automatic interrupt clear is disabled if the interrupt source has the Fast Forcing feature enabled as it is considered uniquely as a FIQ source. (For further details, See “Fast Forcing” on page 251.)

The automatic clear of the interrupt source 0 is performed when AIC_FVR is read.

25.5.1.4 Interrupt Status

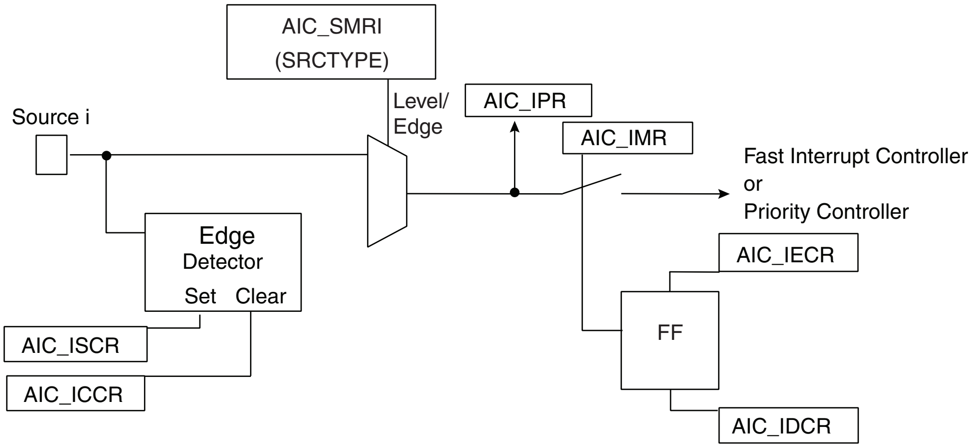
For each interrupt, the AIC operation originates in AIC_IPR (Interrupt Pending Register) and its mask in AIC_IMR (Interrupt Mask Register). AIC_IPR enables the actual activity of the sources, whether masked or not.

The AIC_ISR register reads the number of the current interrupt (see “Priority Controller” on page 247) and the register AIC_CISR gives an image of the signals nIRQ and nFIQ driven on the processor.

Each status referred to above can be used to optimize the interrupt handling of the systems.

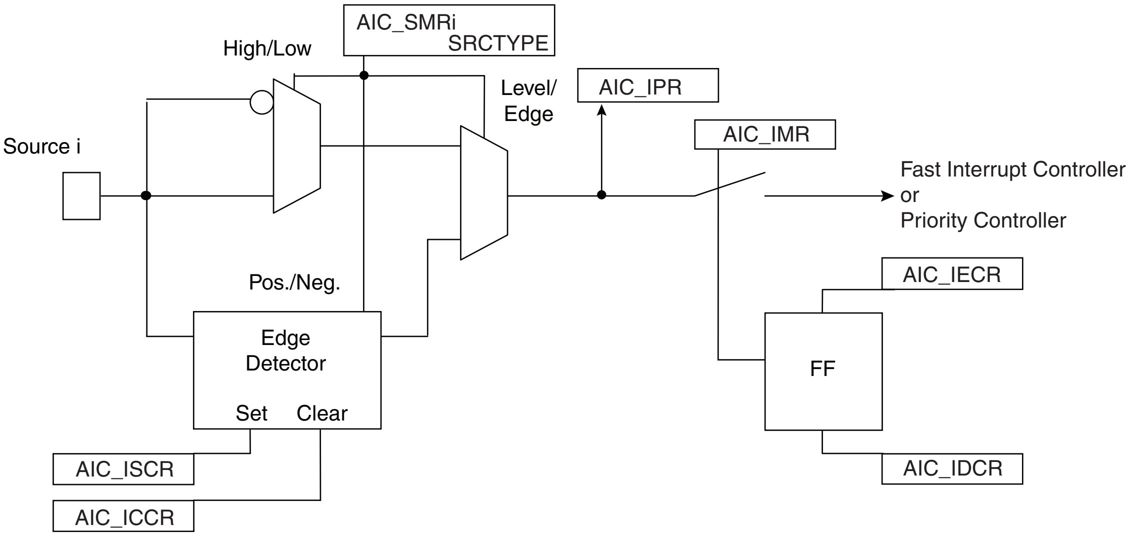
25.5.1.5 Internal Interrupt Source Input Stage

Figure 25-4. Internal Interrupt Source Input Stage


AIC_SMRI
(SRCTYPE)
Source i
Level/
Edge
AIC_IPR
AIC_IMR
Fast Interrupt Controller
or
Priority Controller
Edge
Detector
Set
Clear
AIC_IECR
AIC_ISCR
AIC_ICCR
FF
AIC_IDCR

25.5.1.6 External Interrupt Source Input Stage

Figure 25-5. External Interrupt Source Input Stage


AIC_SMRi
SRCTYPE
High/Low
Level/
Edge
AIC_IPR
AIC_IMR
Source i
Fast Interrupt Controller
or
Priority Controller
Pos./Neg.
AIC_IECR
Edge
Detector
Set
Clear
FF
AIC_ISCR
AIC_IDCR
AIC_ICCR

25.5.2 Interrupt Latencies

Global interrupt latencies depend on several parameters, including:

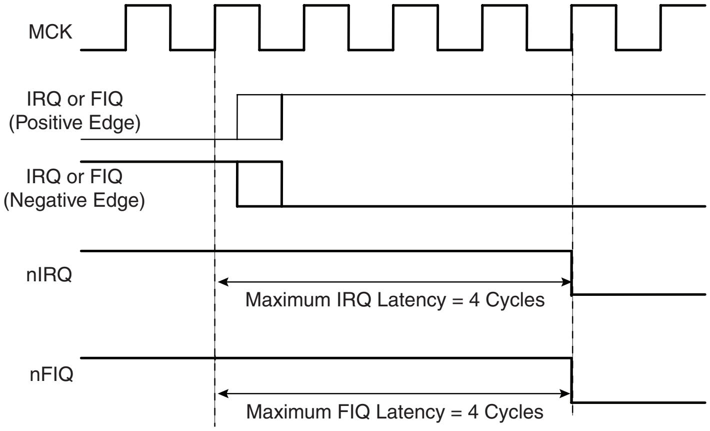
- The time the software masks the interrupts.
- Occurrence, either at the processor level or at the AIC level.
- The execution time of the instruction in progress when the interrupt occurs.
- The treatment of higher priority interrupts and the resynchronization of the hardware signals.

This section addresses only the hardware resynchronizations. It gives details of the latency times between the event on an external interrupt leading in a valid interrupt (edge or level) or the assertion of an internal interrupt source and the assertion of the nIRQ or nFIQ line on the processor. The resynchronization time depends on the programming of the interrupt source and on its type (internal or external). For the standard interrupt, resynchronization times are given assuming there is no higher priority in progress.

The PIO Controller multiplexing has no effect on the interrupt latencies of the external interrupt sources.

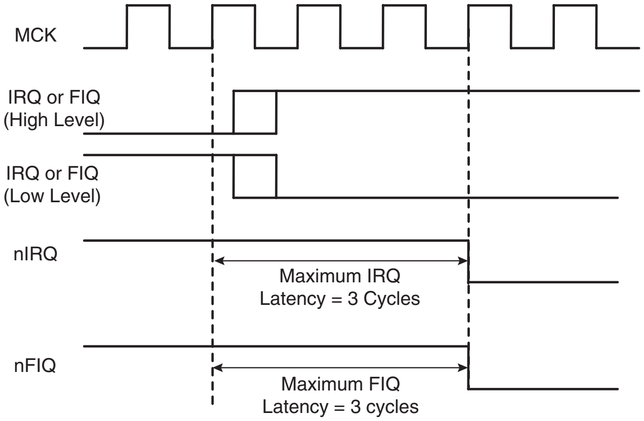
25.5.2.1 External Interrupt Edge Triggered Source

Figure 25-6. External Interrupt Edge Triggered Source


MCK
IRQ or FIQ (Positive Edge)
IRQ or FIQ (Negative Edge)
nIRQ
Maximum IRQ Latency = 4 Cycles
nFIQ
Maximum FIQ Latency = 4 Cycles

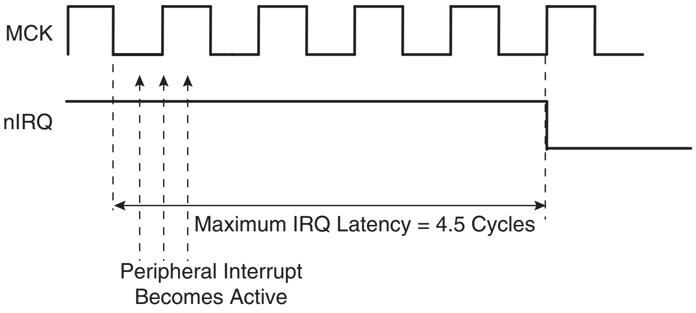
25.5.2.2 External Interrupt Level Sensitive Source

Figure 25-7. External Interrupt Level Sensitive Source


MCK
IRQ or FIQ (High Level)
IRQ or FIQ (Low Level)
nIRQ
Maximum IRQ Latency = 3 Cycles
nFIQ
Maximum FIQ Latency = 3 cycles

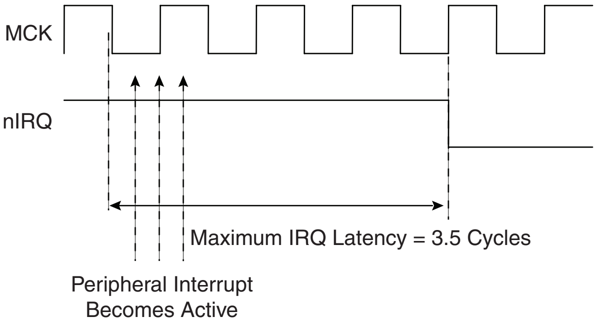
25.5.2.3 Internal Interrupt Edge Triggered Source

Figure 25-8. Internal Interrupt Edge Triggered Source



25.5.2.4 Internal Interrupt Level Sensitive Source

Figure 25-9. Internal Interrupt Level Sensitive Source



25.5.3 Normal Interrupt

25.5.3.1 Priority Controller

An 8-level priority controller drives the nIRQ line of the processor, depending on the interrupt conditions occurring on the interrupt sources 1 to 31 (except for those programmed in Fast Forcing).

Each interrupt source has a programmable priority level of 7 to 0, which is user-definable by writing the PRIOR field of the corresponding AIC_SMR (Source Mode Register). Level 7 is the highest priority and level 0 the lowest.

As soon as an interrupt condition occurs, as defined by the SRCTYPE field of the AIC_SMR (Source Mode Register), the nIRQ line is asserted. As a new interrupt condition might have happened on other interrupt sources since the nIRQ has been asserted, the priority controller determines the current interrupt at the time the AIC_IVR (Interrupt Vector Register) is read. **The read of AIC_IVR is the entry point of the interrupt handling** which allows the AIC to consider that the interrupt has been taken into account by the software.

The current priority level is defined as the priority level of the current interrupt.

If several interrupt sources of equal priority are pending and enabled when the AIC_IVR is read, the interrupt with the lowest interrupt source number is serviced first.

The nIRQ line can be asserted only if an interrupt condition occurs on an interrupt source with a higher priority. If an interrupt condition happens (or is pending) during the interrupt treatment in progress, it is delayed until the software indicates to the AIC the end of the current service by writing the AIC_EOICR (End of Interrupt Command Register). **The write of AIC_EOICR is the exit point of the interrupt handling**.

25.5.3.2 *Interrupt Nesting*

The priority controller utilizes interrupt nesting in order for the high priority interrupt to be handled during the service of lower priority interrupts. This requires the interrupt service routines of the lower interrupts to re-enable the interrupt at the processor level.

When an interrupt of a higher priority happens during an already occurring interrupt service routine, the nIRQ line is re-asserted. If the interrupt is enabled at the core level, the current execution is interrupted and the new interrupt service routine should read the AIC_IVR. At this time, the current interrupt number and its priority level are pushed into an embedded hardware stack, so that they are saved and restored when the higher priority interrupt servicing is finished and the AIC_EOICR is written.

The AIC is equipped with an 8-level wide hardware stack in order to support up to eight interrupt nestings pursuant to having eight priority levels.

25.5.3.3 *Interrupt Vectoring*

The interrupt handler addresses corresponding to each interrupt source can be stored in the registers AIC_SVR1 to AIC_SVR31 (Source Vector Register 1 to 31). When the processor reads AIC_IVR (Interrupt Vector Register), the value written into AIC_SVR corresponding to the current interrupt is returned.

This feature offers a way to branch in one single instruction to the handler corresponding to the current interrupt, as AIC_IVR is mapped at the absolute address 0xFFFF F100 and thus accessible from the ARM interrupt vector at address 0x0000 0018 through the following instruction:

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

When the processor executes this instruction, it loads the read value in AIC_IVR in its program counter, thus branching the execution on the correct interrupt handler.

This feature is often not used when the application is based on an operating system (either real time or not). Operating systems often have a single entry point for all the interrupts and the first task performed is to discern the source of the interrupt.

However, it is strongly recommended to port the operating system on AT91 products by supporting the interrupt vectoring. This can be performed by defining all the AIC_SVR of the interrupt source to be handled by the operating system at the address of its interrupt handler. When doing so, the interrupt vectoring permits a critical interrupt to transfer the execution on a specific very fast handler and not onto the operating system's general interrupt handler. This facilitates the support of hard real-time tasks (input/outputs of voice/audio buffers and software peripheral handling) to be handled efficiently and independently of the application running under an operating system.

25.5.4 Interrupt Handlers

This section gives an overview of the fast interrupt handling sequence when using the AIC. It is assumed that the programmer understands the architecture of the ARM processor, and especially the processor interrupt modes and the associated status bits.

It is assumed that:

1. The Advanced Interrupt Controller has been programmed, AIC_SVR registers are loaded with corresponding interrupt service routine addresses and interrupts are enabled.
2. The instruction at the ARM interrupt exception vector address is required to work with the vectoring

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

When nIRQ is asserted, if the bit "I" of CPSR is 0, the sequence is as follows:

1. The CPSR is stored in SPSR_irq, the current value of the Program Counter is loaded in the Interrupt link register (R14_irq) and the Program Counter (R15) is loaded with 0x18. In the following cycle during fetch at address 0x1C, the ARM core adjusts R14_irq, decrementing it by four.
2. The ARM core enters Interrupt mode, if it has not already done so.
3. When the instruction loaded at address 0x18 is executed, the program counter is loaded with the value read in AIC_IVR. Reading the AIC_IVR has the following effects:
   - Sets the current interrupt to be the pending and enabled interrupt with the highest priority. The current level is the priority level of the current interrupt.
   - De-asserts the nIRQ line on the processor. Even if vectoring is not used, AIC_IVR must be read in order to de-assert nIRQ.
   - Automatically clears the interrupt, if it has been programmed to be edge-triggered.
   - Pushes the current level and the current interrupt number on to the stack.
   - Returns the value written in the AIC_SVR corresponding to the current interrupt.
4. The previous step has the effect of branching to the corresponding interrupt service routine. This should start by saving the link register (R14_irq) and SPSR_IRQ. The link register must be decremented by four when it is saved if it is to be restored directly into the program counter at the end of the interrupt. For example, the instruction `SUB PC, LR, #4` may be used.
5. Further interrupts can then be unmasked by clearing the "I" bit in CPSR, allowing re-assertion of the nIRQ to be taken into account by the core. This can happen if an interrupt with a higher priority than the current interrupt occurs.
6. The interrupt handler can then proceed as required, saving the registers that will be used and restoring them at the end. During this phase, an interrupt of higher priority than the current level will restart the sequence from step 1.

Note: If the interrupt is programmed to be level sensitive, the source of the interrupt must be cleared during this phase.

7. The "I" bit in CPSR must be set in order to mask interrupts before exiting to ensure that the interrupt is completed in an orderly manner.
8. The End of Interrupt Command Register (AIC_EOICR) must be written in order to indicate to the AIC that the current interrupt is finished. This causes the current level to be popped from the stack, restoring the previous current level if one exists on the stack. If another interrupt is pending, with lower or equal priority than the old current level but with higher priority than the new current level, the nIRQ line is re-asserted, but the interrupt sequence does not immediately start because the "I" bit is set in the core. SPSR_irq is restored. Finally, the saved value of the link register is restored directly into the PC. This has 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 SPSR_irq.

Note: The “I” bit in SPSR is significant. If it is set, it indicates that the ARM core was on the verge of masking an interrupt when the mask instruction was interrupted. Hence, when SPSR is restored, the mask instruction is completed (interrupt is masked).

25.5.5 Fast Interrupt

25.5.5.1 Fast Interrupt Source

The interrupt source 0 is the only source which can raise a fast interrupt request to the processor except if fast forcing is used. The interrupt source 0 is generally connected to a FIQ pin of the product, either directly or through a PIO Controller.

25.5.5.2 Fast Interrupt Control

The fast interrupt logic of the AIC has no priority controller. The mode of interrupt source 0 is programmed with the AIC_SMR0 and the field PRIOR of this register is not used even if it reads what has been written. The field SRCTYPE of AIC_SMR0 enables programming the fast interrupt source to be positive-edge triggered or negative-edge triggered or high-level sensitive or low-level sensitive

Writing 0x1 in the AIC_IECR (Interrupt Enable Command Register) and AIC_IDCR (Interrupt Disable Command Register) respectively enables and disables the fast interrupt. The bit 0 of AIC_IMR (Interrupt Mask Register) indicates whether the fast interrupt is enabled or disabled.

25.5.5.3 Fast Interrupt Vectoring

The fast interrupt handler address can be stored in AIC_SVR0 (Source Vector Register 0). The value written into this register is returned when the processor reads AIC_FVR (Fast Vector Register). This offers a way to branch in one single instruction to the interrupt handler, as AIC_FVR is mapped at the absolute address 0xFFFF F104 and thus accessible from the ARM fast interrupt vector at address 0x0000 001C through the following instruction:

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

When the processor executes this instruction it loads the value read in AIC_FVR in its program counter, thus branching the execution on the fast interrupt handler. It also automatically performs the clear of the fast interrupt source if it is programmed in edge-triggered mode.

25.5.5.4 Fast Interrupt Handlers

This section gives an overview of the fast interrupt handling sequence when using the AIC. It is assumed that the programmer understands the architecture of the ARM processor, and especially the processor interrupt modes and associated status bits.

Assuming that:

1. The Advanced Interrupt Controller has been programmed, AIC_SVR0 is loaded with the fast interrupt service routine address, and the interrupt source 0 is enabled.
2. The Instruction at address 0x1C (FIQ exception vector address) is required to vector the fast interrupt:

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

3. The user does not need nested fast interrupts.

When nFIQ is asserted, if the bit “F” of CPSR is 0, the sequence is:

1. The CPSR is stored in SPSR_fiq, the current value of the program counter is loaded in the FIQ link register (R14_FIQ) and the program counter (R15) is loaded with 0x1C. In

the following cycle, during fetch at address 0x20, the ARM core adjusts R14_fiq, decrementing it by four.

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

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

Another way to handle the fast interrupt is to map the interrupt service routine at the address of the ARM vector 0x1C. This method does not use the vectoring, so that reading AIC_FVR must be performed at the very beginning of the handler operation. However, this method saves the execution of a branch instruction.

25.5.5.5 Fast Forcing

The Fast Forcing feature of the advanced interrupt controller provides redirection of any normal Interrupt source on the fast interrupt controller.

Fast Forcing is enabled or disabled by writing to the Fast Forcing Enable Register (AIC_FFER) and the Fast Forcing Disable Register (AIC_FFDR). Writing to these registers results in an update of the Fast Forcing Status Register (AIC_FFSR) that controls the feature for each internal or external interrupt source.

When Fast Forcing is disabled, the interrupt sources are handled as described in the previous pages.

When Fast Forcing is enabled, the edge/level programming and, in certain cases, edge detection of the interrupt source is still active but the source cannot trigger a normal interrupt to the processor and is not seen by the priority handler.

If the interrupt source is programmed in level-sensitive mode and an active level is sampled, Fast Forcing results in the assertion of the nFIQ line to the core.

If the interrupt source is programmed in edge-triggered mode and an active edge is detected, Fast Forcing results in the assertion of the nFIQ line to the core.

The Fast Forcing feature does not affect the Source 0 pending bit in the Interrupt Pending Register (AIC_IPR).

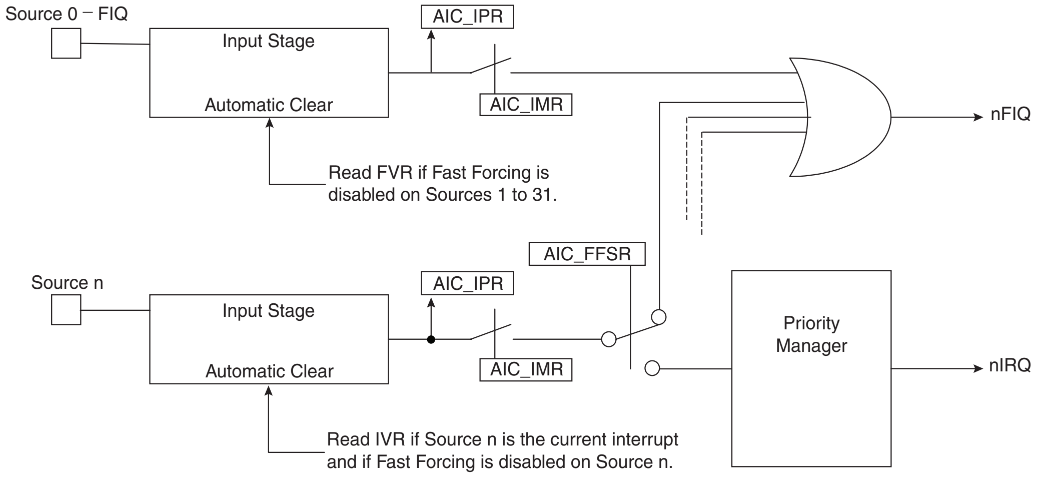
The FIQ Vector Register (AIC_FVR) reads the contents of the Source Vector Register 0 (AIC_SVR0), whatever the source of the fast interrupt may be. The read of the FVR does not clear the Source 0 when the fast forcing feature is used and the interrupt source should be cleared by writing to the Interrupt Clear Command Register (AIC_ICCR).

All enabled and pending interrupt sources that have the fast forcing feature enabled and that are programmed in edge-triggered mode must be cleared by writing to the Interrupt Clear Command Register. In doing so, they are cleared independently and thus lost interrupts are prevented.

The read of AIC_IVR does not clear the source that has the fast forcing feature enabled.

The source 0, reserved to the fast interrupt, continues operating normally and becomes one of the Fast Interrupt sources.

Figure 25-10. Fast Forcing


Source 0 – FIQ
Input Stage
Automatic Clear
AIC_IPR
AIC_IMR
nFIQ
Read FVR if Fast Forcing is disabled on Sources 1 to 31.
AIC_FFSR
Source n
Input Stage
Automatic Clear
AIC_IPR
AIC_IMR
Priority Manager
nIRQ
Read IVR if Source n is the current interrupt and if Fast Forcing is disabled on Source n.

25.5.6 Protect Mode

The Protect Mode permits reading the Interrupt Vector Register without performing the associated automatic operations. This is necessary when working with a debug system. When a debugger, working either with a Debug Monitor or the ARM processor's ICE, stops the applications and updates the opened windows, it might read the AIC User Interface and thus the IVR. This has undesirable consequences:

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

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

This is avoided by using the Protect Mode. Writing PROT in AIC_DCR (Debug Control Register) at 0x1 enables the Protect Mode.

When the Protect Mode is enabled, the AIC performs interrupt stacking only when a write access is performed on the AIC_IVR. Therefore, the Interrupt Service Routines must write (arbitrary data) to the AIC_IVR just after reading it. The new context of the AIC, including the value of the Interrupt Status Register (AIC_ISR), is updated with the current interrupt only when AIC_IVR is written.

An AIC_IVR read on its own (e.g., by a debugger), modifies neither the AIC context nor the AIC_ISR. Extra AIC_IVR reads perform the same operations. However, it is recommended to not stop the processor between the read and the write of AIC_IVR of the interrupt service routine to make sure the debugger does not modify the AIC context.

To summarize, in normal operating mode, the read of AIC_IVR performs the following operations within the AIC:

1. Calculates active interrupt (higher than current or spurious).
2. Determines and returns the vector of the active interrupt.
3. Memorizes the interrupt.
4. Pushes the current priority level onto the internal stack.
5. Acknowledges the interrupt.

However, while the Protect Mode is activated, only operations 1 to 3 are performed when AIC_IVR is read. Operations 4 and 5 are only performed by the AIC when AIC_IVR is written.

Software that has been written and debugged using the Protect Mode runs correctly in Normal Mode without modification. However, in Normal Mode the AIC_IVR write has no effect and can be removed to optimize the code.

25.5.7 Spurious Interrupt

The Advanced Interrupt Controller features protection against spurious interrupts. A spurious interrupt is defined as being the assertion of an interrupt source long enough for the AIC to assert the nIRQ, but no longer present when AIC_IVR is read. This is most prone to occur when:

- An external interrupt source is programmed in level-sensitive mode and an active level occurs for only a short time.
- An internal interrupt source is programmed in level sensitive and the output signal of the corresponding embedded peripheral is activated for a short time. (As in the case for the Watchdog.)
- An interrupt occurs just a few cycles before the software begins to mask it, thus resulting in a pulse on the interrupt source.

The AIC detects a spurious interrupt at the time the AIC_IVR is read while no enabled interrupt source is pending. When this happens, the AIC returns the value stored by the programmer in AIC_SPU (Spurious Vector Register). The programmer must store the address of a spurious interrupt handler in AIC_SPU as part of the application, to enable an as fast as possible return to the normal execution flow. This handler writes in AIC_EOICR and performs a return from interrupt.

25.5.8 General Interrupt Mask

The AIC features a General Interrupt Mask bit to prevent interrupts from reaching the processor. Both the nIRQ and the nFIQ lines are driven to their inactive state if the bit GMSK in AIC_DCR (Debug Control Register) is set. However, this mask does not prevent waking up the processor if it has entered Idle Mode. This function facilitates synchronizing the processor on a next event and, as soon as the event occurs, performs subsequent operations without having to handle an interrupt. It is strongly recommended to use this mask with caution.

25.6 Advanced Interrupt Controller (AIC) User Interface

25.6.1 Base Address

The AIC is mapped at the address **0xFFFF F000**. It has a total 4-Kbyte addressing space. This permits the vectoring feature, as the PC-relative load/store instructions of the ARM processor support only a ± 4-Kbyte offset.

25.6.2 Register Mapping

Table 25-2. Register Mapping

Offset	Register	Name	Access	Reset Value
0000	Source Mode Register 0	AIC_SMR0	Read/Write	0x0
0x04	Source Mode Register 1	AIC_SMR1	Read/Write	0x0
---	---	---	---	---
0x7C	Source Mode Register 31	AIC_SMR31	Read/Write	0x0
0x80	Source Vector Register 0	AIC_SVR0	Read/Write	0x0
0x84	Source Vector Register 1	AIC_SVR1	Read/Write	0x0
---	---	---	---	---
0xFC	Source Vector Register 31	AIC_SVR31	Read/Write	0x0
0x100	Interrupt Vector Register	AIC_IVR	Read-only	0x0
0x104	FIQ Interrupt Vector Register	AIC_FVR	Read-only	0x0
0x108	Interrupt Status Register	AIC_ISR	Read-only	0x0
0x10C	Interrupt Pending Register(2)	AIC_IPR	Read-only	0x0(1)
0x110	Interrupt Mask Register(2)	AIC_IMR	Read-only	0x0
0x114	Core Interrupt Status Register	AIC_CISR	Read-only	0x0
0x118	Reserved	---	---	---
0x11C	Reserved	---	---	---
0x120	Interrupt Enable Command Register(2)	AIC_IECR	Write-only	---
0x124	Interrupt Disable Command Register(2)	AIC_IDCR	Write-only	---
0x128	Interrupt Clear Command Register(2)	AIC_ICCR	Write-only	---
0x12C	Interrupt Set Command Register(2)	AIC_ISCR	Write-only	---
0x130	End of Interrupt Command Register	AIC_EOICR	Write-only	---
0x134	Spurious Interrupt Vector Register	AIC_SPU	Read/Write	0x0
0x138	Debug Control Register	AIC_DCR	Read/Write	0x0
0x13C	Reserved	---	---	---
0x140	Fast Forcing Enable Register(2)	AIC_FFER	Write-only	---
0x144	Fast Forcing Disable Register(2)	AIC_FFDR	Write-only	---
0x148	Fast Forcing Status Register(2)	AIC_FFSR	Read-only	0x0

Notes: 1. The reset value of this register depends on the level of the external interrupt source. All other sources are cleared at reset, thus not pending.

2. PID2...PID31 bit fields refer to the identifiers as defined in the Peripheral Identifiers Section of the product datasheet.

25.6.3 AIC Source Mode Register

Register Name: AIC_SMR0..AIC_SMR31

Access Type: Read/Write

Reset Value: 0x0

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	SRCTYPE		–	–	PRIOR		

- **PRIOR: Priority Level**

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

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

The priority level is not used for the FIQ in the related SMR register AIC_SMRx.

- **SRCTYPE: Interrupt Source Type**

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

Table 25-3.

SRCTYPE		Internal Interrupt Sources	External Interrupt Sources
0	0	High level Sensitive	Low level Sensitive
0	1	Positive edge triggered	Negative edge triggered
1	0	High level Sensitive	High level Sensitive
1	1	Positive edge triggered	Positive edge triggered

25.6.4 AIC Source Vector Register

Register Name: AIC_SVR0..AIC_SVR31

Access Type: Read/Write

Reset Value: 0x0

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

- **VECTOR: Source Vector**

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

25.6.5 AIC Interrupt Vector Register

Register Name: AIC_IVR

Access Type: Read-only

Reset Value: 0x0

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

- **IRQV: Interrupt Vector Register**

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

The Source Vector Register is indexed using the current interrupt number when the Interrupt Vector Register is read.

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

25.6.6 AIC FIQ Vector Register

Register Name: AIC_FVR

Access Type: Read-only

Reset Value: 0x0

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

- **FIQV: FIQ Vector Register**

The FIQ Vector Register contains the vector programmed by the user in the Source Vector Register 0. When there is no fast interrupt, the FIQ Vector Register reads the value stored in AIC_SPU.

25.6.7 AIC Interrupt Status Register

Register Name: AIC_ISR

Access Type: Read-only

Reset Value: 0x0

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

- **IRQID: Current Interrupt Identifier**

The Interrupt Status Register returns the current interrupt source number.

25.6.8 AIC Interrupt Pending Register

Register Name: AIC_IPR

Access Type: Read-only

Reset Value: 0x0

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Pending**

0 = Corresponding interrupt is not pending.

1 = Corresponding interrupt is pending.

25.6.9 AIC Interrupt Mask Register

Register Name: AIC_IMR

Access Type: Read-only

Reset Value: 0x0

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Mask**

0 = Corresponding interrupt is disabled.

1 = Corresponding interrupt is enabled.

25.6.10 AIC Core Interrupt Status Register

Register Name: AIC_CISR

Access Type: Read-only

Reset Value: 0x0

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

- **NFIQ: NFIQ Status**

0 = nFIQ line is deactivated.

1 = nFIQ line is active.

- **NIRQ: NIRQ Status**

0 = nIRQ line is deactivated.

1 = nIRQ line is active.

25.6.11 AIC Interrupt Enable Command Register

Register Name: AIC_IECR

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID3: Interrupt Enable**

0 = No effect.

1 = Enables corresponding interrupt.

25.6.12 AIC Interrupt Disable Command Register

Register Name: AIC_IDCR

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24

23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16

15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8

7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Disable**

0 = No effect.

1 = Disables corresponding interrupt.

25.6.13 AIC Interrupt Clear Command Register

Register Name: AIC_ICCR

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24

23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16

15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8

7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Clear**

0 = No effect.

1 = Clears corresponding interrupt.

25.6.14 AIC Interrupt Set Command Register

Register Name: AIC_ISCR

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Set**

0 = No effect.

1 = Sets corresponding interrupt.

25.6.15 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.

25.6.16 AIC Spurious Interrupt Vector Register

Register Name: AIC_SPU

Access Type: Read/Write

Reset Value: 0x0

31	30	29	28	27	26	25	24
SIQV							

23	22	21	20	19	18	17	16
SIQV							

15	14	13	12	11	10	9	8
SIQV							

7	6	5	4	3	2	1	0
SIQV							

- **SIQV: Spurious Interrupt Vector Register**

The user may store the address of a spurious interrupt handler in this register. The written value is returned in AIC_IVR in case of a spurious interrupt and in AIC_FVR in case of a spurious fast interrupt.

25.6.17 AIC Debug Control Register

Register Name: AIC_DCR

Access Type: Read/Write

Reset Value: 0x0

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	GMSK	PROT

- **PROT: Protection Mode**

0 = The Protection Mode is disabled.

1 = The Protection Mode is enabled.

- **GMSK: General Mask**

0 = The nIRQ and nFIQ lines are normally controlled by the AIC.

1 = The nIRQ and nFIQ lines are tied to their inactive state.

25.6.18 AIC Fast Forcing Enable Register

Register Name: AIC_FFER

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

- **SYS, PID2-PID31: Fast Forcing Enable**

0 = No effect.

1 = Enables the fast forcing feature on the corresponding interrupt.

25.6.19 AIC Fast Forcing Disable Register

Register Name: AIC_FFDR

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

- **SYS, PID2-PID31: Fast Forcing Disable**

0 = No effect.

1 = Disables the Fast Forcing feature on the corresponding interrupt.

25.6.20 AIC Fast Forcing Status Register

Register Name: AIC_FFSR

Access Type: Read-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

- **SYS, PID2-PID31: Fast Forcing Status**

0 = The Fast Forcing feature is disabled on the corresponding interrupt.

1 = The Fast Forcing feature is enabled on the corresponding interrupt.

26. Debug Unit (DBGU)

26.1 Description

The Debug Unit provides a single entry point from the processor for access to all the debug capabilities of Atmel's ARM-based systems.

The Debug Unit features a two-pin UART that can be used for several debug and trace purposes and offers an ideal medium for in-situ programming solutions and debug monitor communications. Moreover, the association with two peripheral data controller channels permits packet handling for these tasks with processor time reduced to a minimum.

The Debug Unit also makes the Debug Communication Channel (DCC) signals provided by the In-circuit Emulator of the ARM processor visible to the software. These signals indicate the status of the DCC read and write registers and generate an interrupt to the ARM processor, making possible the handling of the DCC under interrupt control.

Chip Identifier registers permit recognition of the device and its revision. These registers inform as to the sizes and types of the on-chip memories, as well as the set of embedded peripherals.

Finally, the Debug Unit features a Force NTRST capability that enables the software to decide whether to prevent access to the system via the In-circuit Emulator. This permits protection of the code, stored in ROM.

26.2 Block Diagram

Figure 26-1. Debug Unit Functional Block Diagram

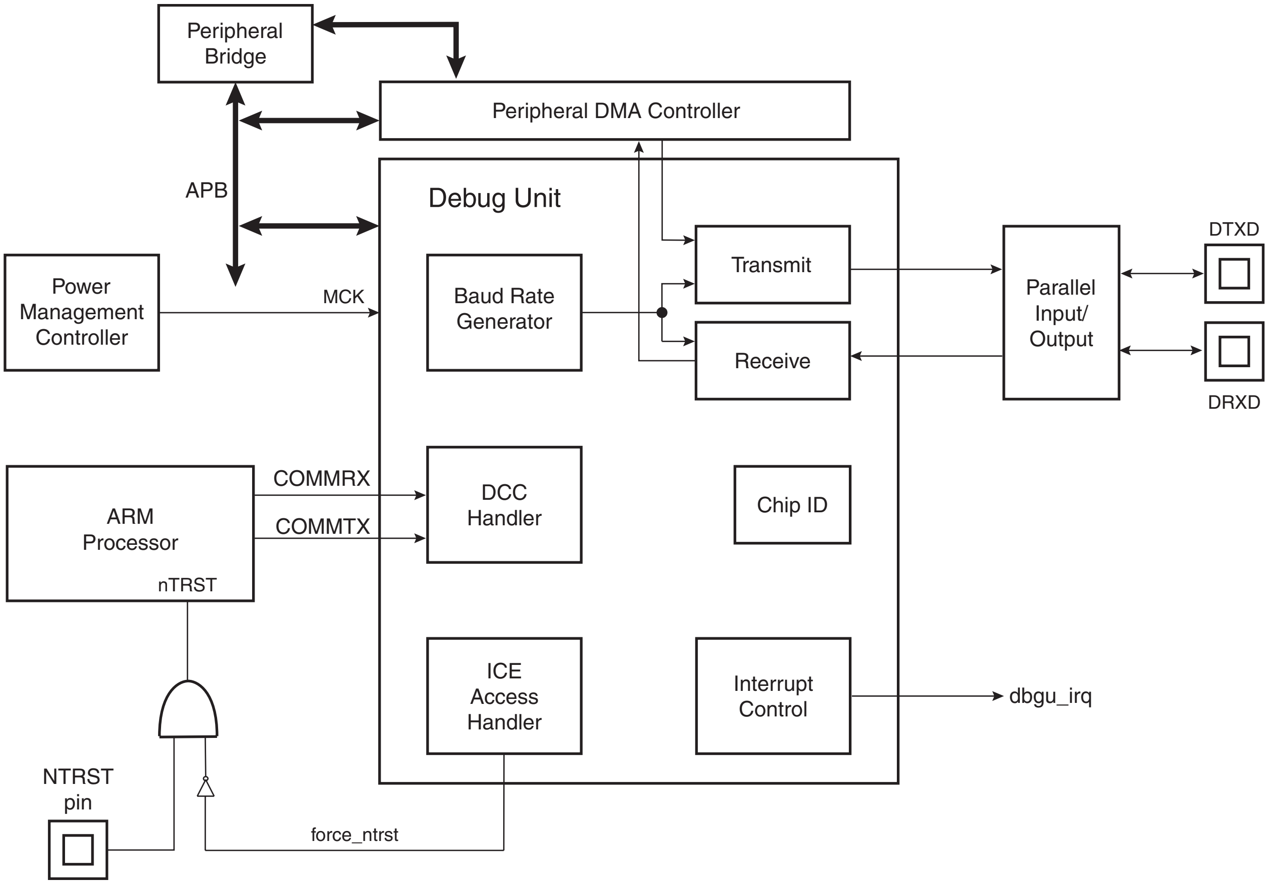
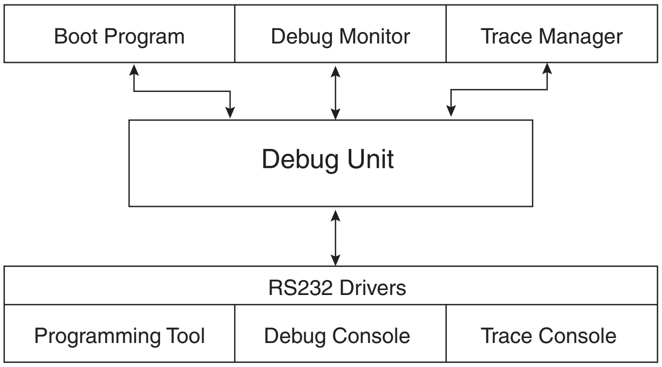

Peripheral Bridge
Peripheral DMA Controller
APB
Debug Unit
Transmit
Power Management Controller
MCK
Baud Rate Generator
Parallel Input/ Output
DTXD
Receive
DRXD
ARM Processor
COMMRX
COMMTX
DCC Handler
Chip ID
nTRST
ICE Access Handler
Interrupt Control
dbgu_irq
NTRST pin
force_ntrst

Table 26-1. Debug Unit Pin Description

Pin Name	Description	Type
DRXD	Debug Receive Data	Input
DTXD	Debug Transmit Data	Output

Figure 26-2. Debug Unit Application Example


Boot Program
Debug Monitor
Trace Manager
Debug Unit
RS232 Drivers
Programming Tool
Debug Console
Trace Console

26.3 Product Dependencies

26.3.1 I/O Lines

Depending on product integration, the Debug Unit pins may be multiplexed with PIO lines. In this case, the programmer must first configure the corresponding PIO Controller to enable I/O lines operations of the Debug Unit.

26.3.2 Power Management

Depending on product integration, the Debug Unit clock may be controllable through the Power Management Controller. In this case, the programmer must first configure the PMC to enable the Debug Unit clock. Usually, the peripheral identifier used for this purpose is 1.

26.3.3 Interrupt Source

Depending on product integration, the Debug Unit interrupt line is connected to one of the interrupt sources of the Advanced Interrupt Controller. Interrupt handling requires programming of the AIC before configuring the Debug Unit. Usually, the Debug Unit interrupt line connects to the interrupt source 1 of the AIC, which may be shared with the real-time clock, the system timer interrupt lines and other system peripheral interrupts, as shown in Figure 26-1. This sharing requires the programmer to determine the source of the interrupt when the source 1 is triggered.

26.4 UART Operations

The Debug Unit operates as a UART, (asynchronous mode only) and supports only 8-bit character handling (with parity). It has no clock pin.

The Debug Unit's UART is made up of a receiver and a transmitter that operate independently, and a common baud rate generator. Receiver timeout and transmitter time guard are not implemented. However, all the implemented features are compatible with those of a standard USART.

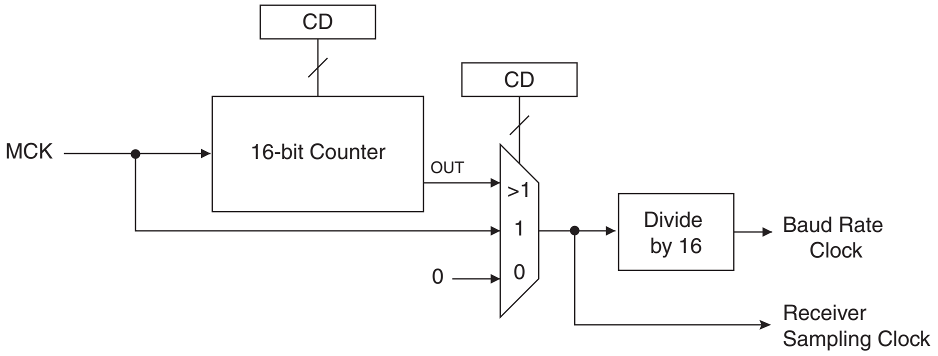
26.4.1 Baud Rate Generator

The baud rate generator provides the bit period clock named baud rate clock to both the receiver and the transmitter.

The baud rate clock is the master clock divided by 16 times the value (CD) written in DBGU_BRGR (Baud Rate Generator Register). If DBGU_BRGR is set to 0, the baud rate clock is disabled and the Debug Unit's UART remains inactive. The maximum allowable baud rate is Master Clock divided by 16. The minimum allowable baud rate is Master Clock divided by (16 x 65536).

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

Figure 26-3. Baud Rate Generator



26.4.2 Receiver

26.4.2.1 Receiver Reset, Enable and Disable

After device reset, the Debug Unit receiver is disabled and must be enabled before being used. The receiver can be enabled by writing the control register DBGU_CR with the bit RXEN at 1. At this command, the receiver starts looking for a start bit.

The programmer can disable the receiver by writing DBGU_CR with the bit RXDIS at 1. If the receiver is waiting for a start bit, it is immediately stopped. However, if the receiver has already detected a start bit and is receiving the data, it waits for the stop bit before actually stopping its operation.

The programmer can also put the receiver in its reset state by writing DBGU_CR with the bit RSTRX at 1. In doing so, the receiver immediately stops its current operations and is disabled, whatever its current state. If RSTRX is applied when data is being processed, this data is lost.

26.4.3 Start Detection and Data Sampling

The Debug Unit only supports asynchronous operations, and this affects only its receiver. The Debug Unit receiver detects the start of a received character by sampling the DRXD signal until it detects a valid start bit. A low level (space) on DRXD is interpreted as a valid start bit if it is detected for more than 7 cycles of the sampling clock, which is 16 times the baud rate. Hence, a space that is longer than 7/16 of the bit period is detected as a valid start bit. A space which is 7/16 of a bit period or shorter is ignored and the receiver continues to wait for a valid start bit.

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

Each subsequent bit is sampled 16 cycles (1-bit period) after the previous one.

Figure 26-4. Start Bit Detection

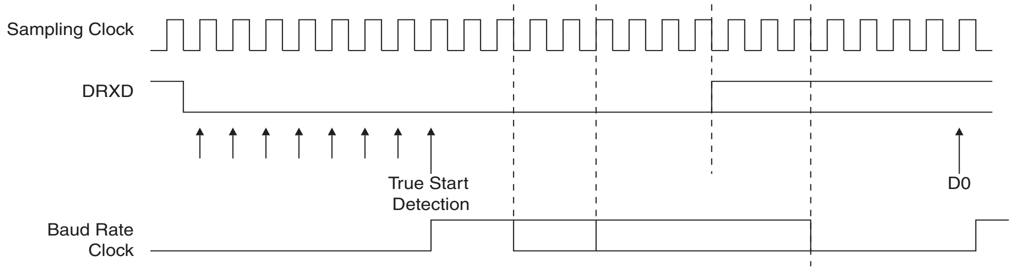
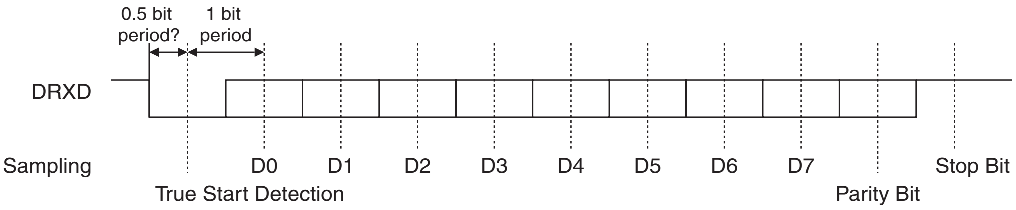

Sampling Clock
DRXD
True Start
Detection
D0
Baud Rate
Clock

Figure 26-5. Character Reception

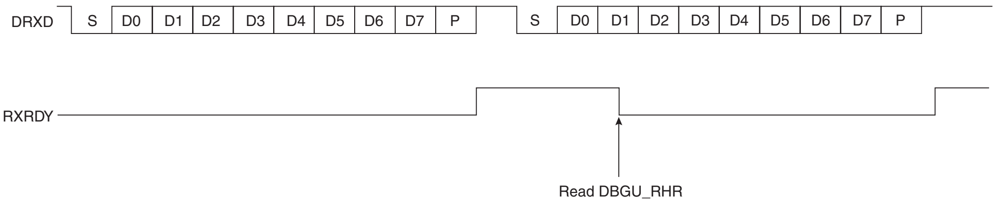
Example: 8-bit, parity enabled 1 stop


0.5 bit
period?
1 bit
period
DRXD
Sampling
D0
D1
D2
D3
D4
D5
D6
D7
Stop Bit
True Start Detection
Parity Bit

26.4.3.1 *Receiver Ready*

When a complete character is received, it is transferred to the DBGU_RHR and the RXRDY status bit in DBGU_SR (Status Register) is set. The bit RXRDY is automatically cleared when the receive holding register DBGU_RHR is read.

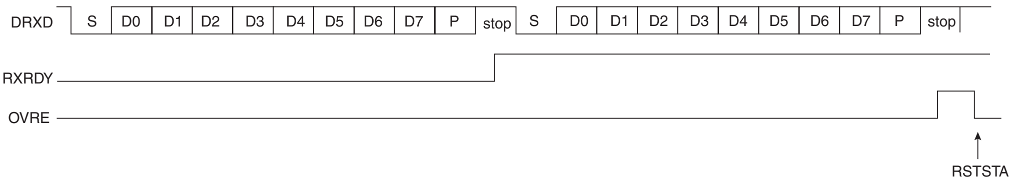
Figure 26-6. Receiver Ready


DRXD
S
D0
D1
D2
D3
D4
D5
D6
D7
P
S
D0
D1
D2
D3
D4
D5
D6
D7
P
RXRDY
Read DBGU_RHR

26.4.3.2 *Receiver Overrun*

If DBGU_RHR has not been read by the software (or the Peripheral Data Controller) since the last transfer, the RXRDY bit is still set and a new character is received, the OVRE status bit in DBGU_SR is set. OVRE is cleared when the software writes the control register DBGU_CR with the bit RSTSTA (Reset Status) at 1.

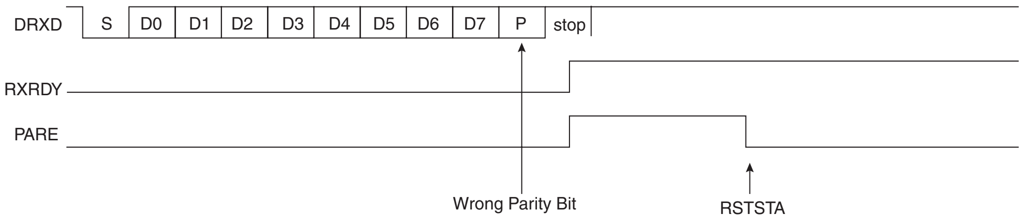
Figure 26-7. Receiver Overrun


DRXD
S
D0
D1
D2
D3
D4
D5
D6
D7
P
stop
S
D0
D1
D2
D3
D4
D5
D6
D7
P
stop
RXRDY
OVRE
RSTSTA

26.4.3.3 *Parity Error*

Each time a character is received, the receiver calculates the parity of the received data bits, in accordance with the field PAR in DBGU_MR. It then compares the result with the received parity bit. If different, the parity error bit PARE in DBGU_SR is set at the same time the RXRDY is set. The parity bit is cleared when the control register DBGU_CR is written with the bit RSTSTA (Reset Status) at 1. If a new character is received before the reset status command is written, the PARE bit remains at 1.

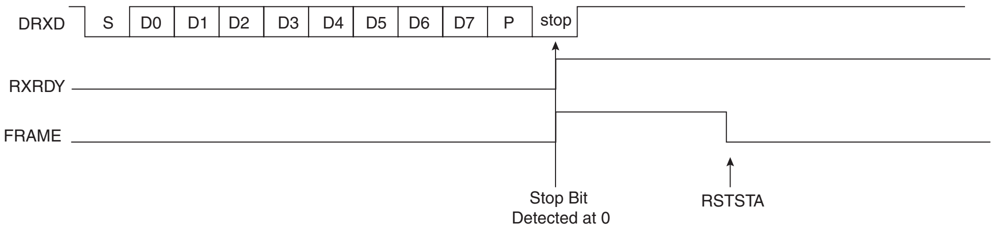
Figure 26-8. Parity Error



26.4.3.4 *Receiver Framing Error*

When a start bit is detected, it generates a character reception when all the data bits have been sampled. The stop bit is also sampled and when it is detected at 0, the FRAME (Framing Error) bit in DBGU_SR is set at the same time the RXRDY bit is set. The bit FRAME remains high until the control register DBGU_CR is written with the bit RSTSTA at 1.

Figure 26-9. Receiver Framing Error



26.4.4 Transmitter

26.4.4.1 *Transmitter Reset, Enable and Disable*

After device reset, the Debug Unit transmitter is disabled and it must be enabled before being used. The transmitter is enabled by writing the control register DBGU_CR with the bit TXEN at 1. From this command, the transmitter waits for a character to be written in the Transmit Holding Register DBGU_THR before actually starting the transmission.

The programmer can disable the transmitter by writing DBGU_CR with the bit TXDIS at 1. If the transmitter is not operating, it is immediately stopped. However, if a character is being processed into the Shift Register and/or a character has been written in the Transmit Holding Register, the characters are completed before the transmitter is actually stopped.

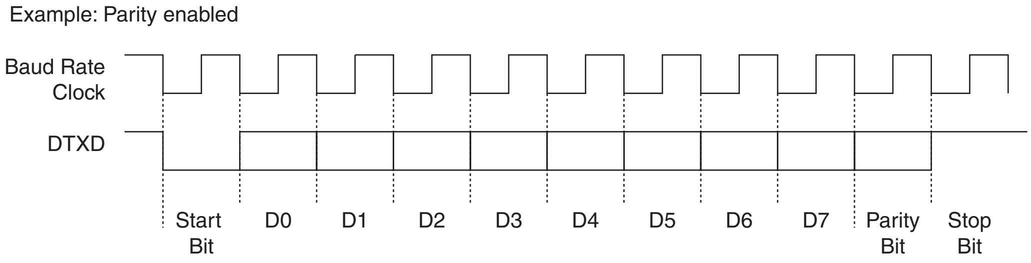
The programmer can also put the transmitter in its reset state by writing the DBGU_CR with the bit RSTTX at 1. This immediately stops the transmitter, whether or not it is processing characters.

26.4.4.2 *Transmit Format*

The Debug Unit transmitter drives the pin DTXD at the baud rate clock speed. The line is driven depending on the format defined in the Mode Register and the data stored in the Shift Register. One start bit at level 0, then the 8 data bits, from the lowest to the highest bit, one optional parity bit and one stop bit at 1 are consecutively shifted out as shown on the following figure. The field

PARE in the mode register DBGU_MR defines whether or not a parity bit is shifted out. When a parity bit is enabled, it can be selected between an odd parity, an even parity, or a fixed space or mark bit.

Figure 26-10. Character Transmission

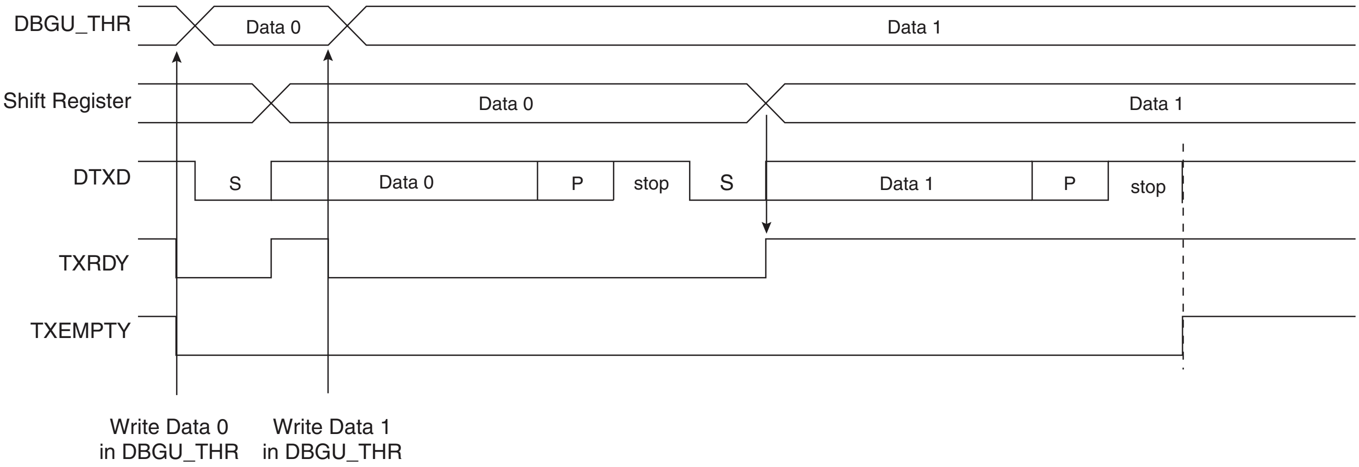

Example: Parity enabled
Baud Rate Clock
DTXD
Start Bit
D0
D1
D2
D3
D4
D5
D6
D7
Parity Bit
Stop Bit

26.4.4.3 *Transmitter Control*

When the transmitter is enabled, the bit TXRDY (Transmitter Ready) is set in the status register DBGU_SR. The transmission starts when the programmer writes in the Transmit Holding Register DBGU_THR, and after the written character is transferred from DBGU_THR to the Shift Register. The bit TXRDY remains high until a second character is written in DBGU_THR. As soon as the first character is completed, the last character written in DBGU_THR is transferred into the shift register and TXRDY rises again, showing that the holding register is empty.

When both the Shift Register and the DBGU_THR are empty, i.e., all the characters written in DBGU_THR have been processed, the bit TXEMPTY rises after the last stop bit has been completed.

Figure 26-11. Transmitter Control


DBGU_THR
Data 0
Data 1
Shift Register
Data 0
Data 1
DTXD
S
Data 0
P
stop
S
Data 1
P
stop
TXRDY
TXEMPTY
Write Data 0 in DBGU_THR
Write Data 1 in DBGU_THR

26.4.5 Peripheral Data Controller

Both the receiver and the transmitter of the Debug Unit's UART are generally connected to a Peripheral Data Controller (PDC) channel.

The peripheral data controller channels are programmed via registers that are mapped within the Debug Unit user interface from the offset 0x100. The status bits are reported in the Debug Unit status register DBGU_SR and can generate an interrupt.

The RXRDY bit triggers the PDC channel data transfer of the receiver. This results in a read of the data in DBGU_RHR. The TXRDY bit triggers the PDC channel data transfer of the transmitter. This results in a write of a data in DBGU_THR.

26.4.6 Test Modes

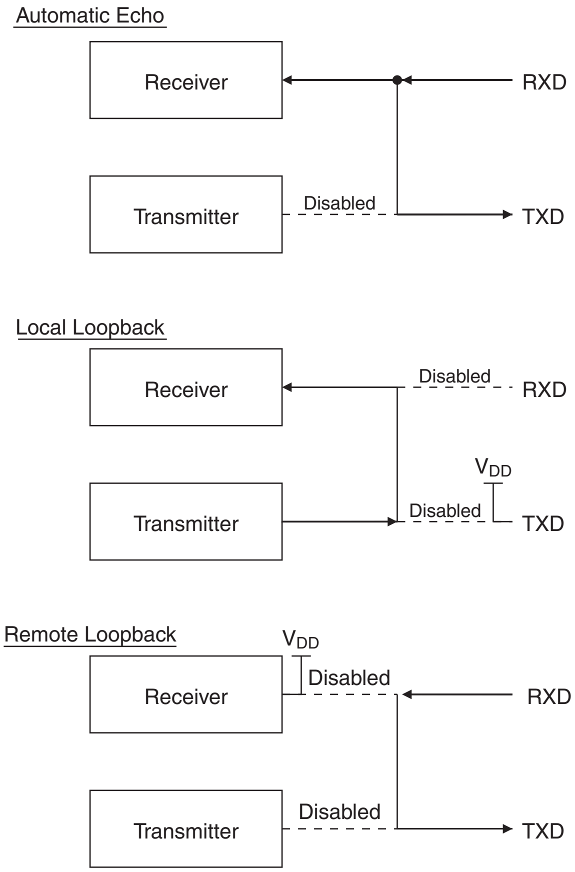
The Debug Unit supports three tests modes. These modes of operation are programmed by using the field CHMODE (Channel Mode) in the mode register DBGU_MR.

The Automatic Echo mode allows bit-by-bit retransmission. When a bit is received on the DRXD line, it is sent to the DTXD line. The transmitter operates normally, but has no effect on the DTXD line.

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

The Remote Loopback mode directly connects the DRXD pin to the DTXD line. The transmitter and the receiver are disabled and have no effect. This mode allows a bit-by-bit retransmission.

Figure 26-12. Test Modes


Automatic Echo
Receiver
RXD
Transmitter
Disabled
TXD
Local Loopback
Receiver
Disabled
RXD
VDD
Transmitter
Disabled
TXD
Remote Loopback
VDD
Receiver
Disabled
RXD
Transmitter
Disabled
TXD

26.4.7 Debug Communication Channel Support

The Debug Unit handles the signals COMMRX and COMMTX that come from the Debug Communication Channel of the ARM Processor and are driven by the In-circuit Emulator.

The Debug Communication Channel contains two registers that are accessible through the ICE Breaker on the JTAG side and through the coprocessor 0 on the ARM Processor side.

As a reminder, the following instructions are used to read and write the Debug Communication Channel:

```
MRC   p14, 0, Rd, c1, c0, 0
```

Returns the debug communication data read register into Rd

```
MCR   p14, 0, Rd, c1, c0, 0
```

Writes the value in Rd to the debug communication data write register.

The bits COMMRX and COMMTX, which indicate, respectively, that the read register has been written by the debugger but not yet read by the processor, and that the write register has been written by the processor and not yet read by the debugger, are wired on the two highest bits of the status register DBGU_SR. These bits can generate an interrupt. This feature permits handling under interrupt a debug link between a debug monitor running on the target system and a debugger.

26.4.8 Chip Identifier

The Debug Unit features two chip identifier registers, DBGU_CIDR (Chip ID Register) and DBGU_EXID (Extension ID). Both registers contain a hard-wired value that is read-only. The first register contains the following fields:

- EXT - shows the use of the extension identifier register
- NVPTYP and NVPSIZ - identifies the type of embedded non-volatile memory and its size
- ARCH - identifies the set of embedded peripherals
- SRAMSIZ - indicates the size of the embedded SRAM
- EPROC - indicates the embedded ARM processor
- VERSION - gives the revision of the silicon

The second register is device-dependent and reads 0 if the bit EXT is 0.

26.5 ICE Access Prevention

The Debug Unit allows blockage of access to the system through the ARM processor's ICE interface. This feature is implemented via the register Force NTRST (DBGU_FNR), that allows assertion of the NTRST signal of the ICE Interface. Writing the bit FNTRST (Force NTRST) to 1 in this register prevents any activity on the TAP controller.

On standard devices, the bit FNTRST resets to 0 and thus does not prevent ICE access.

This feature is especially useful on custom ROM devices for customers who do not want their on-chip code to be visible.

26.6 Debug Unit User Interface

Table 26-2. Debug Unit Memory Map

Offset	Register	Name	Access	Reset Value
0x0000	Control Register	DBGU_CR	Write-only	–
0x0004	Mode Register	DBGU_MR	Read/Write	0x0
0x0008	Interrupt Enable Register	DBGU_IER	Write-only	–
0x000C	Interrupt Disable Register	DBGU_IDR	Write-only	–
0x0010	Interrupt Mask Register	DBGU_IMR	Read-only	0x0
0x0014	Status Register	DBGU_SR	Read-only	–
0x0018	Receive Holding Register	DBGU_RHR	Read-only	0x0
0x001C	Transmit Holding Register	DBGU_THR	Write-only	–
0x0020	Baud Rate Generator Register	DBGU_BRGR	Read/Write	0x0
0x0024 - 0x003C	Reserved	–	–	–
0x0040	Chip ID Register	DBGU_CIDR	Read-only	–
0x0044	Chip ID Extension Register	DBGU_EXID	Read-only	–
0x0048	Force NTRST Register	DBGU_FNR	Read/Write	0x0
0x004C - 0x00FC	Reserved	–	–	–
0x0100 - 0x0124	PDC Area	–	–	–

26.6.1 Debug Unit Control Register

Name: DBGU_CR

Access Type: Write-only

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

- **RSTRX: Reset Receiver**

0 = No effect.

1 = The receiver logic is reset and disabled. If a character is being received, the reception is aborted.

- **RSTTX: Reset Transmitter**

0 = No effect.

1 = The transmitter logic is reset and disabled. If a character is being transmitted, the transmission is aborted.

- **RXEN: Receiver Enable**

0 = No effect.

1 = The receiver is enabled if RXDIS is 0.

- **RXDIS: Receiver Disable**

0 = No effect.

1 = The receiver is disabled. If a character is being processed and RSTRX is not set, the character is completed before the receiver is stopped.

- **TXEN: Transmitter Enable**

0 = No effect.

1 = The transmitter is enabled if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0 = No effect.

1 = The transmitter is disabled. If a character is being processed and a character has been written the DBGU_THR and RSTTX is not set, both characters are completed before the transmitter is stopped.

- **RSTSTA: Reset Status Bits**

0 = No effect.

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

26.6.2 Debug Unit Mode Register

Name: DBGU_MR

Access Type: Read/Write

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

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

15	14	13	12	11	10	9	8
CHMODE		–	–	PAR			–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **PAR: Parity Type**

PAR			Parity Type
0	0	0	Even parity
0	0	1	Odd parity
0	1	0	Space: parity forced to 0
0	1	1	Mark: parity forced to 1
1	x	x	No parity

- **CHMODE: Channel Mode**

CHMODE		Mode Description
0	0	Normal Mode
0	1	Automatic Echo
1	0	Local Loopback
1	1	Remote Loopback

26.6.3 Debug Unit Interrupt Enable Register

Name: DBGU_IER

Access Type: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

- **RXRDY: Enable RXRDY Interrupt**
- **TXRDY: Enable TXRDY Interrupt**
- **ENDRX: Enable End of Receive Transfer Interrupt**
- **ENDTX: Enable End of Transmit Interrupt**
- **OVRE: Enable Overrun Error Interrupt**
- **FRAME: Enable Framing Error Interrupt**
- **PARE: Enable Parity Error Interrupt**
- **TXEMPTY: Enable TXEMPTY Interrupt**
- **TXBUFE: Enable Buffer Empty Interrupt**
- **RXBUFF: Enable Buffer Full Interrupt**
- **COMMTX: Enable COMMTX (from ARM) Interrupt**
- **COMMRX: Enable COMMRX (from ARM) Interrupt**

0 = No effect.

1 = Enables the corresponding interrupt.

26.6.4 Debug Unit Interrupt Disable Register

Name: DBGU_IDR

Access Type: Write-only

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

- **RXRDY: Disable RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **ENDRX: Disable End of Receive Transfer Interrupt**
- **ENDTX: Disable End of Transmit Interrupt**
- **OVRE: Disable Overrun Error Interrupt**
- **FRAME: Disable Framing Error Interrupt**
- **PARE: Disable Parity Error Interrupt**
- **TXEMPTY: Disable TXEMPTY Interrupt**
- **TXBUFE: Disable Buffer Empty Interrupt**
- **RXBUFF: Disable Buffer Full Interrupt**
- **COMMTX: Disable COMMTX (from ARM) Interrupt**
- **COMMRX: Disable COMMRX (from ARM) Interrupt**

0 = No effect.

1 = Disables the corresponding interrupt.

26.6.5 Debug Unit Interrupt Mask Register

Name: DBGU_IMR

Access Type: Read-only

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

- **RXRDY: Mask RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **ENDRX: Mask End of Receive Transfer Interrupt**
- **ENDTX: Mask End of Transmit Interrupt**
- **OVRE: Mask Overrun Error Interrupt**
- **FRAME: Mask Framing Error Interrupt**
- **PARE: Mask Parity Error Interrupt**
- **TXEMPTY: Mask TXEMPTY Interrupt**
- **TXBUFE: Mask TXBUFE Interrupt**
- **RXBUFF: Mask RXBUFF Interrupt**
- **COMMTX: Mask COMMTX Interrupt**
- **COMMRX: Mask COMMRX Interrupt**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

26.6.6 Debug Unit Status Register

Name: DBGU_SR

Access Type: Read-only

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

- **RXRDY: Receiver Ready**

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

1 = At least one complete character has been received, transferred to DBGU_RHR and not yet read.

- **TXRDY: Transmitter Ready**

0 = A character has been written to DBGU_THR and not yet transferred to the Shift Register, or the transmitter is disabled.

1 = There is no character written to DBGU_THR not yet transferred to the Shift Register.

- **ENDRX: End of Receiver Transfer**

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

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

- **ENDTX: End of Transmitter Transfer**

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

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

- **OVRE: Overrun Error**

0 = No overrun error has occurred since the last RSTSTA.

1 = At least one overrun error has occurred since the last RSTSTA.

- **FRAME: Framing Error**

0 = No framing error has occurred since the last RSTSTA.

1 = At least one framing error has occurred since the last RSTSTA.

- **PARE: Parity Error**

0 = No parity error has occurred since the last RSTSTA.

1 = At least one parity error has occurred since the last RSTSTA.

- **TXEMPTY: Transmitter Empty**

0 = There are characters in DBGU_THR, or characters being processed by the transmitter, or the transmitter is disabled.

1 = There are no characters in DBGU_THR and there are no characters being processed by the transmitter.

- **TXBUFE: Transmission Buffer Empty**

0 = The buffer empty signal from the transmitter PDC channel is inactive.

1 = The buffer empty signal from the transmitter PDC channel is active.

- **RXBUFF: Receive Buffer Full**

0 = The buffer full signal from the receiver PDC channel is inactive.

1 = The buffer full signal from the receiver PDC channel is active.

- **COMMTX: Debug Communication Channel Write Status**

0 = COMMTX from the ARM processor is inactive.

1 = COMMTX from the ARM processor is active.

- **COMMRX: Debug Communication Channel Read Status**

0 = COMMRX from the ARM processor is inactive.

1 = COMMRX from the ARM processor is active.

26.6.7 Debug Unit Receiver Holding Register

Name: DBGU_RHR

Access Type: Read-only

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

- **RXCHR: Received Character**

Last received character if RXRDY is set.

26.6.8 Debug Unit Transmit Holding Register

Name: DBGU_THR

Access Type: Write-only

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

- **TXCHR: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set.

26.6.9 Debug Unit Baud Rate Generator Register

Name: DBGU_BRGR

Access Type: Read/Write

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

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

15	14	13	12	11	10	9	8
CD							

7	6	5	4	3	2	1	0
CD							

- **CD: Clock Divisor**

CD	Baud Rate Clock
0	Disabled
1	MCK
2 to 65535	MCK / (CD x 16)

26.6.10 Debug Unit Chip ID Register

Name: DBGU_CIDR

Access Type: Read-only

31	30	29	28	27	26	25	24
EXT	NVPTYP			ARCH			

23	22	21	20	19	18	17	16
ARCH				SRAMSIZ			

15	14	13	12	11	10	9	8
NVPSIZ2				NVPSIZ			

7	6	5	4	3	2	1	0
EPROC			VERSION				

- **VERSION: Version of the Device**

- **EPROC: Embedded Processor**

EPROC			Processor
0	0	1	ARM946ES
0	1	0	ARM7TDMI
1	0	0	ARM920T
1	0	1	ARM926EJS

- **NVPSIZ: Nonvolatile Program Memory Size**

NVPSIZ				Size
0	0	0	0	None
0	0	0	1	8K bytes
0	0	1	0	16K bytes
0	0	1	1	32K bytes
0	1	0	0	Reserved
0	1	0	1	64K bytes
0	1	1	0	Reserved
0	1	1	1	128K bytes
1	0	0	0	Reserved
1	0	0	1	256K bytes
1	0	1	0	512K bytes
1	0	1	1	Reserved
1	1	0	0	1024K bytes
1	1	0	1	Reserved
1	1	1	0	2048K bytes
1	1	1	1	Reserved

- **NVPSIZ2 Second Nonvolatile Program Memory Size**

NVPSIZ2				Size
0	0	0	0	None
0	0	0	1	8K bytes
0	0	1	0	16K bytes
0	0	1	1	32K bytes
0	1	0	0	Reserved
0	1	0	1	64K bytes
0	1	1	0	Reserved
0	1	1	1	128K bytes
1	0	0	0	Reserved
1	0	0	1	256K bytes
1	0	1	0	512K bytes
1	0	1	1	Reserved
1	1	0	0	1024K bytes
1	1	0	1	Reserved
1	1	1	0	2048K bytes
1	1	1	1	Reserved

- **SRAMSIZ: Internal SRAM Size**

SRAMSIZ				Size
0	0	0	0	Reserved
0	0	0	1	1K bytes
0	0	1	0	2K bytes
0	0	1	1	6K bytes
0	1	0	0	112K bytes
0	1	0	1	4K bytes
0	1	1	0	80K bytes
0	1	1	1	160K bytes
1	0	0	0	8K bytes
1	0	0	1	16K bytes
1	0	1	0	32K bytes
1	0	1	1	64K bytes
1	1	0	0	128K bytes
1	1	0	1	256K bytes
1	1	1	0	96K bytes
1	1	1	1	512K bytes

- **ARCH: Architecture Identifier**

ARCH		
Hex	**Bin**	**Architecture**
0x19	0001 1001	AT91SAM9xx Series
0x29	0010 1001	AT91SAM9XExx Series
0x34	0011 0100	AT91x34 Series
0x37	0011 0111	AT91CAP7 Series
0x39	0011 1001	AT91CAP9 Series
0x3B	0011 1011	AT91CAP11 Series
0x40	0100 0000	AT91x40 Series
0x42	0100 0010	AT91x42 Series
0x55	0101 0101	AT91x55 Series
0x60	0110 0000	AT91SAM7Axx Series
0x61	0110 0001	AT91SAM7AQxx Series
0x63	0110 0011	AT91x63 Series
0x70	0111 0000	AT91SAM7Sxx Series
0x71	0111 0001	AT91SAM7XCxx Series
0x72	0111 0010	AT91SAM7SExx Series
0x73	0111 0011	AT91SAM7Lxx Series
0x75	0111 0101	AT91SAM7Xxx Series
0x92	1001 0010	AT91x92 Series
0xF0	1111 0000	AT75Cxx Series

- **NVPTYP: Nonvolatile Program Memory Type**

NVPTYP			Memory
0	0	0	ROM
0	0	1	ROMless or on-chip Flash
1	0	0	SRAM emulating ROM
0	1	0	Embedded Flash Memory
0	1	1	ROM and Embedded Flash Memory NVPSIZ is ROM size NVPSIZ2 is Flash size

- **EXT: Extension Flag**

0 = Chip ID has a single register definition without extension

1 = An extended Chip ID exists.

26.6.11 Debug Unit Chip ID Extension Register

Name: DBGU_EXID

Access Type: Read-only

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

- **EXID: Chip ID Extension**

Reads 0 if the bit EXT in DBGU_CIDR is 0.

26.7 Debug Unit Force NTRST Register

Name: DBGU_FNR

Access Type: Read/Write

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

- **FNTRST: Force NTRST**

0 = NTRST of the ARM processor’s TAP controller is driven by the power_on_reset signal.

1 = NTRST of the ARM processor’s TAP controller is held low.

27. Parallel Input/Output Controller (PIO)

27.1 Description

The Parallel Input/Output Controller (PIO) manages up to 32 fully programmable input/output lines. Each I/O line may be dedicated as a general-purpose I/O or be assigned to a function of an embedded peripheral. This assures effective optimization of the pins of a product.

Each I/O line is associated with a bit number in all of the 32-bit registers of the 32-bit wide User Interface.

Each I/O line of the PIO Controller features:

- An input change interrupt enabling level change detection on any I/O line.
- A glitch filter providing rejection of pulses lower than one-half of clock cycle.
- Multi-drive capability similar to an open drain I/O line.
- Control of the the pull-up of the I/O line.
- Input visibility and output control.

The PIO Controller also features a synchronous output providing up to 32 bits of data output in a single write operation.

27.2 Block Diagram

Figure 27-1. Block Diagram

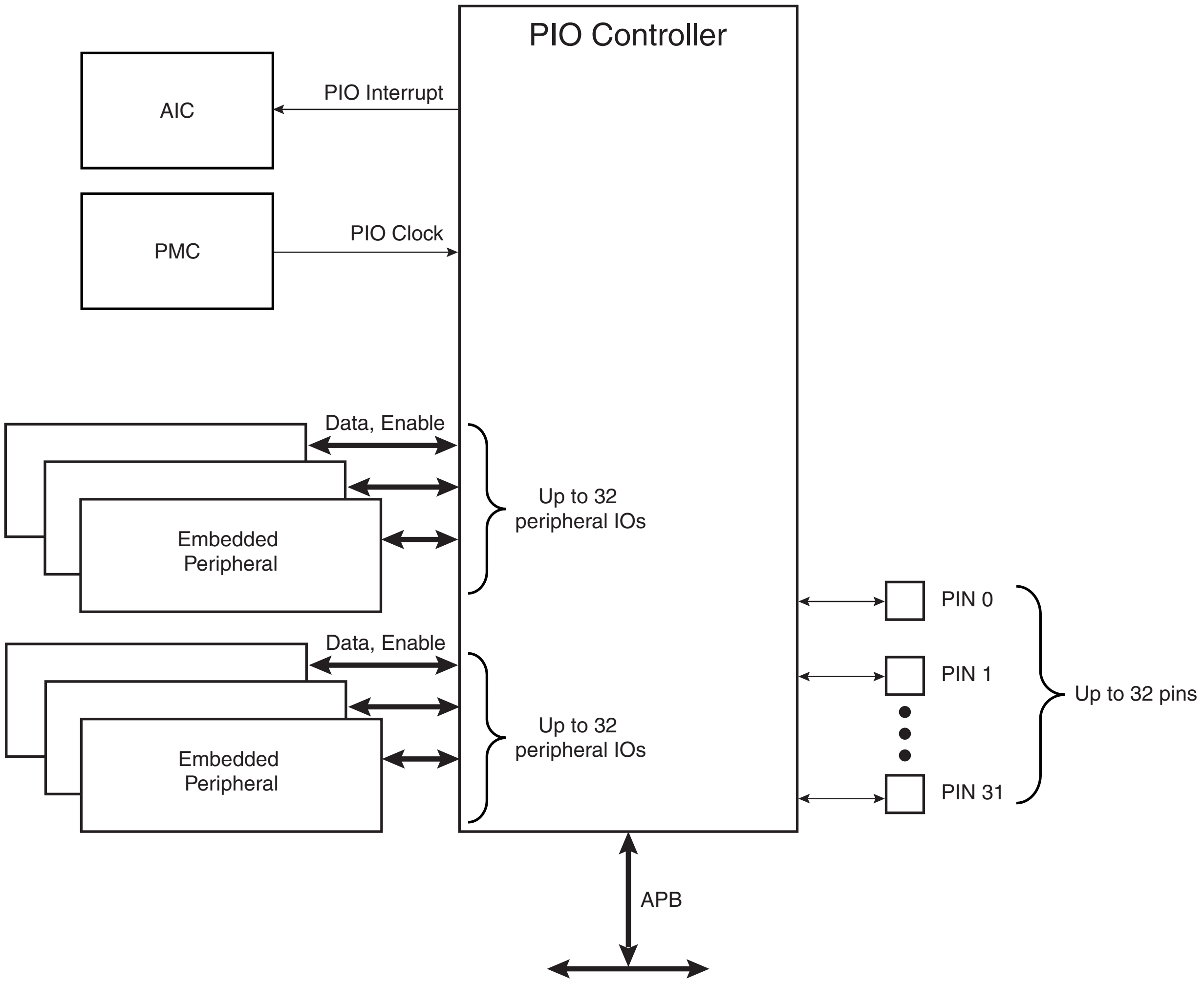
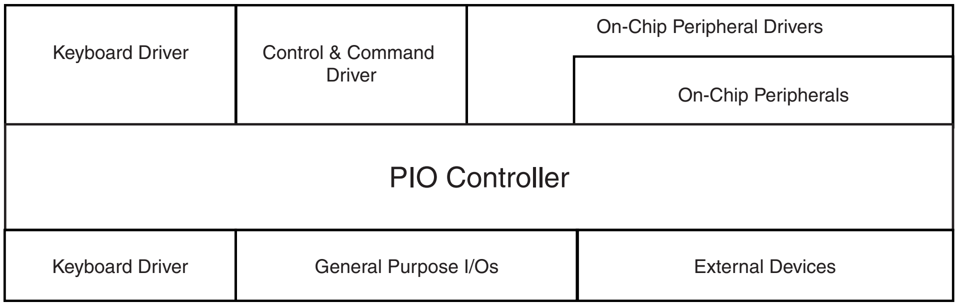

PIO Controller
AIC
PIO Interrupt
PMC
PIO Clock
Data, Enable
Embedded Peripheral
Up to 32 peripheral IOs
Data, Enable
Embedded Peripheral
Up to 32 peripheral IOs
PIN 0
PIN 1
PIN 31
Up to 32 pins
APB

Figure 27-2. Application Block Diagram


Keyboard Driver
Control & Command Driver
On-Chip Peripheral Drivers
On-Chip Peripherals
PIO Controller
Keyboard Driver
General Purpose I/Os
External Devices

27.3 Product Dependencies

27.3.1 Pin Multiplexing

Each pin is configurable, according to product definition as either a general-purpose I/O line only, or as an I/O line multiplexed with one or two peripheral I/Os. As the multiplexing is hardware-defined and thus product-dependent, the hardware designer and programmer must carefully determine the configuration of the PIO controllers required by their application. When an I/O line is general-purpose only, i.e. not multiplexed with any peripheral I/O, programming of the PIO Controller regarding the assignment to a peripheral has no effect and only the PIO Controller can control how the pin is driven by the product.

27.3.2 External Interrupt Lines

The interrupt signals FIQ and IRQ0 to IRQn are most generally multiplexed through the PIO Controllers. However, it is not necessary to assign the I/O line to the interrupt function as the PIO Controller has no effect on inputs and the interrupt lines (FIQ or IRQs) are used only as inputs.

27.3.3 Power Management

The Power Management Controller controls the PIO Controller clock in order to save power. Writing any of the registers of the user interface does not require the PIO Controller clock to be enabled. This means that the configuration of the I/O lines does not require the PIO Controller clock to be enabled.

However, when the clock is disabled, not all of the features of the PIO Controller are available. Note that the Input Change Interrupt and the read of the pin level require the clock to be validated.

After a hardware reset, the PIO clock is disabled by default.

The user must configure the Power Management Controller before any access to the input line information.

27.3.4 Interrupt Generation

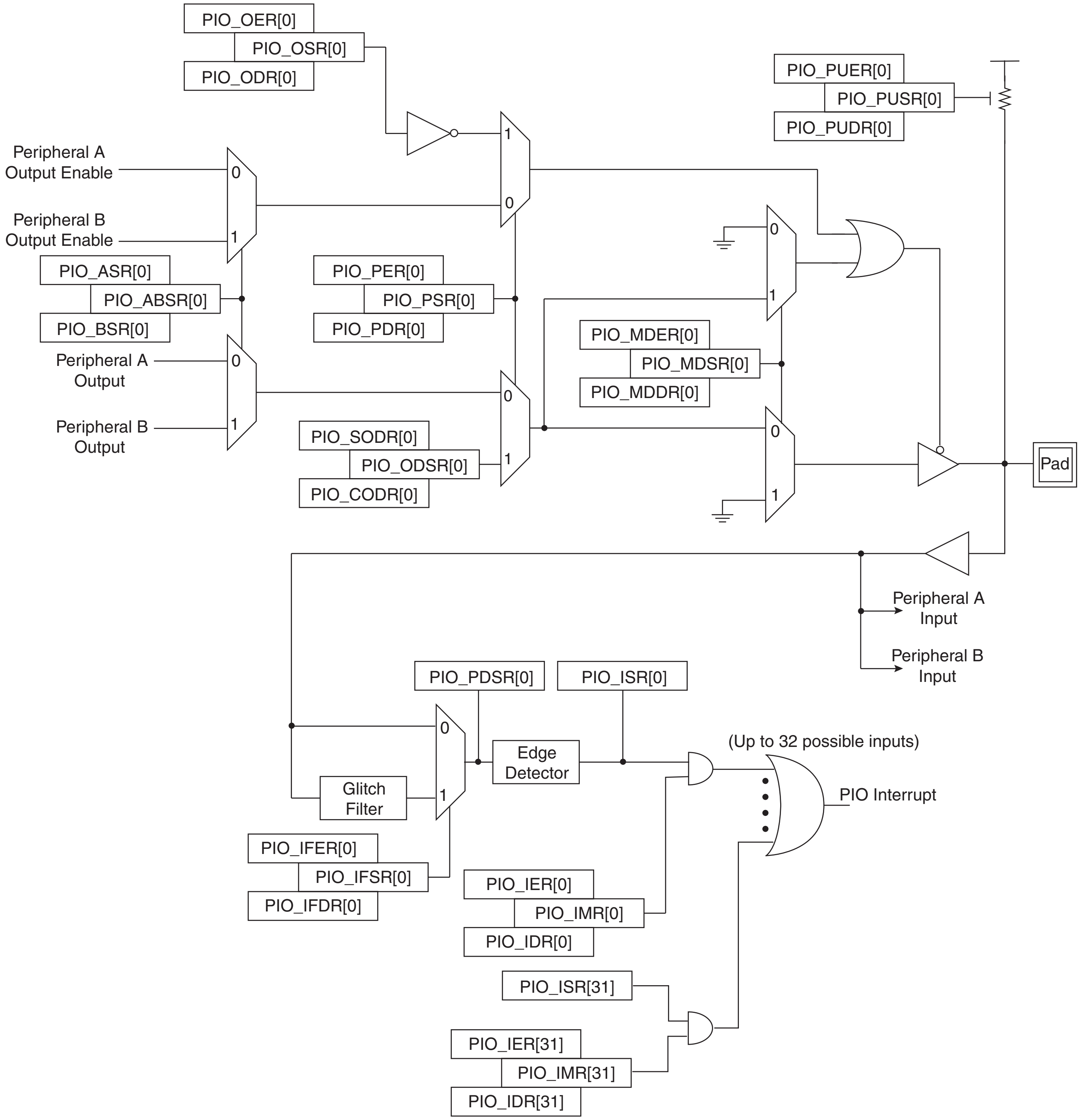
For interrupt handling, the PIO Controllers are considered as user peripherals. This means that the PIO Controller interrupt lines are connected among the interrupt sources 2 to 31. Refer to the PIO Controller peripheral identifier in the product description to identify the interrupt sources dedicated to the PIO Controllers.

The PIO Controller interrupt can be generated only if the PIO Controller clock is enabled.

27.4 Functional Description

The PIO Controller features up to 32 fully-programmable I/O lines. Most of the control logic associated to each I/O is represented in Figure 27-3. In this description each signal shown represents but one of up to 32 possible indexes.

Figure 27-3. I/O Line Control Logic


PIO_OER[0]
PIO_OSR[0]
PIO_ODR[0]
PIO_PUER[0]
PIO_PUSR[0]
PIO_PUDR[0]
Peripheral A Output Enable
Peripheral B Output Enable
PIO_ASR[0]
PIO_ABSR[0]
PIO_BSR[0]
PIO_PER[0]
PIO_PSR[0]
PIO_PDR[0]
PIO_MDER[0]
PIO_MDSR[0]
PIO_MDDR[0]
Peripheral A Output
Peripheral B Output
PIO_SODR[0]
PIO_ODSR[0]
PIO_CODR[0]
Pad
Peripheral A Input
Peripheral B Input
PIO_PDSR[0]
PIO_ISR[0]
Edge Detector
Glitch Filter
(Up to 32 possible inputs)
PIO Interrupt
PIO_IFER[0]
PIO_IFSR[0]
PIO_IFDR[0]
PIO_IER[0]
PIO_IMR[0]
PIO_IDR[0]
PIO_ISR[31]
PIO_IER[31]
PIO_IMR[31]
PIO_IDR[31]
0
1

27.4.1 Pull-up Resistor Control

Each I/O line is designed with an embedded pull-up resistor. The pull-up resistor can be enabled or disabled by writing respectively PIO_PUER (Pull-up Enable Register) and PIO_PUDR (Pull-up Disable Resistor). Writing in these registers results in setting or clearing the corresponding bit in PIO_PUSR (Pull-up Status Register). Reading a 1 in PIO_PUSR means the pull-up is disabled and reading a 0 means the pull-up is enabled.

Control of the pull-up resistor is possible regardless of the configuration of the I/O line.

After reset, all of the pull-ups are enabled, i.e. PIO_PUSR resets at the value 0x0.

27.4.2 I/O Line or Peripheral Function Selection

When a pin is multiplexed with one or two peripheral functions, the selection is controlled with the registers PIO_PER (PIO Enable Register) and PIO_PDR (PIO Disable Register). The register PIO_PSR (PIO Status Register) is the result of the set and clear registers and indicates whether the pin is controlled by the corresponding peripheral or by the PIO Controller. A value of 0 indicates that the pin is controlled by the corresponding on-chip peripheral selected in the PIO_ABSR (AB Select Status Register). A value of 1 indicates the pin is controlled by the PIO controller.

If a pin is used as a general purpose I/O line (not multiplexed with an on-chip peripheral), PIO_PER and PIO_PDR have no effect and PIO_PSR returns 1 for the corresponding bit.

After reset, most generally, the I/O lines are controlled by the PIO controller, i.e. PIO_PSR resets at 1. However, in some events, it is important that PIO lines are controlled by the peripheral (as in the case of memory chip select lines that must be driven inactive after reset or for address lines that must be driven low for booting out of an external memory). Thus, the reset value of PIO_PSR is defined at the product level, depending on the multiplexing of the device.

27.4.3 Peripheral A or B Selection

The PIO Controller provides multiplexing of up to two peripheral functions on a single pin. The selection is performed by writing PIO_ASR (A Select Register) and PIO_BSR (Select B Register). PIO_ABSR (AB Select Status Register) indicates which peripheral line is currently selected. For each pin, the corresponding bit at level 0 means peripheral A is selected whereas the corresponding bit at level 1 indicates that peripheral B is selected.

Note that multiplexing of peripheral lines A and B only affects the output line. The peripheral input lines are always connected to the pin input.

After reset, PIO_ABSR is 0, thus indicating that all the PIO lines are configured on peripheral A. However, peripheral A generally does not drive the pin as the PIO Controller resets in I/O line mode.

Writing in PIO_ASR and PIO_BSR manages PIO_ABSR regardless of the configuration of the pin. However, assignment of a pin to a peripheral function requires a write in the corresponding peripheral selection register (PIO_ASR or PIO_BSR) in addition to a write in PIO_PDR.

27.4.4 Output Control

When the I/0 line is assigned to a peripheral function, i.e. the corresponding bit in PIO_PSR is at 0, the drive of the I/O line is controlled by the peripheral. Peripheral A or B, depending on the value in PIO_ABSR, determines whether the pin is driven or not.

When the I/O line is controlled by the PIO controller, the pin can be configured to be driven. This is done by writing PIO_OER (Output Enable Register) and PIO_ODR (Output Disable Register).

The results of these write operations are detected in PIO_OSR (Output Status Register). When a bit in this register is at 0, the corresponding I/O line is used as an input only. When the bit is at 1, the corresponding I/O line is driven by the PIO controller.

The level driven on an I/O line can be determined by writing in PIO_SODR (Set Output Data Register) and PIO_CODR (Clear Output Data Register). These write operations respectively set and clear PIO_ODSR (Output Data Status Register), which represents the data driven on the I/O lines. Writing in PIO_OER and PIO_ODR manages PIO_OSR whether the pin is configured to be controlled by the PIO controller or assigned to a peripheral function. This enables configuration of the I/O line prior to setting it to be managed by the PIO Controller.

Similarly, writing in PIO_SODR and PIO_CODR effects PIO_ODSR. This is important as it defines the first level driven on the I/O line.

27.4.5 Synchronous Data Output

Controlling all parallel busses using several PIOs requires two successive write operations in the PIO_SODR and PIO_CODR registers. This may lead to unexpected transient values. The PIO controller offers a direct control of PIO outputs by single write access to PIO_ODSR (Output Data Status Register). Only bits unmasked by PIO_OWSR (Output Write Status Register) are written. The mask bits in the PIO_OWSR are set by writing to PIO_OWER (Output Write Enable Register) and cleared by writing to PIO_OWDR (Output Write Disable Register).

After reset, the synchronous data output is disabled on all the I/O lines as PIO_OWSR resets at 0x0.

27.4.6 Multi Drive Control (Open Drain)

Each I/O can be independently programmed in Open Drain by using the Multi Drive feature. This feature permits several drivers to be connected on the I/O line which is driven low only by each device. An external pull-up resistor (or enabling of the internal one) is generally required to guarantee a high level on the line.

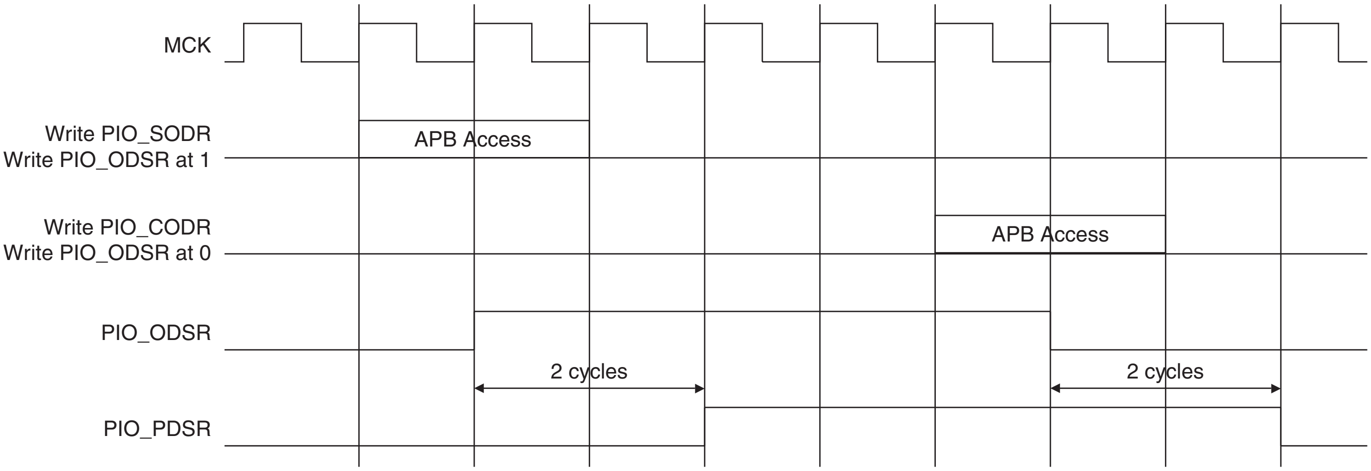
The Multi Drive feature is controlled by PIO_MDER (Multi-driver Enable Register) and PIO_MDDR (Multi-driver Disable Register). The Multi Drive can be selected whether the I/O line is controlled by the PIO controller or assigned to a peripheral function. PIO_MDSR (Multi-driver Status Register) indicates the pins that are configured to support external drivers.

After reset, the Multi Drive feature is disabled on all pins, i.e. PIO_MDSR resets at value 0x0.

27.4.7 Output Line Timings

Figure 27-4 shows how the outputs are driven either by writing PIO_SODR or PIO_CODR, or by directly writing PIO_ODSR. This last case is valid only if the corresponding bit in PIO_OWSR is set. Figure 27-4 also shows when the feedback in PIO_PDSR is available.

Figure 27-4. Output Line Timings



27.4.8 Inputs

The level on each I/O line can be read through PIO_PDSR (Pin Data Status Register). This register indicates the level of the I/O lines regardless of their configuration, whether uniquely as an input or driven by the PIO controller or driven by a peripheral.

Reading the I/O line levels requires the clock of the PIO controller to be enabled, otherwise PIO_PDSR reads the levels present on the I/O line at the time the clock was disabled.

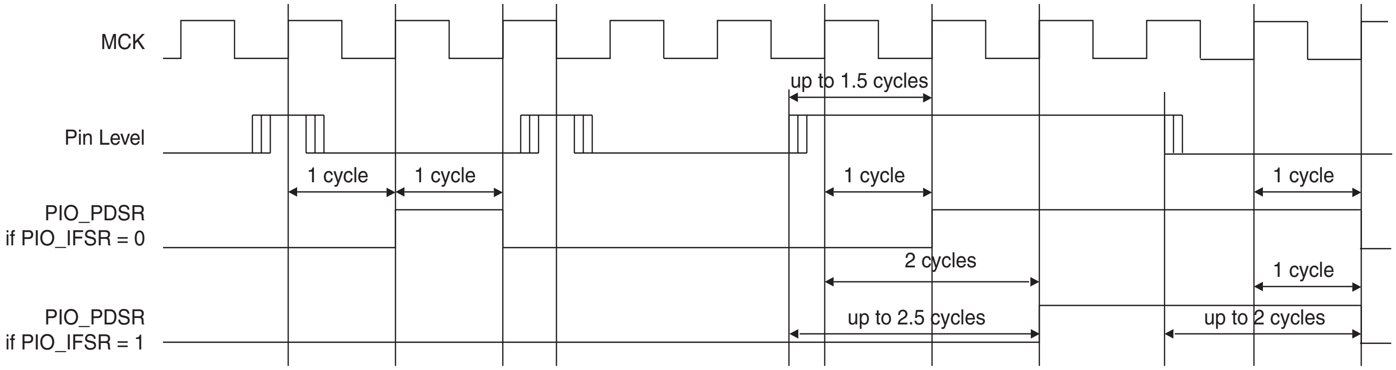
27.4.9 Input Glitch Filtering

Optional input glitch filters are independently programmable on each I/O line. When the glitch filter is enabled, a glitch with a duration of less than 1/2 Master Clock (MCK) cycle is automatically rejected, while a pulse with a duration of 1 Master Clock cycle or more is accepted. For pulse durations between 1/2 Master Clock cycle and 1 Master Clock cycle the pulse may or may not be taken into account, depending on the precise timing of its occurrence. Thus for a pulse to be visible it must exceed 1 Master Clock cycle, whereas for a glitch to be reliably filtered out, its duration must not exceed 1/2 Master Clock cycle. The filter introduces one Master Clock cycle latency if the pin level change occurs before a rising edge. However, this latency does not appear if the pin level change occurs before a falling edge. This is illustrated in Figure 27-5.

The glitch filters are controlled by the register set; PIO_IFER (Input Filter Enable Register), PIO_IFDR (Input Filter Disable Register) and PIO_IFSR (Input Filter Status Register). Writing PIO_IFER and PIO_IFDR respectively sets and clears bits in PIO_IFSR. This last register enables the glitch filter on the I/O lines.

When the glitch filter is enabled, it does not modify the behavior of the inputs on the peripherals. It acts only on the value read in PIO_PDSR and on the input change interrupt detection. The glitch filters require that the PIO Controller clock is enabled.

Figure 27-5. Input Glitch Filter Timing


MCK
up to 1.5 cycles
Pin Level
1 cycle
1 cycle
1 cycle
1 cycle
PIO_PDSR
if PIO_IFSR = 0
2 cycles
1 cycle
PIO_PDSR
if PIO_IFSR = 1
up to 2.5 cycles
up to 2 cycles

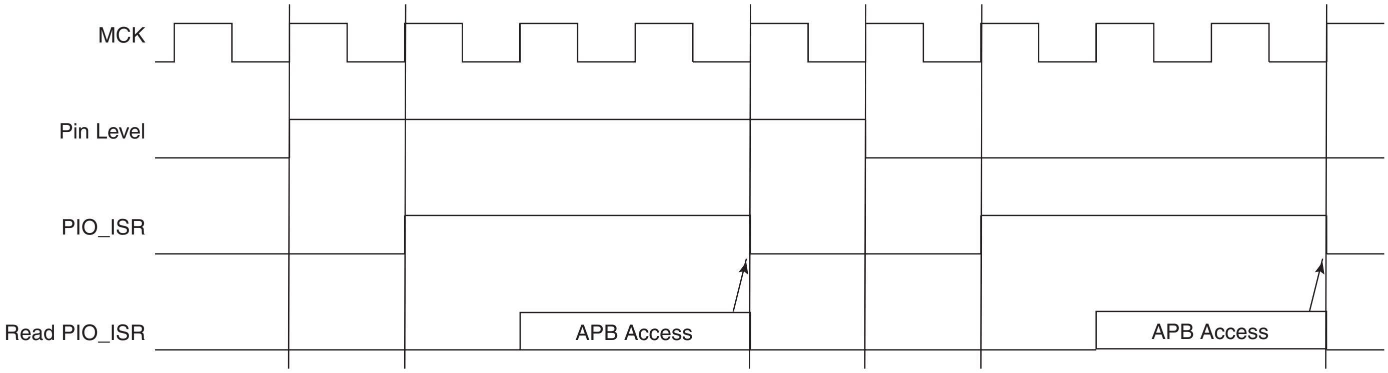
27.4.10 Input Change Interrupt

The PIO Controller can be programmed to generate an interrupt when it detects an input change on an I/O line. The Input Change Interrupt is controlled by writing PIO_IER (Interrupt Enable Register) and PIO_IDR (Interrupt Disable Register), which respectively enable and disable the input change interrupt by setting and clearing the corresponding bit in PIO_IMR (Interrupt Mask Register). As Input change detection is possible only by comparing two successive samplings of the input of the I/O line, the PIO Controller clock must be enabled. The Input Change Interrupt is available, regardless of the configuration of the I/O line, i.e. configured as an input only, controlled by the PIO Controller or assigned to a peripheral function.

When an input change is detected on an I/O line, the corresponding bit in PIO_ISR (Interrupt Status Register) is set. If the corresponding bit in PIO_IMR is set, the PIO Controller interrupt line is asserted. The interrupt signals of the thirty-two channels are ORed-wired together to generate a single interrupt signal to the Advanced Interrupt Controller.

When the software reads PIO_ISR, all the interrupts are automatically cleared. This signifies that all the interrupts that are pending when PIO_ISR is read must be handled.

Figure 27-6. Input Change Interrupt Timings


MCK
Pin Level
PIO_ISR
Read PIO_ISR
APB Access
APB Access

27.5 I/O Lines Programming Example

The programing example as shown in Table 27-1 below is used to define the following configuration.

- 4-bit output port on I/O lines 0 to 3, (should be written in a single write operation), open-drain, with pull-up resistor

- Four output signals on I/O lines 4 to 7 (to drive LEDs for example), driven high and low, no pull-up resistor
- Four input signals on I/O lines 8 to 11 (to read push-button states for example), with pull-up resistors, glitch filters and input change interrupts
- Four input signals on I/O line 12 to 15 to read an external device status (polled, thus no input change interrupt), no pull-up resistor, no glitch filter
- I/O lines 16 to 19 assigned to peripheral A functions with pull-up resistor
- I/O lines 20 to 23 assigned to peripheral B functions, no pull-up resistor
- I/O line 24 to 27 assigned to peripheral A with Input Change Interrupt and pull-up resistor

Table 27-1. Programming Example

Register	Value to be Written
PIO_PER	0x0000 FFFF
PIO_PDR	0x0FFF 0000
PIO_OER	0x0000 00FF
PIO_ODR	0x0FFF FF00
PIO_IFER	0x0000 0F00
PIO_IFDR	0x0FFF F0FF
PIO_SODR	0x0000 0000
PIO_CODR	0x0FFF FFFF
PIO_IER	0x0F00 0F00
PIO_IDR	0x00FF F0FF
PIO_MDER	0x0000 000F
PIO_MDDR	0x0FFF FFF0
PIO_PUDR	0x00F0 00F0
PIO_PUER	0x0F0F FF0F
PIO_ASR	0x0F0F 0000
PIO_BSR	0x00F0 0000
PIO_OWER	0x0000 000F
PIO_OWDR	0x0FFF FFF0

27.6 User Interface

Each I/O line controlled by the PIO Controller is associated with a bit in each of the PIO Controller User Interface registers. Each register is 32 bits wide. If a parallel I/O line is not defined, writing to the corresponding bits has no effect. Undefined bits read zero. If the I/O line is not multiplexed with any peripheral, the I/O line is controlled by the PIO Controller and PIO_PSR returns 1 systematically.

Table 27-2. Register Mapping

Offset	Register	Name	Access	Reset Value
0x0000	PIO Enable Register	PIO_PER	Write-only	–
0x0004	PIO Disable Register	PIO_PDR	Write-only	–
0x0008	PIO Status Register	PIO_PSR	Read-only	(1)
0x000C	Reserved			
0x0010	Output Enable Register	PIO_OER	Write-only	–
0x0014	Output Disable Register	PIO_ODR	Write-only	–
0x0018	Output Status Register	PIO_OSR	Read-only	0x0000 0000
0x001C	Reserved			
0x0020	Glitch Input Filter Enable Register	PIO_IFER	Write-only	–
0x0024	Glitch Input Filter Disable Register	PIO_IFDR	Write-only	–
0x0028	Glitch Input Filter Status Register	PIO_IFSR	Read-only	0x0000 0000
0x002C	Reserved			
0x0030	Set Output Data Register	PIO_SODR	Write-only	–
0x0034	Clear Output Data Register	PIO_CODR	Write-only	
0x0038	Output Data Status Register	PIO_ODSR	Read-only or(2) Read/Write	–
0x003C	Pin Data Status Register	PIO_PDSR	Read-only	(3)
0x0040	Interrupt Enable Register	PIO_IER	Write-only	–
0x0044	Interrupt Disable Register	PIO_IDR	Write-only	–
0x0048	Interrupt Mask Register	PIO_IMR	Read-only	0x00000000
0x004C	Interrupt Status Register(4)	PIO_ISR	Read-only	0x00000000
0x0050	Multi-driver Enable Register	PIO_MDER	Write-only	–
0x0054	Multi-driver Disable Register	PIO_MDDR	Write-only	–
0x0058	Multi-driver Status Register	PIO_MDSR	Read-only	0x00000000
0x005C	Reserved			
0x0060	Pull-up Disable Register	PIO_PUDR	Write-only	–
0x0064	Pull-up Enable Register	PIO_PUER	Write-only	–
0x0068	Pad Pull-up Status Register	PIO_PUSR	Read-only	0x00000000
0x006C	Reserved			

Table 27-2. Register Mapping (Continued)

Offset	Register	Name	Access	Reset Value
0x0070	Peripheral A Select Register(5)	PIO_ASR	Write-only	–
0x0074	Peripheral B Select Register(5)	PIO_BSR	Write-only	–
0x0078	AB Status Register(5)	PIO_ABSR	Read-only	0x00000000
0x007C to 0x009C	Reserved			
0x00A0	Output Write Enable	PIO_OWER	Write-only	–
0x00A4	Output Write Disable	PIO_OWDR	Write-only	–
0x00A8	Output Write Status Register	PIO_OWSR	Read-only	0x00000000
0x00AC	Reserved			

Notes:
1. Reset value of PIO_PSR depends on the product implementation.
2. PIO_ODSR is Read-only or Read/Write depending on PIO_OWSR I/O lines.
3. Reset value of PIO_PDSR depends on the level of the I/O lines. Reading the I/O line levels requires the clock of the PIO Controller to be enabled, otherwise PIO_PDSR reads the levels present on the I/O line at the time the clock was disabled.
4. PIO_ISR is reset at 0x0. However, the first read of the register may read a different value as input changes may have occurred.
5. Only this set of registers clears the status by writing 1 in the first register and sets the status by writing 1 in the second register.

27.6.1 PIO Controller PIO Enable Register

Name: PIO_PER

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: PIO Enable**

0 = No effect.

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

27.6.2 PIO Controller PIO Disable Register

Name: PIO_PDR

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: PIO Disable**

0 = No effect.

1 = Disables the PIO from controlling the corresponding pin (enables peripheral control of the pin).

27.6.3 PIO Controller PIO Status Register

Name: PIO_PSR

Access Type: Read-only

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

- **P0-P31: PIO Status**

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

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

27.6.4 PIO Controller Output Enable Register

Name: PIO_OER

Access Type: Write-only

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

- **P0-P31: Output Enable**

0 = No effect.

1 = Enables the output on the I/O line.

27.6.5 PIO Controller Output Disable Register

Name: PIO_ODR

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Output Disable**

0 = No effect.

1 = Disables the output on the I/O line.

27.6.6 PIO Controller Output Status Register

Name: PIO_OSR

Access Type: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Output Status**

0 = The I/O line is a pure input.

1 = The I/O line is enabled in output.

27.6.7 PIO Controller Input Filter Enable Register

Name: PIO_IFER

Access Type: Write-only

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

- **P0-P31: Input Filter Enable**

0 = No effect.

1 = Enables the input glitch filter on the I/O line.

27.6.8 PIO Controller Input Filter Disable Register

Name: PIO_IFDR

Access Type: Write-only

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

- **P0-P31: Input Filter Disable**

0 = No effect.

1 = Disables the input glitch filter on the I/O line.

27.6.9 PIO Controller Input Filter Status Register

Name: PIO_IFSR

Access Type: Read-only

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

- **P0-P31: Input Filer Status**

0 = The input glitch filter is disabled on the I/O line.

1 = The input glitch filter is enabled on the I/O line.

27.6.10 PIO Controller Set Output Data Register

Name: PIO_SODR

Access Type: Write-only

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

- **P0-P31: Set Output Data**

0 = No effect.

1 = Sets the data to be driven on the I/O line.

27.6.11 PIO Controller Clear Output Data Register

Name: PIO_CODR

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Set Output Data**

0 = No effect.

1 = Clears the data to be driven on the I/O line.

27.6.12 PIO Controller Output Data Status Register

Name: PIO_ODSR

Access Type: Read-only or Read/Write

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Output Data Status**

0 = The data to be driven on the I/O line is 0.

1 = The data to be driven on the I/O line is 1.

27.6.13 PIO Controller Pin Data Status Register

Name: PIO_PDSR

Access Type: Read-only

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

- **P0-P31: Output Data Status**

0 = The I/O line is at level 0.

1 = The I/O line is at level 1.

27.6.14 PIO Controller Interrupt Enable Register

Name: PIO_IER

Access Type: Write-only

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

- **P0-P31: Input Change Interrupt Enable**

0 = No effect.

1 = Enables the Input Change Interrupt on the I/O line.

27.6.15 PIO Controller Interrupt Disable Register

Name: PIO_IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Input Change Interrupt Disable**

0 = No effect.

1 = Disables the Input Change Interrupt on the I/O line.

27.6.16 PIO Controller Interrupt Mask Register

Name: PIO_IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Input Change Interrupt Mask**

0 = Input Change Interrupt is disabled on the I/O line.

1 = Input Change Interrupt is enabled on the I/O line.

27.6.17 PIO Controller Interrupt Status Register

Name: PIO_ISR

Access Type: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Input Change Interrupt Status**

0 = No Input Change has been detected on the I/O line since PIO_ISR was last read or since reset.

1 = At least one Input Change has been detected on the I/O line since PIO_ISR was last read or since reset.

27.6.18 PIO Multi-driver Enable Register

Name: PIO_MDER

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Multi Drive Enable.**

0 = No effect.

1 = Enables Multi Drive on the I/O line.

27.6.19 PIO Multi-driver Disable Register

Name: PIO_MDDR

Access Type: Write-only

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

- **P0-P31: Multi Drive Disable.**

0 = No effect.

1 = Disables Multi Drive on the I/O line.

27.6.20 PIO Multi-driver Status Register

Name: PIO_MDSR

Access Type: Read-only

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

- **P0-P31: Multi Drive Status.**

0 = The Multi Drive is disabled on the I/O line. The pin is driven at high and low level.

1 = The Multi Drive is enabled on the I/O line. The pin is driven at low level only.

27.6.21 PIO Pull Up Disable Register

Name: PIO_PUDR

Access Type: Write-only

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

- **P0-P31: Pull Up Disable.**

0 = No effect.

1 = Disables the pull up resistor on the I/O line.

27.6.22 PIO Pull Up Enable Register

Name: PIO_PUER

Access Type: Write-only

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

- **P0-P31: Pull Up Enable.**

0 = No effect.

1 = Enables the pull up resistor on the I/O line.

27.6.23 PIO Pull Up Status Register

Name: PIO_PUSR

Access Type: Read-only

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

- **P0-P31: Pull Up Status.**

0 = Pull Up resistor is enabled on the I/O line.

1 = Pull Up resistor is disabled on the I/O line.

27.6.24 PIO Peripheral A Select Register

Name: PIO_ASR

Access Type: Write-only

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

- **P0-P31: Peripheral A Select.**

0 = No effect.

1 = Assigns the I/O line to the Peripheral A function.

27.6.25 PIO Peripheral B Select Register

Name: PIO_BSR

Access Type: Write-only

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

- **P0-P31: Peripheral B Select.**

0 = No effect.

1 = Assigns the I/O line to the peripheral B function.

27.6.26 PIO Peripheral A B Status Register

Name: PIO_ABSR

Access Type: Read-only

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

- **P0-P31: Peripheral A B Status.**

0 = The I/O line is assigned to the Peripheral A.

1 = The I/O line is assigned to the Peripheral B.

27.6.27 PIO Output Write Enable Register

Name: PIO_OWER

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Output Write Enable.**

0 = No effect.

1 = Enables writing PIO_ODSR for the I/O line.

27.6.28 PIO Output Write Disable Register

Name: PIO_OWDR

Access Type: Write-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Output Write Disable.**

0 = No effect.

1 = Disables writing PIO_ODSR for the I/O line.

27.6.29 PIO Output Write Status Register

Name: PIO_OWSR

Access Type: Read-only

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

- **P0-P31: Output Write Status.**

0 = Writing PIO_ODSR does not affect the I/O line.

1 = Writing PIO_ODSR affects the I/O line.

28. Serial Peripheral Interface (SPI)

28.1 Description

The Serial Peripheral Interface (SPI) circuit is a synchronous serial data link that provides communication with external devices in Master or Slave Mode. It also enables communication between processors if an external processor is connected to the system.

The Serial Peripheral Interface is essentially a shift register that serially transmits data bits to other SPIs. During a data transfer, one SPI system acts as the “master”' which controls the data flow, while the other devices act as “slaves'' which have data shifted into and out by the master. Different CPUs can take turn being masters (Multiple Master Protocol opposite to Single Master Protocol where one CPU is always the master while all of the others are always slaves) and one master may simultaneously shift data into multiple slaves. However, only one slave may drive its output to write data back to the master at any given time.

A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

The SPI system consists of two data lines and two control lines:

- Master Out Slave In (MOSI): This data line supplies the output data from the master shifted into the input(s) of the slave(s).
- Master In Slave Out (MISO): This data line supplies the output data from a slave to the input of the master. There may be no more than one slave transmitting data during any particular transfer.
- Serial Clock (SPCK): This control line is driven by the master and regulates the flow of the data bits. The master may transmit data at a variety of baud rates; the SPCK line cycles once for each bit that is transmitted.
- Slave Select (NSS): This control line allows slaves to be turned on and off by hardware.

28.2 Block Diagram

Figure 28-1. Block Diagram

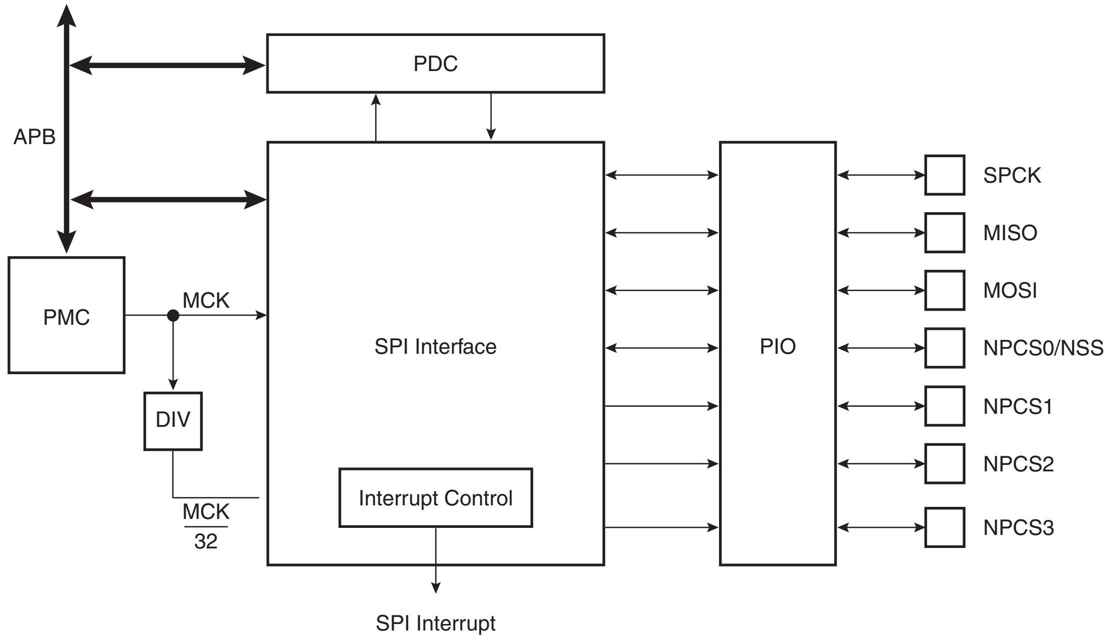
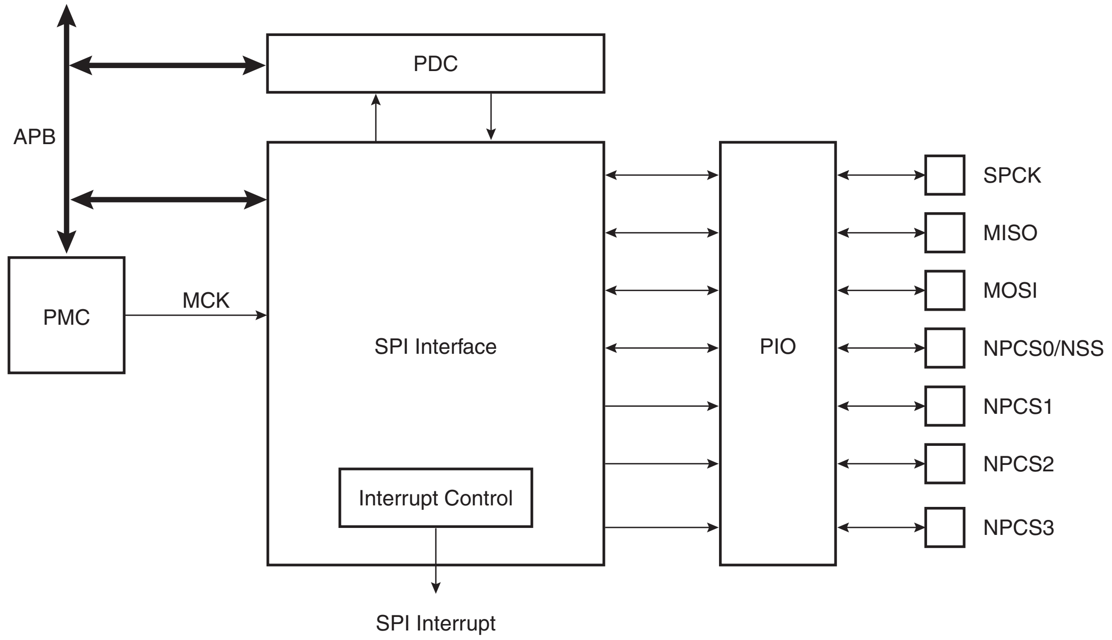
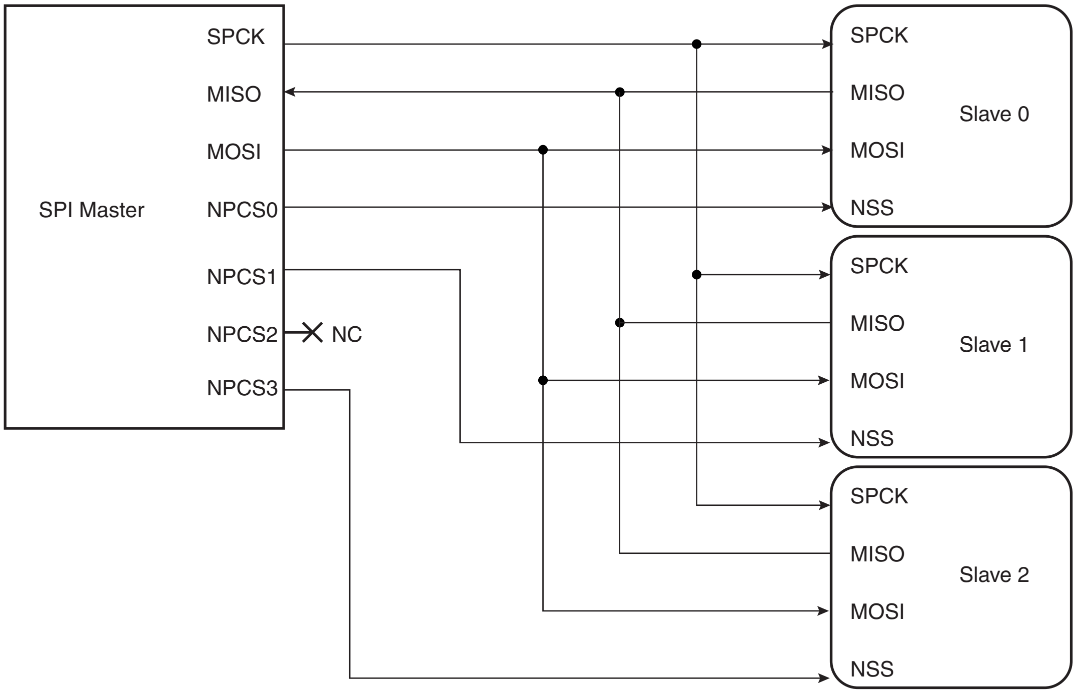

PDC
APB
SPI Interface
PIO
PMC
MCK
DIV
MCK
32
Interrupt Control
SPI Interrupt
SPCK
MISO
MOSI
NPCS0/NSS
NPCS1
NPCS2
NPCS3

Figure 28-2. Block Diagram


PDC
APB
SPI Interface
PIO
PMC
MCK
Interrupt Control
SPI Interrupt
SPCK
MISO
MOSI
NPCS0/NSS
NPCS1
NPCS2
NPCS3

28.3 Application Block Diagram

Figure 28-3. Application Block Diagram: Single Master/Multiple Slave Implementation


SPI Master
SPCK
MISO
MOSI
NPCS0
NPCS1
NPCS2
NC
NPCS3
SPCK
MISO
MOSI
NSS
Slave 0
SPCK
MISO
MOSI
NSS
Slave 1
SPCK
MISO
MOSI
NSS
Slave 2

28.4 Signal Description

Table 28-1. Signal Description

Pin Name	Pin Description	Type	
		Master	**Slave**
MISO	Master In Slave Out	Input	Output
MOSI	Master Out Slave In	Output	Input
SPCK	Serial Clock	Output	Input
NPCS1-NPCS3	Peripheral Chip Selects	Output	Unused
NPCS0/NSS	Peripheral Chip Select/Slave Select	Output	Input

28.5 Product Dependencies

28.5.1 I/O Lines

The pins used for interfacing the compliant external devices are multiplexed with PIO lines. The programmer must first program the PIOA controller to select the SPI I/O alternate functions.

28.5.2 Power Management

The SPI may be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the SPI clock.

28.5.3 Interrupt

The SPI interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling the SPI interrupt requires programming the AIC before configuring the SPI.

28.6 Functional Description

28.6.1 Modes of Operation

The SPI operates in Master Mode or in Slave Mode.

Operation in Master Mode is programmed by writing at 1 the MSTR bit in the Mode Register. The pins NPCS0 to NPCS3 are all configured as outputs, the SPCK pin is driven, the MISO line is wired on the receiver input and the MOSI line driven as an output by the transmitter.

If the MSTR bit is written at 0, the SPI operates in Slave Mode. The MISO line is driven by the transmitter output, the MOSI line is wired on the receiver input, the SPCK pin is driven by the transmitter to synchronize the receiver. The NPCS0 pin becomes an input, and is used as a Slave Select signal (NSS). The pins NPCS1 to NPCS3 are not driven and can be used for other purposes.

The data transfers are identically programmable for both modes of operations. The baud rate generator is activated only in Master Mode.

28.6.2 Data Transfer

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the Chip Select Register. The clock phase is programmed with the NCPHA bit. These two parameters determine the edges of the clock signal on which data is driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations that are incompatible with one another. Thus, a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are used and fixed in different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

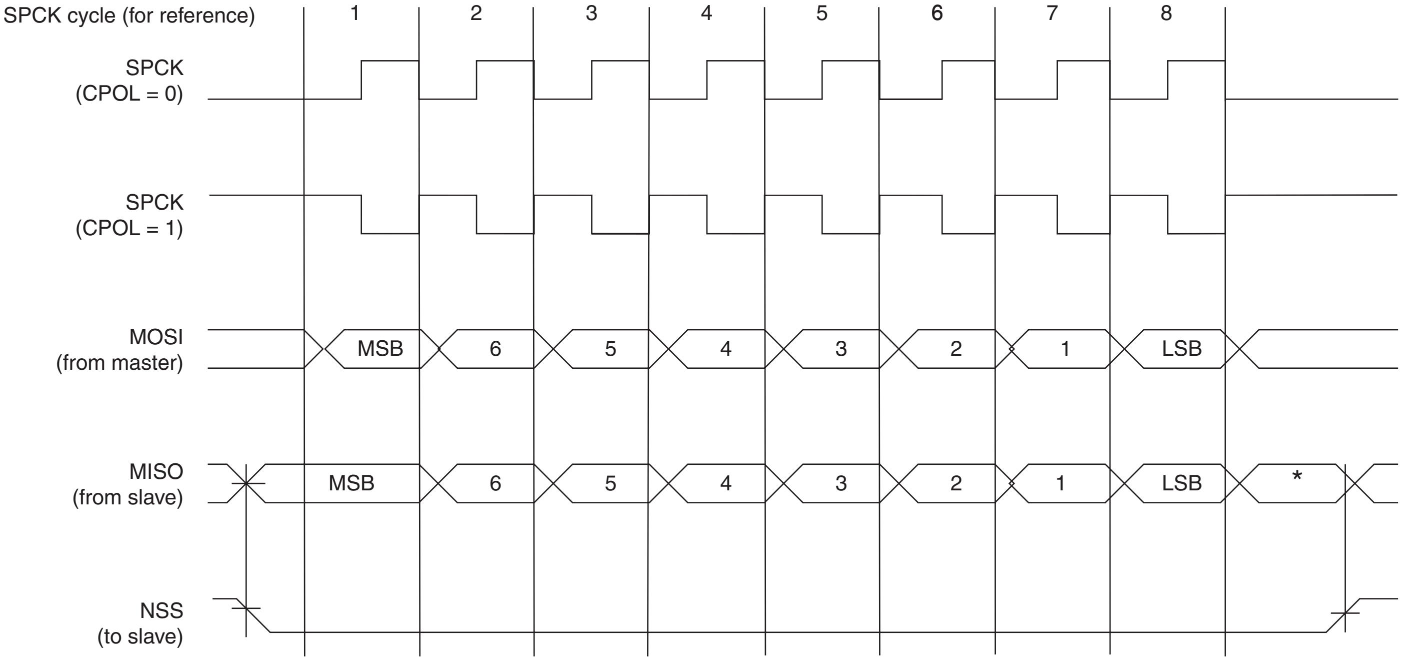
Table 28-2 shows the four modes and corresponding parameter settings.

Table 28-2. SPI Bus Protocol Mode

SPI Mode	CPOL	NCPHA
0	0	1
1	0	0
2	1	1
3	1	0

Figure 28-4 and Figure 28-5 show examples of data transfers.

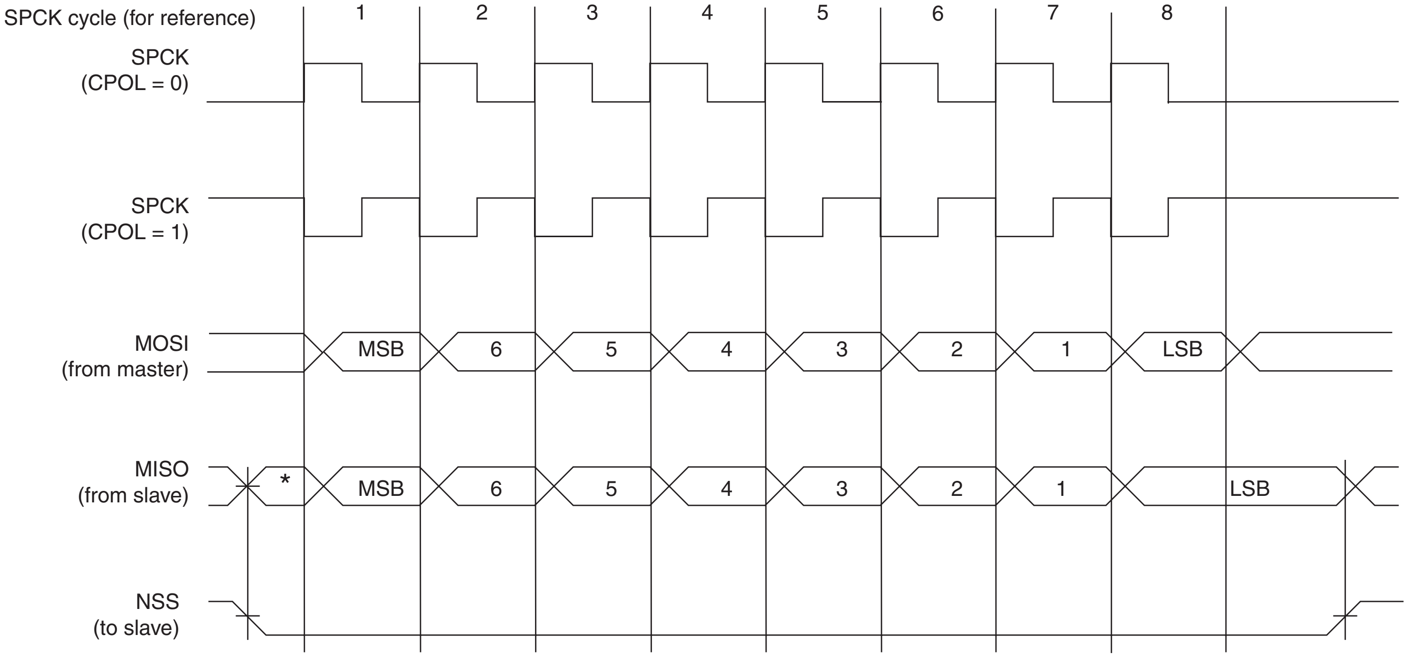
Figure 28-4. SPI Transfer Format (NCPHA = 1, 8 bits per transfer)



SPCK cycle (for reference)
1
2
3
4
5
6
7
8
SPCK (CPOL = 0)
SPCK (CPOL = 1)
MOSI (from master)
MSB
6
5
4
3
2
1
LSB
MISO (from slave)
MSB
6
5
4
3
2
1
LSB
*
NSS (to slave)

* Not defined, but normally MSB of previous character received.

Figure 28-5. SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



SPCK cycle (for reference)
1
2
3
4
5
6
7
8
SPCK (CPOL = 0)
SPCK (CPOL = 1)
MOSI (from master)
MSB
6
5
4
3
2
1
LSB
MISO (from slave)
*
MSB
6
5
4
3
2
1
LSB
NSS (to slave)

* Not defined but normally LSB of previous character transmitted.

28.6.3 Master Mode Operations

When configured in Master Mode, the SPI operates on the clock generated by the internal programmable baud rate generator. It fully controls the data transfers to and from the slave(s) connected to the SPI bus. The SPI drives the chip select line to the slave and the serial clock signal (SPCK).

The SPI features two holding registers, the Transmit Data Register and the Receive Data Register, and a single Shift Register. The holding registers maintain the data flow at a constant rate.

After enabling the SPI, a data transfer begins when the processor writes to the SPI_TDR (Transmit Data Register). The written data is immediately transferred in the Shift Register and transfer on the SPI bus starts. While the data in the Shift Register is shifted on the MOSI line, the MISO line is sampled and shifted in the Shift Register. Transmission cannot occur without reception.

Before writting the TDR, the PCS field must be set in order to select a slave.

If new data is written in SPI_TDR during the transfer, it stays in it until the current transfer is completed. Then, the received data is transferred from the Shift Register to SPI_RDR, the data in SPI_TDR is loaded in the Shift Register and a new transfer starts.

The transfer of a data written in SPI_TDR in the Shift Register is indicated by the TDRE bit (Transmit Data Register Empty) in the Status Register (SPI_SR). When new data is written in SPI_TDR, this bit is cleared. The TDRE bit is used to trigger the Transmit PDC channel.

The end of transfer is indicated by the TXEMPTY flag in the SPI_SR register. If a transfer delay (DLYBCT) is greater than 0 for the last transfer, TXEMPTY is set after the completion of said delay. The master clock (MCK) can be switched off at this time.

The transfer of received data from the Shift Register in SPI_RDR is indicated by the RDRF bit (Receive Data Register Full) in the Status Register (SPI_SR). When the received data is read, the RDRF bit is cleared.

If the SPI_RDR (Receive Data Register) has not been read before new data is received, the Overrun Error bit (OVRES) in SPI_SR is set. As long as this flag is set, data is loaded in SPI_RDR. The user has to read the status register to clear the OVRES bit.

Figure 28-7 on page 323 shows a block diagram of the SPI when operating in Master Mode. Figure 28-8 on page 324 shows a flow chart describing how transfers are handled.

28.6.3.1 *Master Mode Block Diagram*

Figure 28-6. Master Mode Block Diagram w/ FDIV

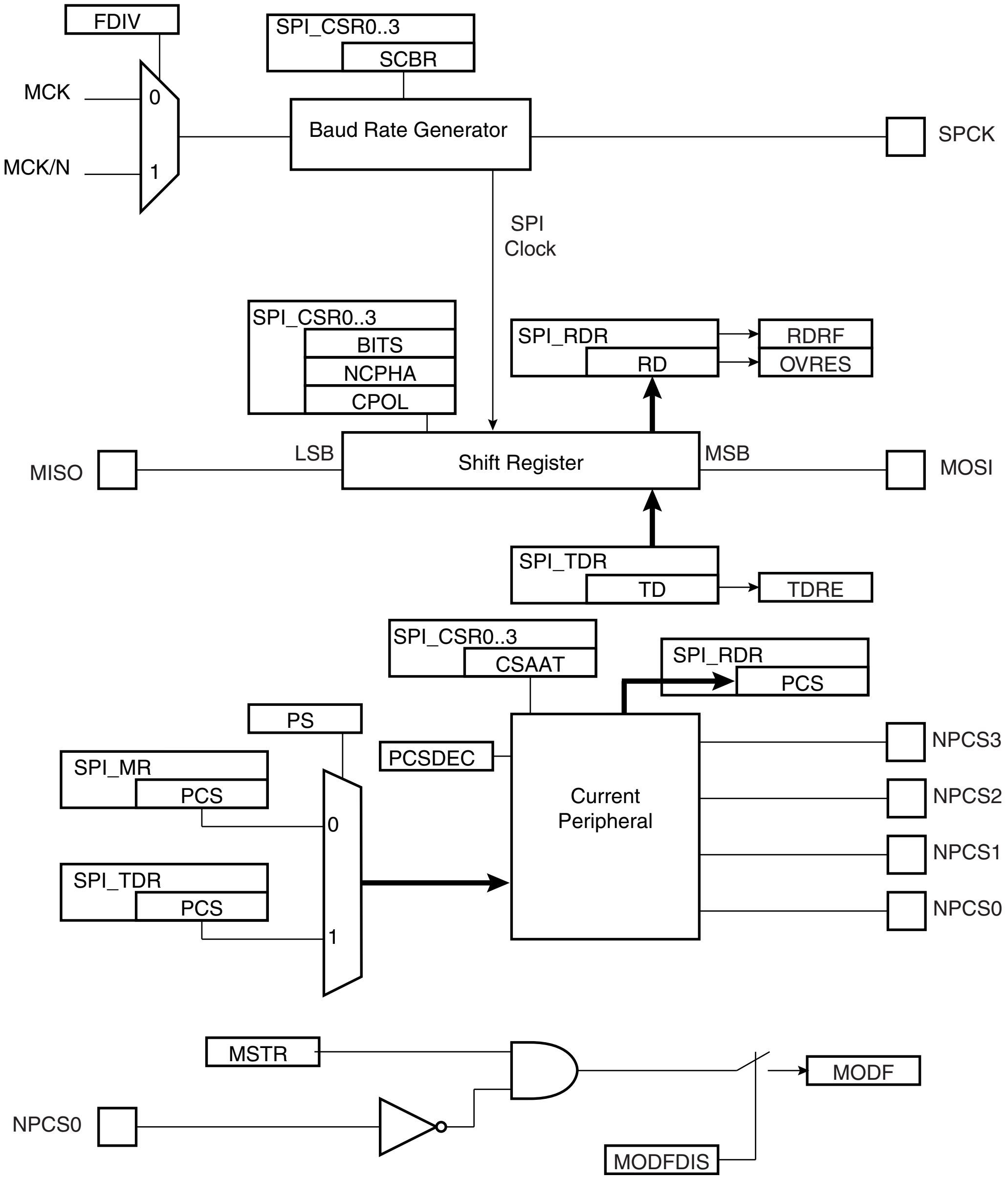
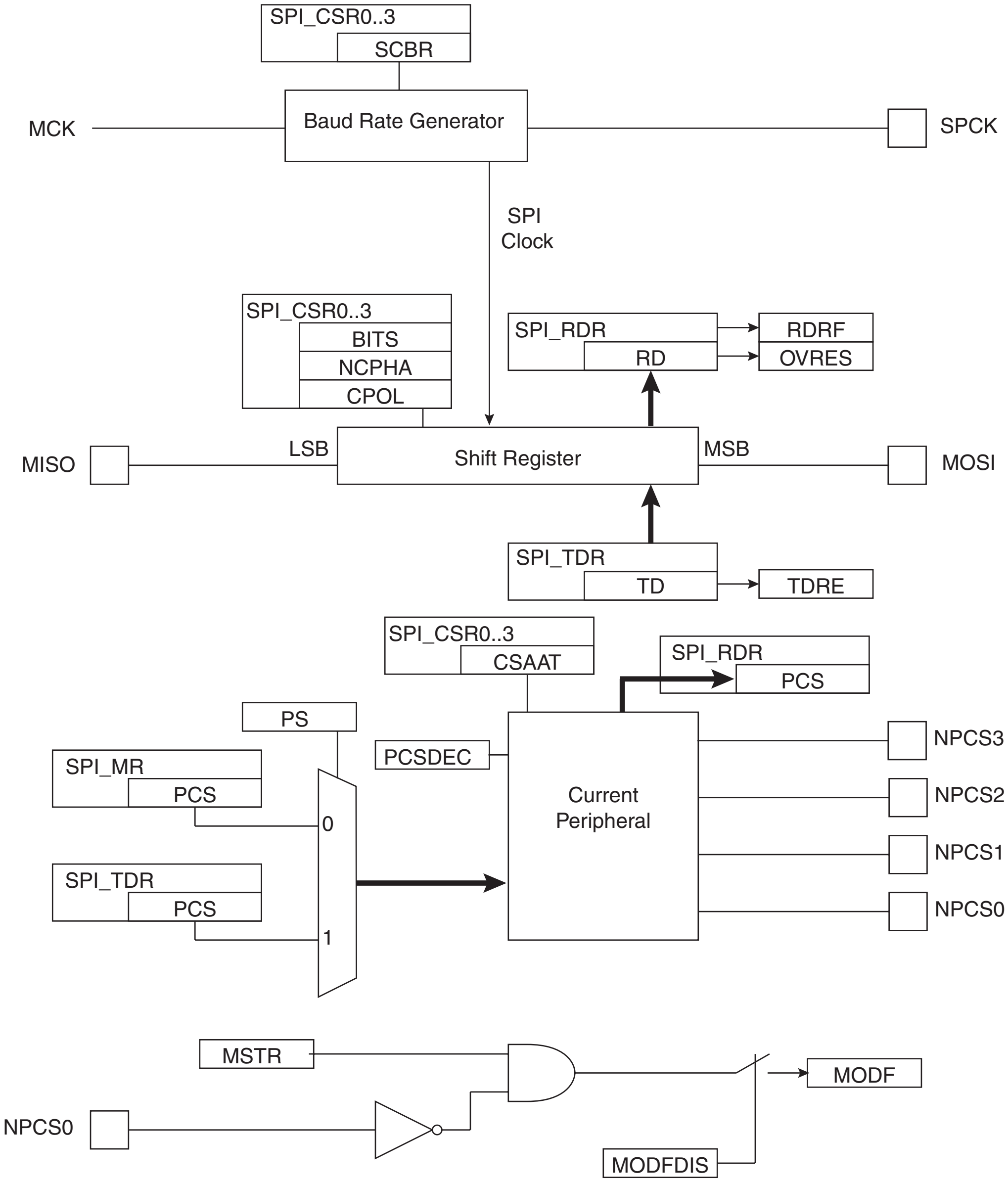
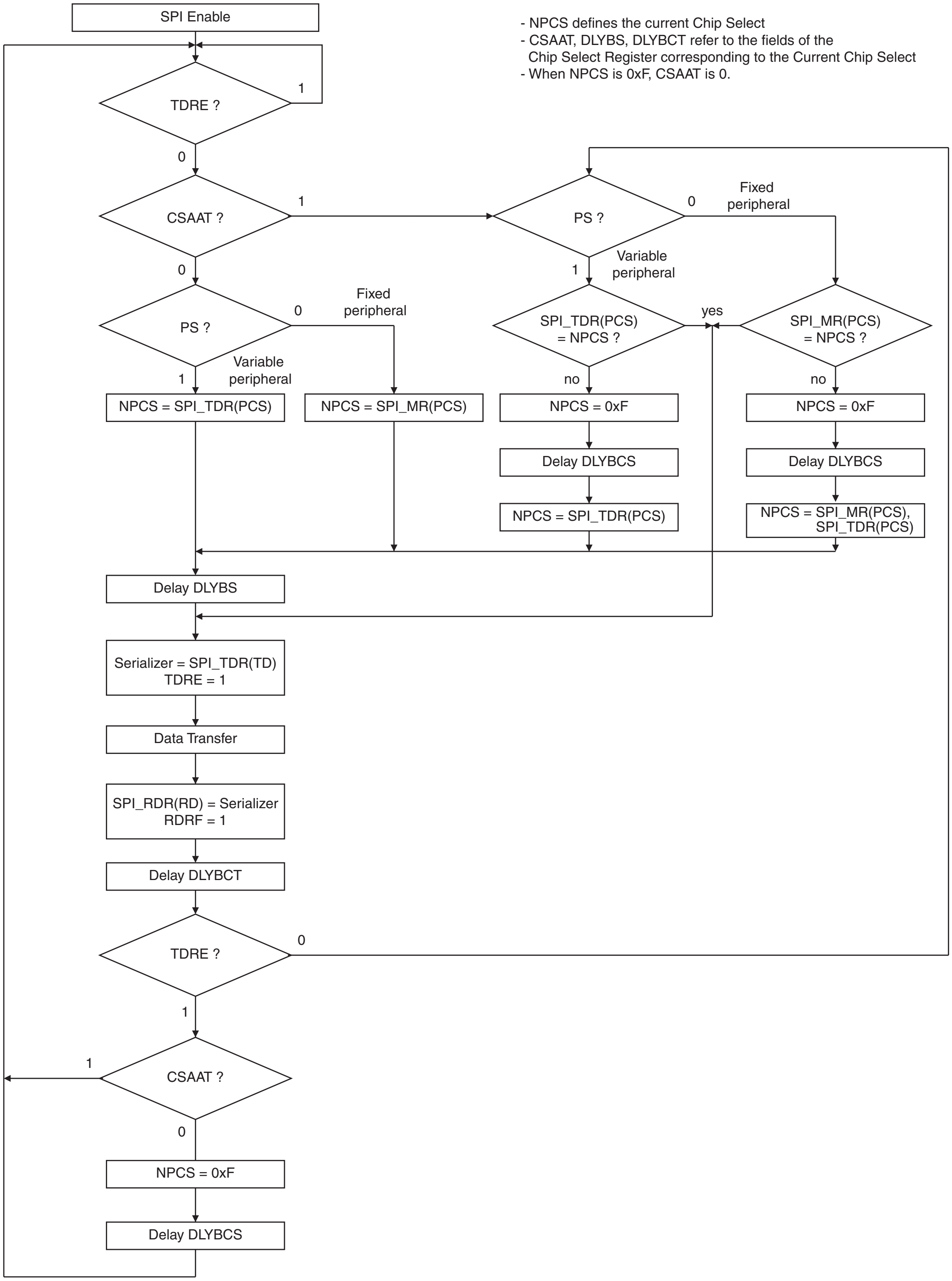

FDIV
SPI_CSR0..3
SCBR
MCK
0
MCK/N
1
Baud Rate Generator
SPCK
SPI
Clock
SPI_CSR0..3
BITS
NCPHA
CPOL
SPI_RDR
RD
RDRF
OVRES
MISO
LSB
Shift Register
MSB
MOSI
SPI_TDR
TD
TDRE
SPI_CSR0..3
CSAAT
SPI_RDR
PCS
PS
SPI_MR
PCS
PCSDEC
Current
Peripheral
NPCS3
NPCS2
NPCS1
NPCS0
0
SPI_TDR
PCS
1
MSTR
MODF
NPCS0
MODFDIS

Figure 28-7. Master Mode Block Diagram w/o FDIV


SPI_CSR0..3
SCBR
MCK
Baud Rate Generator
SPCK
SPI
Clock
SPI_CSR0..3
BITS
NCPHA
CPOL
SPI_RDR
RD
RDRF
OVRES
MISO
LSB
Shift Register
MSB
MOSI
SPI_TDR
TD
TDRE
SPI_CSR0..3
CSAAT
SPI_RDR
PCS
PS
PCSDEC
SPI_MR
PCS
0
SPI_TDR
PCS
1
Current
Peripheral
NPCS3
NPCS2
NPCS1
NPCS0
MSTR
MODF
NPCS0
MODFDIS

28.6.3.2 *Master Mode Flow Diagram*

Figure 28-8. Master Mode Flow Diagram


SPI Enable
- NPCS defines the current Chip Select
- CSAAT, DLYBS, DLYBCT refer to the fields of the Chip Select Register corresponding to the Current Chip Select
- When NPCS is 0xF, CSAAT is 0.
TDRE ?
1
0
CSAAT ?
1
0
PS ?
0
Fixed peripheral
Variable peripheral
1
PS ?
0
Fixed peripheral
Variable peripheral
1
SPI_TDR(PCS) = NPCS ?
yes
SPI_MR(PCS) = NPCS ?
no
no
NPCS = SPI_TDR(PCS)
NPCS = SPI_MR(PCS)
NPCS = 0xF
NPCS = 0xF
Delay DLYBCS
Delay DLYBCS
NPCS = SPI_TDR(PCS)
NPCS = SPI_MR(PCS), SPI_TDR(PCS)
Delay DLYBS
Serializer = SPI_TDR(TD) TDRE = 1
Data Transfer
SPI_RDR(RD) = Serializer RDRF = 1
Delay DLYBCT
TDRE ?
0
1
CSAAT ?
1
0
NPCS = 0xF
Delay DLYBCS

28.6.3.3 Clock Generation

The SPI Baud rate clock is generated by dividing the Master Clock (MCK) or the Master Clock divided by 32, by a value between 1 and 255. The selection between Master Clock or Master Clock divided by 32 is done by the FDIV value set in the Mode Register

This allows a maximum operating baud rate at up to Master Clock and a minimum operating baud rate of MCK divided by 255*32.

Programming the SCBR field at 0 is forbidden. Triggering a transfer while SCBR is at 0 can lead to unpredictable results.

At reset, SCBR is 0 and the user has to program it at a valid value before performing the first transfer.

The divisor can be defined independently for each chip select, as it has to be programmed in the SCBR field of the Chip Select Registers. This allows the SPI to automatically adapt the baud rate for each interfaced peripheral without reprogramming.

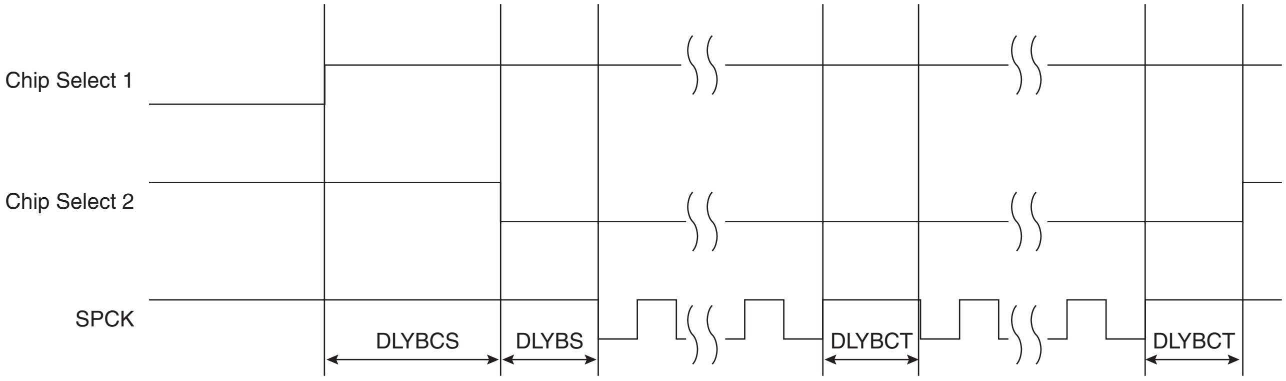
28.6.3.4 Transfer Delays

Figure 28-9 shows a chip select transfer change and consecutive transfers on the same chip select. Three delays can be programmed to modify the transfer waveforms:

- The delay between chip selects, programmable only once for all the chip selects by writing the DLYBCS field in the Mode Register. Allows insertion of a delay between release of one chip select and before assertion of a new one.
- The delay before SPCK, independently programmable for each chip select by writing the field DLYBS. Allows the start of SPCK to be delayed after the chip select has been asserted.
- The delay between consecutive transfers, independently programmable for each chip select by writing the DLYBCT field. Allows insertion of a delay between two transfers occurring on the same chip select

These delays allow the SPI to be adapted to the interfaced peripherals and their speed and bus release time.

Figure 28-9. Programmable Delays



28.6.3.5 Peripheral Selection

The serial peripherals are selected through the assertion of the NPCS0 to NPCS3 signals. By default, all the NPCS signals are high before and after each transfer.

The peripheral selection can be performed 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

Fixed Peripheral Select is activated by writing the PS bit to zero in SPI_MR (Mode Register). In this case, the current peripheral is defined by the PCS field in SPI_MR and the PCS field in the SPI_TDR has no effect.

Variable Peripheral Select is activated by setting PS bit to one. The PCS field in SPI_TDR is used to select the current peripheral. This means that the peripheral selection can be defined for each new data.

The Fixed Peripheral Selection allows buffer transfers with a single peripheral. Using the PDC is an optimal means, as the size of the data transfer between the memory and the SPI is either 8 bits or 16 bits. However, changing the peripheral selection requires the Mode Register to be reprogrammed.

The Variable Peripheral Selection allows buffer transfers with multiple peripherals without reprogramming the Mode Register. Data written in SPI_TDR is 32 bits wide and defines the real data to be transmitted and the peripheral it is destined to. Using the PDC in this mode requires 32-bit wide buffers, with the data in the LSBs and the PCS and LASTXFER fields in the MSBs, however the SPI still controls the number of bits (8 to16) to be transferred through MISO and MOSI lines with the chip select configuration registers. This is not the optimal means in term of memory size for the buffers, but it provides a very effective means to exchange data with several peripherals without any intervention of the processor.

28.6.3.6 Peripheral Chip Select Decoding

The user can program the SPI to operate with up to 15 peripherals by decoding the four Chip Select lines, NPCS0 to NPCS3 with an external logic. This can be enabled by writing the PCSDEC bit at 1 in the Mode Register (SPI_MR).

When operating without decoding, the SPI makes sure that in any case only one chip select line is activated, i.e. driven low at a time. If two bits are defined low in a PCS field, only the lowest numbered chip select is driven low.

When operating with decoding, the SPI directly outputs the value defined by the PCS field of either the Mode Register or the Transmit Data Register (depending on PS).

As the SPI sets a default value of 0xF on the chip select lines (i.e. all chip select lines at 1) when not processing any transfer, only 15 peripherals can be decoded.

The SPI has only four Chip Select Registers, not 15. As a result, when decoding is activated, each chip select defines the characteristics of up to four peripherals. As an example, SPI_CRS0 defines the characteristics of the externally decoded peripherals 0 to 3, corresponding to the PCS values 0x0 to 0x3. Thus, the user has to make sure to connect compatible peripherals on the decoded chip select lines 0 to 3, 4 to 7, 8 to 11 and 12 to 14.

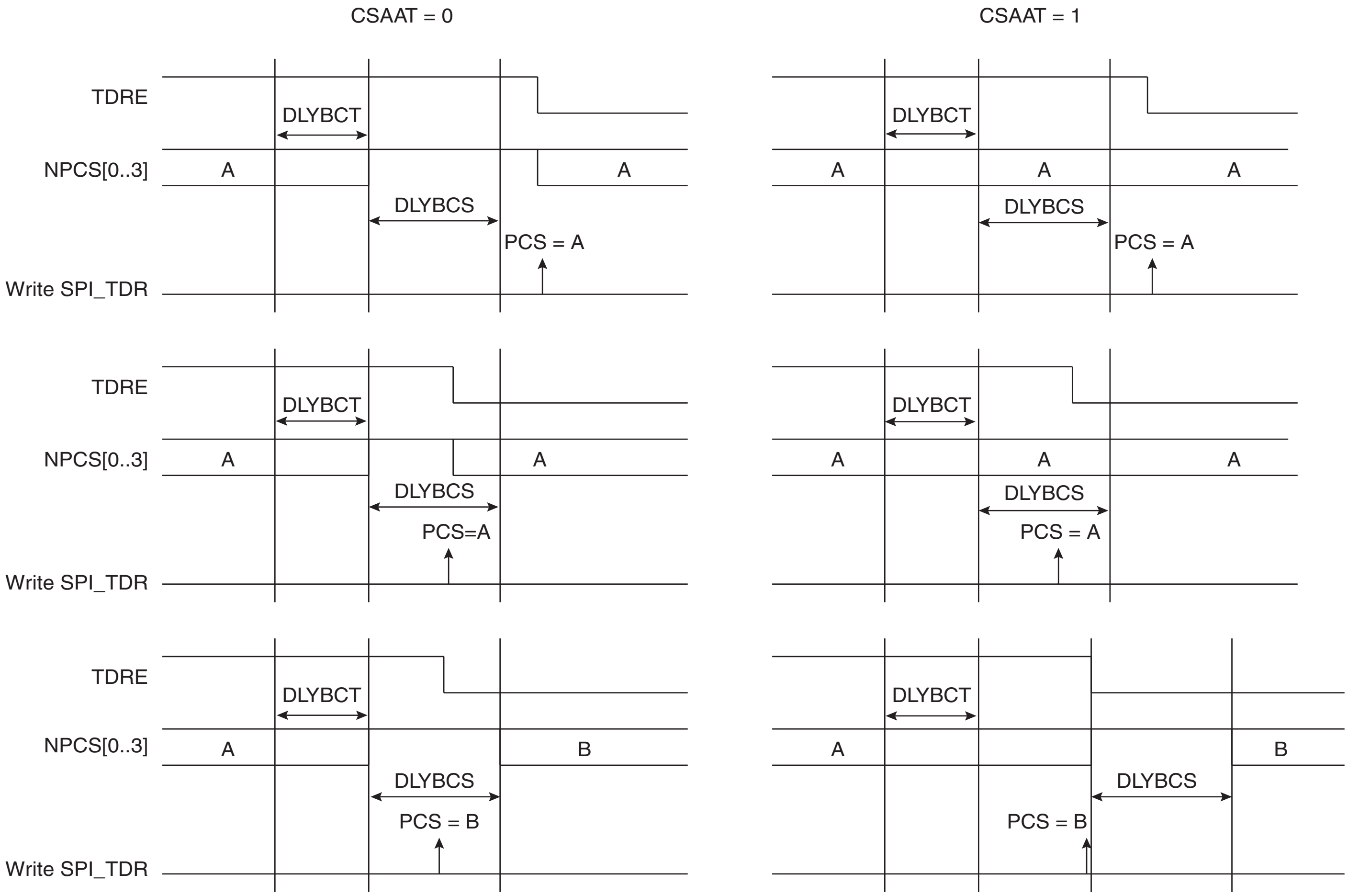
28.6.3.7 Peripheral Deselection

When operating normally, as soon as the transfer of the last data written in SPI_TDR is completed, the NPCS lines all rise. This might lead to runtime error if the processor is too long in responding to an interrupt, and thus might lead to difficulties for interfacing with some serial peripherals requiring the chip select line to remain active during a full set of transfers.

To facilitate interfacing with such devices, the Chip Select Register can be programmed with the CSAAT bit (Chip Select Active After Transfer) at 1. This allows the chip select lines to remain in their current state (low = active) until transfer to another peripheral is required.

Figure 28-10 shows different peripheral deselection cases and the effect of the CSAAT bit.

Figure 28-10. Peripheral Deselection



28.6.3.8 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. NPCS0, MOSI, MISO and SPCK must be configured in open drain through the PIO controller, so that external pull up resistors are needed to guarantee high level.

When a mode fault is detected, the MODF bit in the SPI_SR is set until the SPI_SR is read and the SPI is automatically disabled until re-enabled by writing the SPIEN bit in the SPI_CR (Control Register) at 1.

By default, the Mode Fault detection circuitry is enabled. The user can disable Mode Fault detection by setting the MODFDIS bit in the SPI Mode Register (SPI_MR).

28.6.4 SPI Slave Mode

When operating in Slave Mode, the SPI processes data bits on the clock provided on the SPI clock pin (SPCK).

The SPI waits for NSS to go active before receiving the serial clock from an external master. When NSS falls, the clock is validated on the serializer, which processes the number of bits defined by the BITS field of the Chip Select Register 0 (SPI_CSR0). These bits are processed following a phase and a polarity defined respectively by the NCPHA and CPOL bits of the SPI_CSR0. Note that BITS, CPOL and NCPHA of the other Chip Select Registers have no effect when the SPI is programmed in Slave Mode.

The bits are shifted out on the MISO line and sampled on the MOSI line.

When all the bits are processed, the received data is transferred in the Receive Data Register and the RDRF bit rises. If the SPI_RDR (Receive Data Register) has not been read before new data is received, the Overrun Error bit (OVRES) in SPI_SR is set. As long as this flag is set, data is loaded in SPI_RDR. The user has to read the status register to clear the OVRES bit.

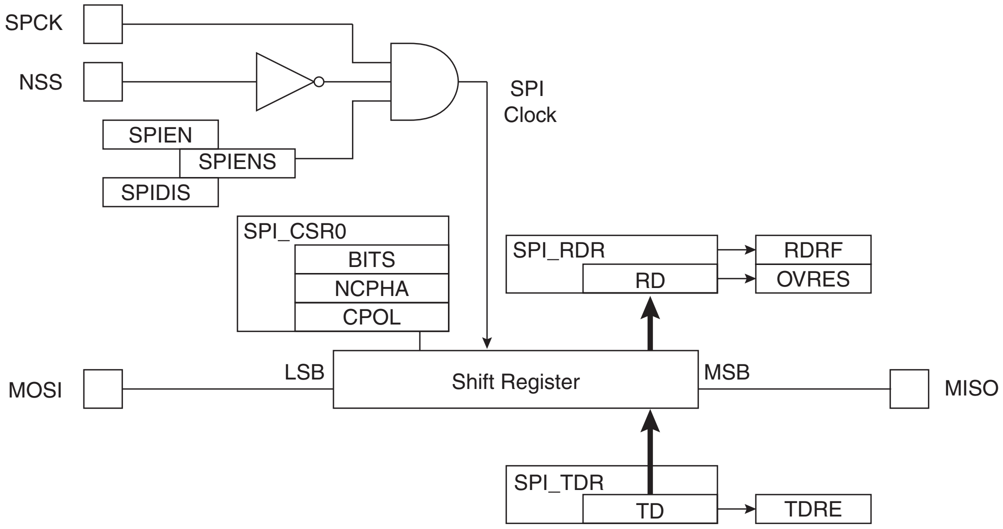
When a transfer starts, the data shifted out is the data present in the Shift Register. If no data has been written in the Transmit Data Register (SPI_TDR), the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift Register resets at 0.

When a first data is written in SPI_TDR, it is transferred immediately in the Shift Register and the TDRE bit rises. If new data is written, it remains in SPI_TDR until a transfer occurs, i.e. NSS falls and there is a valid clock on the SPCK pin. When the transfer occurs, the last data written in SPI_TDR is transferred in the Shift Register and the TDRE bit rises. This enables frequent updates of critical variables with single transfers.

Then, a new data is loaded in the Shift Register from the Transmit Data Register. In case no character is ready to be transmitted, i.e. no character has been written in SPI_TDR since the last load from SPI_TDR to the Shift Register, the Shift Register is not modified and the last received character is retransmitted.

Figure 28-11 shows a block diagram of the SPI when operating in Slave Mode.

Figure 28-11. Slave Mode Functional Block Diagram



28.7 Serial Peripheral Interface (SPI) User Interface

Table 28-3. SPI Register Mapping

Offset	Register	Register Name	Access	Reset
0x00	Control Register	SPI_CR	Write-only	---
0x04	Mode Register	SPI_MR	Read/Write	0x0
0x08	Receive Data Register	SPI_RDR	Read-only	0x0
0x0C	Transmit Data Register	SPI_TDR	Write-only	---
0x10	Status Register	SPI_SR	Read-only	0x00000000(1)
0x14	Interrupt Enable Register	SPI_IER	Write-only	---
0x18	Interrupt Disable Register	SPI_IDR	Write-only	---
0x1C	Interrupt Mask Register	SPI_IMR	Read-only	0x0
0x20 - 0x2C	Reserved			
0x30	Chip Select Register 0	SPI_CSR0	Read/Write	0x0
0x34	Chip Select Register 1	SPI_CSR1	Read/Write	0x0
0x38	Chip Select Register 2	SPI_CSR2	Read/Write	0x0
0x3C	Chip Select Register 3	SPI_CSR3	Read/Write	0x0
0x004C - 0x00F8	Reserved	–	–	–
0x004C - 0x00FC	Reserved	–	–	–
0x100 - 0x124	Reserved for the PDC			

1.Technically, the SPI_SR register is reset to 0x00000000. However, if the SPI clock is enabled, the value may be read as 0x000000F0 right after reset due to the value of the corresponding PDC-related status inputs for register bits 7 down to 4.

28.7.1 SPI Control Register

Name: SPI_CR

Access Type: Write-only

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

As soon as SPIDIS is set, SPI finishes its tranfer.

All pins are set in input mode and no data is received or transmitted.

If a transfer is in progress, the transfer is finished before the SPI is disabled.

If both SPIEN and SPIDIS are equal to one when the control register is written, the SPI is disabled.

- **SWRST: SPI Software Reset**

0 = No effect.

1 = Reset the SPI. A software-triggered hardware reset of the SPI interface is performed.

The SPI is in slave mode after software reset.

PDC channels are not affected by software reset.

- **LASTXFER: Last Transfer**

0 = No effect.

1 = The current NPCS will be deasserted after the character written in TD has been transferred. When CSAAT is set, this allows to close the communication with the current serial peripheral by raising the corresponding NPCS line as soon as TD transfer has completed.

28.7.2 SPI Mode Register

Name: SPI_MR

Access Type: Read/Write

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

- **MSTR: Master/Slave Mode**

0 = SPI is in Slave mode.

1 = SPI is in Master mode.

- **PS: Peripheral Select**

0 = Fixed Peripheral Select.

1 = Variable Peripheral Select.

- **PCSDEC: Chip Select Decode**

0 = The chip selects are directly connected to a peripheral device.

1 = The four chip select lines are connected to a 4- to 16-bit decoder.

When PCSDEC equals one, up to 15 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 15 chip selects according to the following rules:

SPI_CSR0 defines peripheral chip select signals 0 to 3.

SPI_CSR1 defines peripheral chip select signals 4 to 7.

SPI_CSR2 defines peripheral chip select signals 8 to 11.

SPI_CSR3 defines peripheral chip select signals 12 to 14.

- **FDIV: Clock Selection**

0 = The SPI operates at MCK.

1 = The SPI operates at MCK/32.

- **MODFDIS: Mode Fault Detection**

0 = Mode fault detection is enabled.

1 = Mode fault detection is disabled.

- **LLB: Local Loopback Enable**

0 = Local loopback path disabled.

1 = Local loopback path enabled (

LLB controls the local loopback on the data serializer for testing in Master Mode only. (MISO is internally connected on MOSI.)

- **PCS: Peripheral Chip Select**

This field is only used if Fixed Peripheral Select is active (PS = 0).

If PCSDEC = 0:

PCS = xxx0	NPCS[3:0] = 1110
PCS = xx01	NPCS[3:0] = 1101
PCS = x011	NPCS[3:0] = 1011
PCS = 0111	NPCS[3:0] = 0111
PCS = 1111	forbidden (no peripheral is selected)

(x = don’t care)

If PCSDEC = 1:

NPCS[3:0] output signals = PCS.

- **DLYBCS: Delay Between Chip Selects**

This field defines the delay from NPCS inactive to the activation of another NPCS. The DLYBCS time guarantees non-overlapping chip selects and solves bus contentions in case of peripherals having long data float times.

If DLYBCS is less than or equal to six, six MCK periods (or 6*N MCK periods if FDIV is set) will be inserted by default.

Otherwise, the following equation determines the delay:

$$\text{Delay Between Chip Selects} = \frac{DLYBCS}{MCK}$$

If FDIV is 0:

$$\text{Delay Between Chip Selects} = \frac{DLYBCS}{MCK}$$

If FDIV is 1:

$$\text{Delay Between Chip Selects} = \frac{DLYBCS \times N}{MCK}$$

28.7.3 SPI Receive Data Register

Name: SPI_RDR

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
RD							

7	6	5	4	3	2	1	0
RD							

- **RD: Receive Data**

Data received by the SPI Interface is stored in this register right-justified. Unused bits read zero.

- **PCS: Peripheral Chip Select**

In Master Mode only, these bits indicate the value on the NPCS pins at the end of a transfer. Otherwise, these bits read zero.

28.7.4 SPI Transmit Data Register

Name: SPI_TDR

Access Type: Write-only

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

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

15	14	13	12	11	10	9	8
TD							

7	6	5	4	3	2	1	0
TD							

- **TD: Transmit Data**

Data to be transmitted by the SPI Interface is stored in this register. Information to be transmitted must be written to the transmit data register in a right-justified format.

- **PCS: Peripheral Chip Select**

This field is only used if Variable Peripheral Select is active (PS = 1).

If PCSDEC = 0:

PCS = xxx0	NPCS[3:0] = 1110
PCS = xx01	NPCS[3:0] = 1101
PCS = x011	NPCS[3:0] = 1011
PCS = 0111	NPCS[3:0] = 0111
PCS = 1111	forbidden (no peripheral is selected)

(x = don’t care)

If PCSDEC = 1:

NPCS[3:0] output signals = PCS

- **LASTXFER: Last Transfer**

0 = No effect.

1 = The current NPCS will be deasserted after the character written in TD has been transferred. When CSAAT is set, this allows to close the communication with the current serial peripheral by raising the corresponding NPCS line as soon as TD transfer has completed.

This field is only used if Variable Peripheral Select is active (PS = 1).

28.7.5 SPI Status Register

Name: SPI_SR

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR

7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full**

0 = No data has been received since the last read of SPI_RDR

1 = Data has been received and the received data has been transferred from the serializer to SPI_RDR since the last read of SPI_RDR.

- **TDRE: Transmit Data Register Empty**

0 = Data has been written to SPI_TDR and not yet transferred to the serializer.

1 = The last data written in the Transmit Data Register has been transferred to the serializer.

TDRE equals zero when the SPI is disabled or at reset. The SPI enable command sets this bit to one.

- **MODF: Mode Fault Error**

0 = No Mode Fault has been detected since the last read of SPI_SR.

1 = A Mode Fault occurred since the last read of the SPI_SR.

- **OVRES: Overrun Error Status**

0 = No overrun has been detected since the last read of SPI_SR.

1 = An overrun has occurred since the last read of SPI_SR.

An overrun occurs when SPI_RDR is loaded at least twice from the serializer since the last read of the SPI_RDR.

- **ENDRX: End of RX buffer**

0 = The Receive Counter Register has not reached 0 since the last write in SPI_RCR(1) or SPI_RNCR(1).

1 = The Receive Counter Register has reached 0 since the last write in SPI_RCR(1) or SPI_RNCR(1).

- **ENDTX: End of TX buffer**

0 = The Transmit Counter Register has not reached 0 since the last write in SPI_TCR(1) or SPI_TNCR(1).

1 = The Transmit Counter Register has reached 0 since the last write in SPI_TCR(1) or SPI_TNCR(1).

- **RXBUFF: RX Buffer Full**

0 = SPI_RCR(1) or SPI_RNCR(1) has a value other than 0.

1 = Both SPI_RCR(1) and SPI_RNCR(1) have a value of 0.

- **TXBUFE: TX Buffer Empty**

0 = SPI_TCR(1) or SPI_TNCR(1) has a value other than 0.

1 = Both SPI_TCR(1) and SPI_TNCR(1) have a value of 0.

- **NSSR: NSS Rising**

0 = No rising edge detected on NSS pin since last read.

1 = A rising edge occurred on NSS pin since last read.

- **TXEMPTY: Transmission Registers Empty**

0 = As soon as data is written in SPI_TDR.

1 = SPI_TDR and internal shifter are empty. If a transfer delay has been defined, TXEMPTY is set after the completion of such delay.

- **SPIENS: SPI Enable Status**

0 = SPI is disabled.

1 = SPI is enabled.

Note: 1. SPI_RCR, SPI_RNCR, SPI_TCR, SPI_TNCR are physically located in the PDC.

28.7.6 SPI Interrupt Enable Register

Name: SPI_IER

Access Type: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR

7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full Interrupt Enable**
- **TDRE: SPI Transmit Data Register Empty Interrupt Enable**
- **MODF: Mode Fault Error Interrupt Enable**
- **OVRES: Overrun Error Interrupt Enable**
- **ENDRX: End of Receive Buffer Interrupt Enable**
- **ENDTX: End of Transmit Buffer Interrupt Enable**
- **RXBUFF: Receive Buffer Full Interrupt Enable**
- **TXBUFE: Transmit Buffer Empty Interrupt Enable**
- **TXEMPTY: Transmission Registers Empty Enable**
- **NSSR: NSS Rising Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

28.7.7 SPI Interrupt Disable Register

Name: SPI_IDR

Access Type: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR

7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full Interrupt Disable**

- **TDRE: SPI Transmit Data Register Empty Interrupt Disable**

- **MODF: Mode Fault Error Interrupt Disable**

- **OVRES: Overrun Error Interrupt Disable**

- **ENDRX: End of Receive Buffer Interrupt Disable**

- **ENDTX: End of Transmit Buffer Interrupt Disable**

- **RXBUFF: Receive Buffer Full Interrupt Disable**

- **TXBUFE: Transmit Buffer Empty Interrupt Disable**

- **TXEMPTY: Transmission Registers Empty Disable**

- **NSSR: NSS Rising Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

28.7.8 SPI Interrupt Mask Register

Name: SPI_IMR

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR

7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full Interrupt Mask**

- **TDRE: SPI Transmit Data Register Empty Interrupt Mask**

- **MODF: Mode Fault Error Interrupt Mask**

- **OVRES: Overrun Error Interrupt Mask**

- **ENDRX: End of Receive Buffer Interrupt Mask**

- **ENDTX: End of Transmit Buffer Interrupt Mask**

- **RXBUFF: Receive Buffer Full Interrupt Mask**

- **TXBUFE: Transmit Buffer Empty Interrupt Mask**

- **TXEMPTY: Transmission Registers Empty Mask**

- **NSSR: NSS Rising Interrupt Mask**

0 = The corresponding interrupt is not enabled.

1 = The corresponding interrupt is enabled.

28.7.9 SPI Chip Select Register

Name: SPI_CSR0... SPI_CSR3

Access Type: Read/Write

31	30	29	28	27	26	25	24
DLYBCT							

23	22	21	20	19	18	17	16
DLYBS							

15	14	13	12	11	10	9	8
SCBR							

7	6	5	4	3	2	1	0
BITS				CSAAT	–	NCPHA	CPOL

- **CPOL: Clock Polarity**

0 = The inactive state value of SPCK is logic level zero.

1 = The inactive state value of SPCK is logic level one.

CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with NCPHA to produce the required clock/data relationship between master and slave devices.

- **NCPHA: Clock Phase**

0 = Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.

1 = Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.

NCPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. NCPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.

- **CSAAT: Chip Select Active After Transfer**

0 = The Peripheral Chip Select Line rises as soon as the last transfer is achieved.

1 = The Peripheral Chip Select does not rise after the last transfer is achieved. It remains active until a new transfer is requested on a different chip select.

- **BITS: Bits Per Transfer**

The BITS field determines the number of data bits transferred. Reserved values should not be used.

BITS	Bits Per Transfer
0000	8
0001	9
0010	10
0011	11
0100	12
0101	13
0110	14
0111	15
1000	16
1001	Reserved

BITS	Bits Per Transfer
1010	Reserved
1011	Reserved
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

- **SCBR: Serial Clock Baud Rate**

In Master Mode, the SPI Interface uses a modulus counter to derive the SPCK baud rate from the Master Clock MCK. The Baud rate is selected by writing a value from 1 to 255 in the SCBR field. The following equations determine the SPCK baud rate:

$$\text{SPCK Baudrate} = \frac{MCK}{SCBR}$$

If FDIV is 0:

$$\text{SPCK Baudrate} = \frac{MCK}{SCBR}$$

If FDIV is 1:

$$\text{SPCK Baudrate} = \frac{MCK}{(N \times SCBR)}$$

Note: N = 32

Programming the SCBR field at 0 is forbidden. Triggering a transfer while SCBR is at 0 can lead to unpredictable results.

At reset, SCBR is 0 and the user has to program it at a valid value before performing the first transfer.

- **DLYBS: Delay Before SPCK**

This field defines the delay from NPCS valid to the first valid SPCK transition.

When DLYBS equals zero, the NPCS valid to SPCK transition is 1/2 the SPCK clock period.

Otherwise, the following equations determine the delay:

$$\text{Delay Before SPCK} = \frac{DLYBS}{MCK}$$

If FDIV is 0:

$$\text{Delay Before SPCK} = \frac{DLYBS}{MCK}$$

If FDIV is 1:

$$\text{Delay Before SPCK} = \frac{N \times DLYBS}{MCK}$$

Note: N = 32

- **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, no delay between consecutive transfers is inserted and the clock keeps its duty cycle over the character transfers.

Otherwise, the following equation determines the delay:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{MCK}$$

If FDIV is 0:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{MCK}$$

If FDIV is 1:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times N \times DLYBCT}{MCK} + \frac{N \times SCBR}{2MCK}$$

Note: N = 32

29. Universal Synchronous Asynchronous Receiver Transmitter (USART)

29.1 Description

The Universal Synchronous Asynchronous Receiver Transceiver (USART) provides one full duplex universal synchronous asynchronous serial link. Data frame format is widely programmable (data length, parity, number of stop bits) to support a maximum of standards. The receiver implements parity error, framing error and overrun error detection. The receiver time-out enables handling variable-length frames and the transmitter timeguard facilitates communications with slow remote devices. Multidrop communications are also supported through address bit handling in reception and transmission.

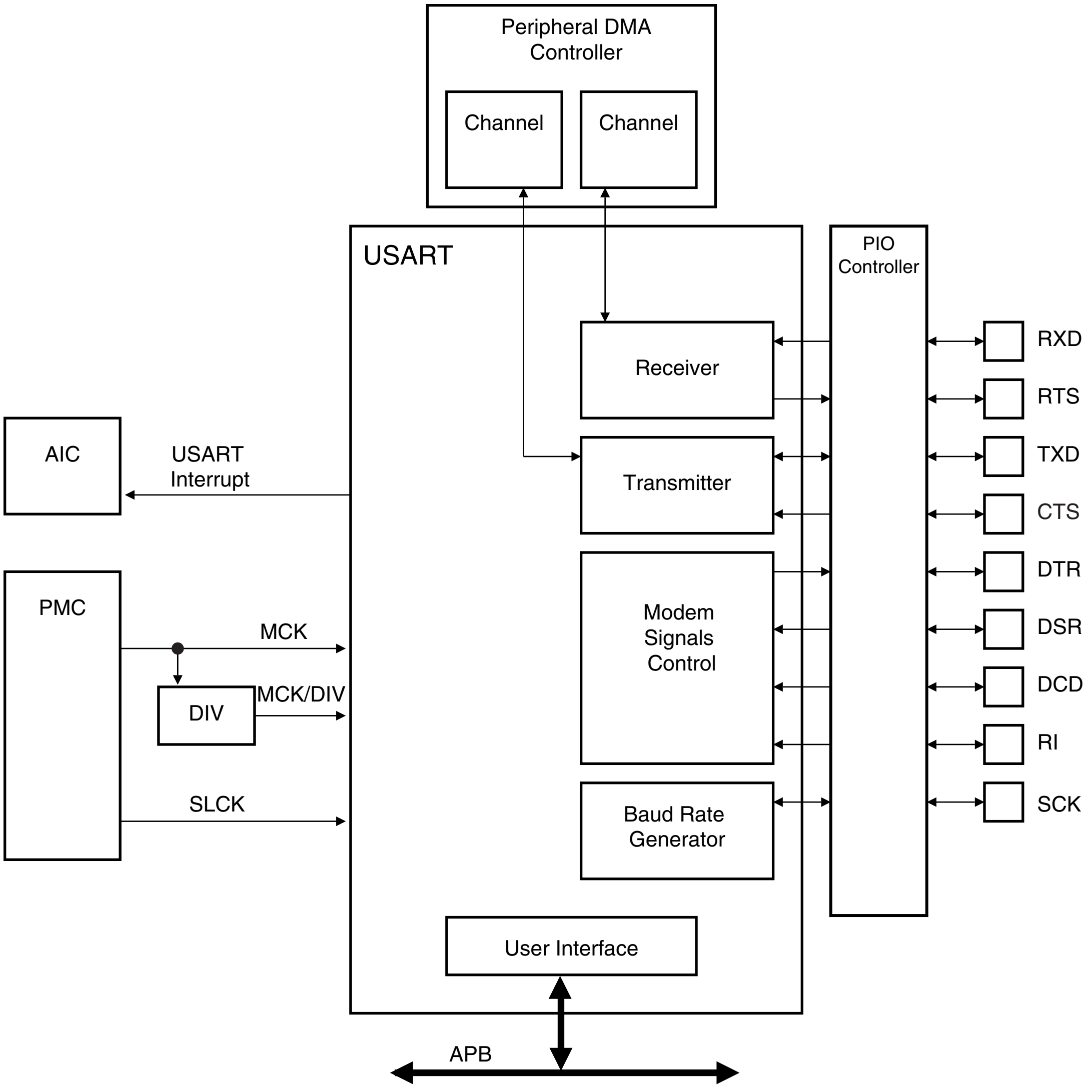
The USART features three test modes: remote loopback, local loopback and automatic echo.

The USART supports specific operating modes providing interfaces on RS485 buses, with ISO7816 T = 0 or T = 1 smart card slots, infrared transceivers and connection to modem ports. The hardware handshaking feature enables an out-of-band flow control by automatic management of the pins RTS and CTS.

The USART supports the connection to the Peripheral DMA Controller, which enables data transfers to the transmitter and from the receiver. The PDC provides chained buffer management without any intervention of the processor.

29.2 Block Diagram

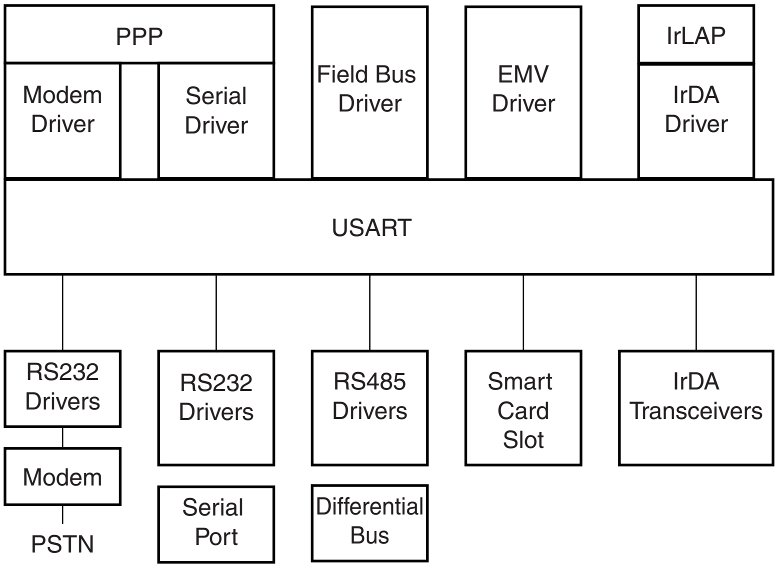
Figure 29-1. USART Block Diagram


Peripheral DMA Controller
Channel
Channel
USART
PIO Controller
Receiver
Transmitter
Modem Signals Control
Baud Rate Generator
User Interface
AIC
USART Interrupt
PMC
MCK
DIV
MCK/DIV
SLCK
APB
RXD
RTS
TXD
CTS
DTR
DSR
DCD
RI
SCK

Note: The following USART0 and USART1 pins are not available through PIO on AT91CAP7E: DTR, DSR, DCD, and RI.

29.3 Application Block Diagram

Figure 29-2. Application Block Diagram


PPP
Modem Driver
Serial Driver
Field Bus Driver
EMV Driver
IrLAP
IrDA Driver
USART
RS232 Drivers
Modem
PSTN
RS232 Drivers
Serial Port
RS485 Drivers
Differential Bus
Smart Card Slot
IrDA Transceivers

29.4 I/O Lines Description

Table 29-1. I/O Line Description

Name	Description	Type	Active Level
SCK	Serial Clock	I/O	
TXD	Transmit Serial Data	I/O	
RXD	Receive Serial Data	Input	
RI	Ring Indicator	Input	Low
DSR	Data Set Ready	Input	Low
DCD	Data Carrier Detect	Input	Low
DTR	Data Terminal Ready	Output	Low
CTS	Clear to Send	Input	Low
RTS	Request to Send	Output	Low

29.5 Product Dependencies

29.5.1 I/O Lines

The pins used for interfacing the USART are multiplexed with the PIO lines. The programmer must first program the PIOA controller to select the USART I/O alternate functions. If I/O lines of the USART are not used by the application, they can be used for other purposes by the PIO Controller.

To prevent the TXD line from falling when the USART is disabled, the use of an internal pull up is mandatory. If the hardware handshaking feature or Modem mode is used, the internal pull up on TXD must also be enabled.

All the pins of the modems may or may not be implemented on the USART. On USARTs not equipped with the corresponding pin, the associated control bits and statuses have no effect on the behavior of the USART.

29.5.2 Power Management

The USART is not continuously clocked. The programmer must first enable the USART Clock in the Power Management Controller (PMC) before using the USART. However, if the application does not require USART operations, the USART clock can be stopped when not needed and be restarted later. In this case, the USART will resume its operations where it left off.

Configuring the USART does not require the USART clock to be enabled.

29.5.3 Interrupt

The USART interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the USART interrupt requires the AIC to be programmed first. Note that it is not recommended to use the USART interrupt line in edge sensitive mode.

29.6 Functional Description

The USART is capable of managing several types of serial synchronous or asynchronous communications.

It supports the following communication modes:

- 5- to 9-bit full-duplex asynchronous serial communication
  - MSB- or LSB-first
  - 1, 1.5 or 2 stop bits
  - Parity even, odd, marked, space or none
  - By 8 or by 16 over-sampling receiver frequency
  - Optional hardware handshaking
  - Optional modem signals management
  - Optional break management
  - Optional multidrop serial communication
- High-speed 5- to 9-bit full-duplex synchronous serial communication
  - MSB- or LSB-first
  - 1 or 2 stop bits
  - Parity even, odd, marked, space or none
  - By 8 or by 16 over-sampling frequency
  - Optional hardware handshaking
  - Optional modem signals management
  - Optional break management
  - Optional multidrop serial communication
- RS485 with driver control signal
- ISO7816, T0 or T1 protocols for interfacing with smart cards
  - NACK handling, error counter with repetition and iteration limit
- InfraRed IrDA Modulation and Demodulation
- Test modes
  - Remote loopback, local loopback, automatic echo

29.6.1 Baud Rate Generator

The Baud Rate Generator provides the bit period clock named the Baud Rate Clock to both the receiver and the transmitter.

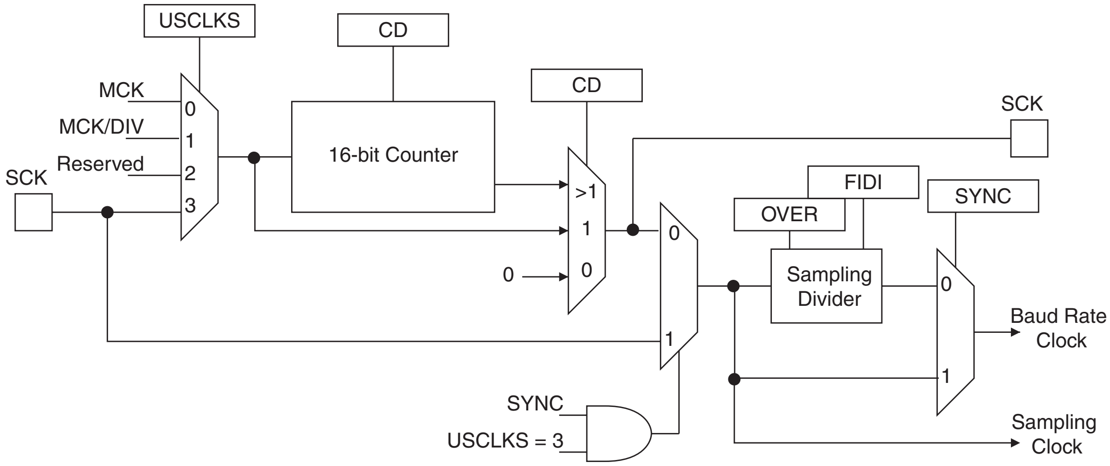
The Baud Rate Generator clock source can be selected by setting the USCLKS field in the Mode Register (US_MR) between:

- the Master Clock MCK
- a division of the Master Clock, the divider being product dependent, but generally set to 8
- the external clock, available on the SCK pin

The Baud Rate Generator is based upon a 16-bit divider, which is programmed with the CD field of the Baud Rate Generator Register (US_BRGR). If CD is programmed at 0, the Baud Rate Generator does not generate any clock. If CD is programmed at 1, the divider is bypassed and becomes inactive.

If the external SCK clock is selected, the duration of the low and high levels of the signal provided on the SCK pin must be longer than a Master Clock (MCK) period. The frequency of the signal provided on SCK must be at least 4.5 times lower than MCK.

Figure 29-3. Baud Rate Generator


USCLKS
CD
CD
MCK
MCK/DIV
Reserved
SCK
0
1
2
3
16-bit Counter
>1
1
0
0
SCK
FIDI
OVER
SYNC
0
1
Sampling Divider
0
1
Baud Rate Clock
SYNC
USCLKS = 3
Sampling Clock

29.6.1.1 Baud Rate in Asynchronous Mode

If the USART is programmed to operate in asynchronous mode, the selected clock is first divided by CD, which is field programmed in the Baud Rate Generator Register (US_BRGR). The resulting clock is provided to the receiver as a sampling clock and then divided by 16 or 8, depending on the programming of the OVER bit in US_MR.

If OVER is set to 1, the receiver sampling is 8 times higher than the baud rate clock. If OVER is cleared, the sampling is performed at 16 times the baud rate clock.

The following formula performs the calculation of the Baud Rate.

$$Baudrate = \frac{SelectedClock}{(8(2 - Over)CD)}$$

This gives a maximum baud rate of MCK divided by 8, assuming that MCK is the highest possible clock and that OVER is programmed at 1.

Baud Rate Calculation Example

Table 29-2 shows calculations of CD to obtain a baud rate at 38400 bauds for different source clock frequencies. This table also shows the actual resulting baud rate and the error.

Table 29-2. Baud Rate Example (OVER = 0)

Source Clock	Expected Baud Rate	Calculation Result	CD	Actual Baud Rate	Error
MHz	Bit/s			Bit/s	
3 686 400	38 400	6.00	6	38 400.00	0.00%
4 915 200	38 400	8.00	8	38 400.00	0.00%
5 000 000	38 400	8.14	8	39 062.50	1.70%
7 372 800	38 400	12.00	12	38 400.00	0.00%
8 000 000	38 400	13.02	13	38 461.54	0.16%
12 000 000	38 400	19.53	20	37 500.00	2.40%
12 288 000	38 400	20.00	20	38 400.00	0.00%
14 318 180	38 400	23.30	23	38 908.10	1.31%
14 745 600	38 400	24.00	24	38 400.00	0.00%
18 432 000	38 400	30.00	30	38 400.00	0.00%
24 000 000	38 400	39.06	39	38 461.54	0.16%
24 576 000	38 400	40.00	40	38 400.00	0.00%
25 000 000	38 400	40.69	40	38 109.76	0.76%
32 000 000	38 400	52.08	52	38 461.54	0.16%
32 768 000	38 400	53.33	53	38 641.51	0.63%
33 000 000	38 400	53.71	54	38 194.44	0.54%
40 000 000	38 400	65.10	65	38 461.54	0.16%
50 000 000	38 400	81.38	81	38 580.25	0.47%

The baud rate is calculated with the following formula:

$$BaudRate = MCK / CD \times 16$$

The baud rate error is calculated with the following formula. It is not recommended to work with an error higher than 5%.

$$Error = 1 - \left(\frac{ExpectedBaudRate}{ActualBaudRate}\right)$$

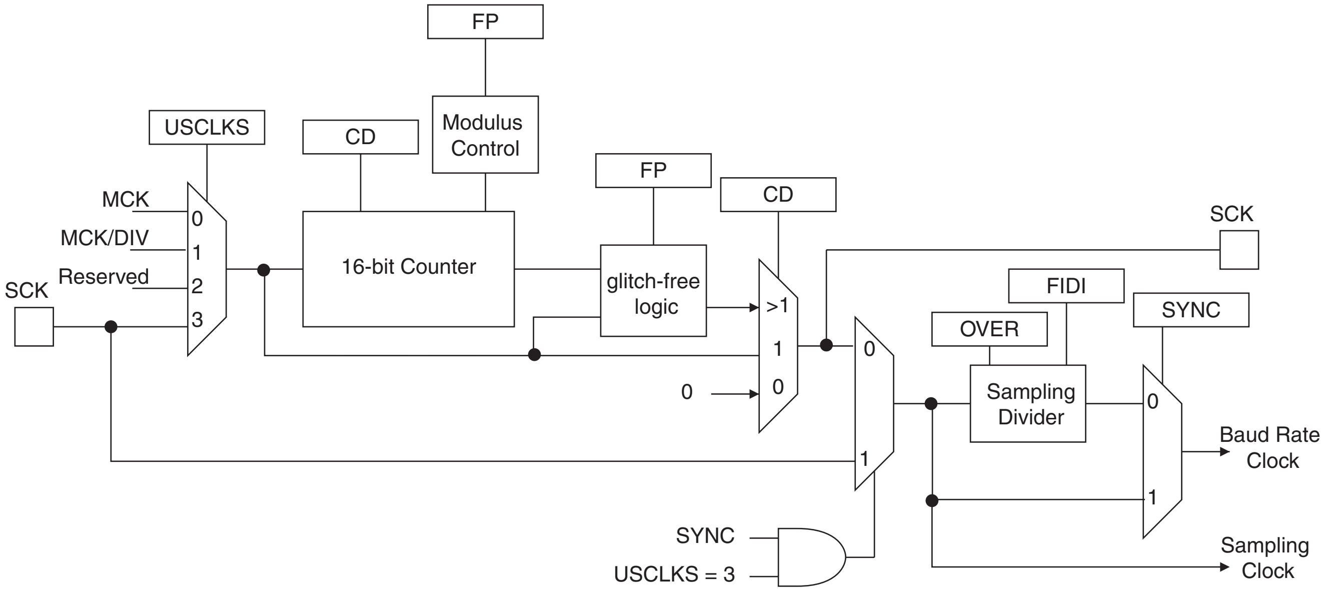
29.6.1.2 *Fractional Baud Rate in Asynchronous Mode*

The Baud Rate generator previously defined is subject to the following limitation: the output frequency changes by only integer multiples of the reference frequency. An approach to this problem is to integrate a fractional N clock generator that has a high resolution. The generator architecture is modified to obtain Baud Rate changes by a fraction of the reference source clock. This fractional part is programmed with the FP field in the Baud Rate Generator Register (US_BRGR). If FP is not 0, the fractional part is activated. The resolution is one eighth of the clock divider. This feature is only available when using USART normal mode. The fractional Baud Rate is calculated using the following formula:

$$Baudrate = \frac{SelectedClock}{\left(8(2 - Over)\left(CD + \frac{FP}{8}\right)\right)}$$

The modified architecture is presented below:

Figure 29-4. Fractional Baud Rate Generator


FP
USCLKS
CD
Modulus Control
FP
CD
MCK
MCK/DIV
Reserved
SCK
0
1
2
3
16-bit Counter
glitch-free logic
>1
1
0
0
SCK
FIDI
OVER
SYNC
0
1
Sampling Divider
0
1
Baud Rate Clock
Sampling Clock
SYNC
USCLKS = 3

29.6.1.3 Baud Rate in Synchronous Mode

If the USART is programmed to operate in synchronous mode, the selected clock is simply divided by the field CD in US_BRGR.

$$BaudRate = \frac{SelectedClock}{CD}$$

In synchronous mode, if the external clock is selected (USCLKS = 3), the clock is provided directly by the signal on the USART SCK pin. No division is active. The value written in US_BRGR has no effect. The external clock frequency must be at least 4.5 times lower than the system clock.

When either the external clock SCK or the internal clock divided (MCK/DIV) is selected, the value programmed in CD must be even if the user has to ensure a 50:50 mark/space ratio on the SCK pin. If the internal clock MCK is selected, the Baud Rate Generator ensures a 50:50 duty cycle on the SCK pin, even if the value programmed in CD is odd.

29.6.1.4 Baud Rate in ISO 7816 Mode

The ISO7816 specification defines the bit rate with the following formula:

$$B = \frac{Di}{Fi} \times f$$

where:

- B is the bit rate
- Di is the bit-rate adjustment factor
- Fi is the clock frequency division factor
- f is the ISO7816 clock frequency (Hz)

Di is a binary value encoded on a 4-bit field, named DI, as represented in Table 29-3.

Table 29-3. Binary and Decimal Values for Di

DI field	0001	0010	0011	0100	0101	0110	1000	1001
Di (decimal)	1	2	4	8	16	32	12	20

Fi is a binary value encoded on a 4-bit field, named FI, as represented in Table 29-4.

Table 29-4. Binary and Decimal Values for Fi

FI field	0000	0001	0010	0011	0100	0101	0110	1001	1010	1011	1100	1101
Fi (decimal	372	372	558	744	1116	1488	1860	512	768	1024	1536	2048

Table 29-5 shows the resulting Fi/Di Ratio, which is the ratio between the ISO7816 clock and the baud rate clock.

Table 29-5. Possible Values for the Fi/Di Ratio

Fi/Di	372	558	774	1116	1488	1806	512	768	1024	1536	2048
1	372	558	744	1116	1488	1860	512	768	1024	1536	2048
2	186	279	372	558	744	930	256	384	512	768	1024
4	93	139.5	186	279	372	465	128	192	256	384	512
8	46.5	69.75	93	139.5	186	232.5	64	96	128	192	256
16	23.25	34.87	46.5	69.75	93	116.2	32	48	64	96	128
32	11.62	17.43	23.25	34.87	46.5	58.13	16	24	32	48	64
12	31	46.5	62	93	124	155	42.66	64	85.33	128	170.6
20	18.6	27.9	37.2	55.8	74.4	93	25.6	38.4	51.2	76.8	102.4

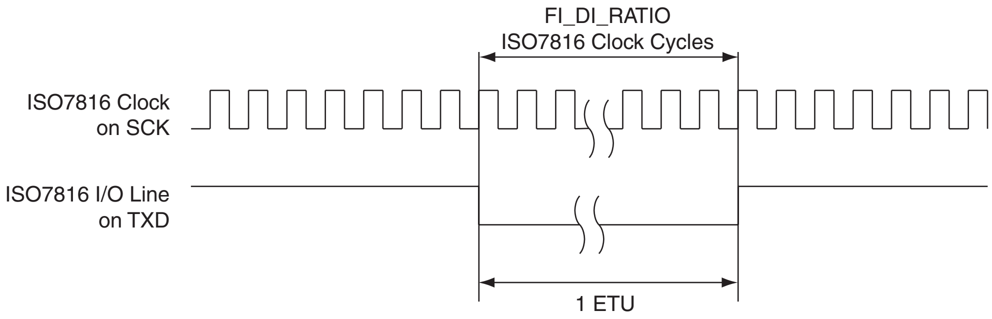
If the USART is configured in ISO7816 Mode, the clock selected by the USCLKS field in the Mode Register (US_MR) is first divided by the value programmed in the field CD in the Baud Rate Generator Register (US_BRGR). The resulting clock can be provided to the SCK pin to feed the smart card clock inputs. This means that the CLKO bit can be set in US_MR.

This clock is then divided by the value programmed in the FI_DI_RATIO field in the FI_DI_Ratio register (US_FIDI). This is performed by the Sampling Divider, which performs a division by up to 2047 in ISO7816 Mode. The non-integer values of the Fi/Di Ratio are not supported and the user must program the FI_DI_RATIO field to a value as close as possible to the expected value.

The FI_DI_RATIO field resets to the value 0x174 (372 in decimal) and is the most common divider between the ISO7816 clock and the bit rate (Fi = 372, Di = 1).

Figure 29-5 shows the relation between the Elementary Time Unit, corresponding to a bit time, and the ISO 7816 clock.

Figure 29-5. Elementary Time Unit (ETU)



29.6.2 Receiver and Transmitter Control

After reset, the receiver is disabled. The user must enable the receiver by setting the RXEN bit in the Control Register (US_CR). However, the receiver registers can be programmed before the receiver clock is enabled.

After reset, the transmitter is disabled. The user must enable it by setting the TXEN bit in the Control Register (US_CR). However, the transmitter registers can be programmed before being enabled.

The Receiver and the Transmitter can be enabled together or independently.

At any time, the software can perform a reset on the receiver or the transmitter of the USART by setting the corresponding bit, RSTRX and RSTTX respectively, in the Control Register (US_CR). The software resets clear the status flag and reset internal state machines but the user interface configuration registers hold the value configured prior to software reset. Regardless of what the receiver or the transmitter is performing, the communication is immediately stopped.

The user can also independently disable the receiver or the transmitter by setting RXDIS and TXDIS respectively in US_CR. If the receiver is disabled during a character reception, the USART waits until the end of reception of the current character, then the reception is stopped. If the transmitter is disabled while it is operating, the USART waits the end of transmission of both the current character and character being stored in the Transmit Holding Register (US_THR). If a timeguard is programmed, it is handled normally.

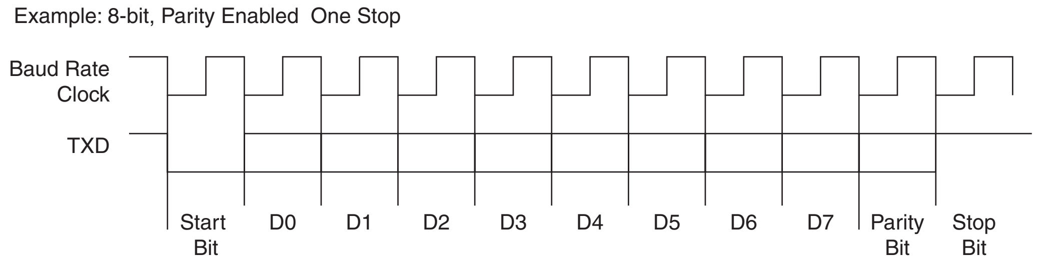
29.6.3 Synchronous and Asynchronous Modes

29.6.3.1 Transmitter Operations

The transmitter performs the same in both synchronous and asynchronous operating modes (SYNC = 0 or SYNC = 1). One start bit, up to 9 data bits, one optional parity bit and up to two stop bits are successively shifted out on the TXD pin at each falling edge of the programmed serial clock.

The number of data bits is selected by the CHRL field and the MODE 9 bit in the Mode Register (US_MR). Nine bits are selected by setting the MODE 9 bit regardless of the CHRL field. The parity bit is set according to the PAR field in US_MR. The even, odd, space, marked or none parity bit can be configured. The MSBF field in US_MR configures which data bit is sent first. If written at 1, the most significant bit is sent first. At 0, the less significant bit is sent first. The number of stop bits is selected by the NBSTOP field in US_MR. The 1.5 stop bit is supported in asynchronous mode only.

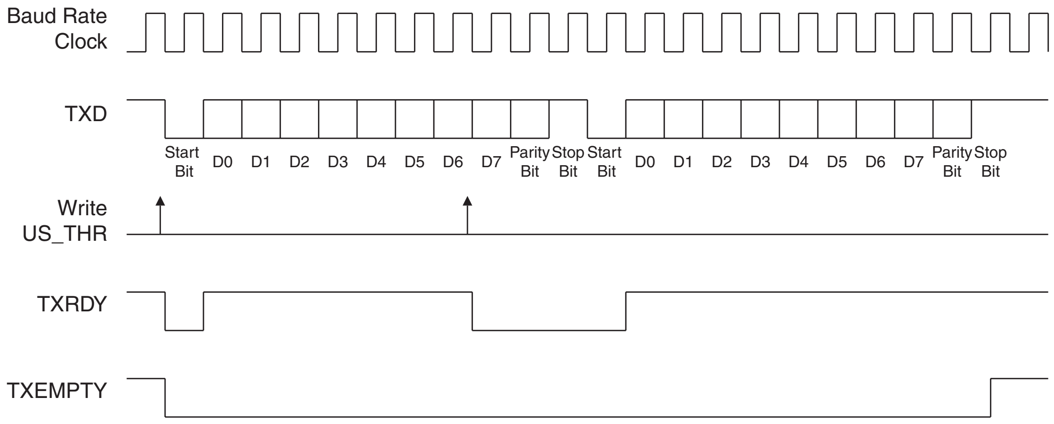
Figure 29-6. Character Transmit


Example: 8-bit, Parity Enabled One Stop
Baud Rate Clock
TXD
Start Bit
D0
D1
D2
D3
D4
D5
D6
D7
Parity Bit
Stop Bit

The characters are sent by writing in the Transmit Holding Register (US_THR). The transmitter reports two status bits in the Channel Status Register (US_CSR): TXRDY (Transmitter Ready), which indicates that US_THR is empty and TXEMPTY, which indicates that all the characters written in US_THR have been processed. When the current character processing is completed, the last character written in US_THR is transferred into the Shift Register of the transmitter and US_THR becomes empty, thus TXRDY raises.

Both TXRDY and TXEMPTY bits are low since the transmitter is disabled. Writing a character in US_THR while TXRDY is active has no effect and the written character is lost.

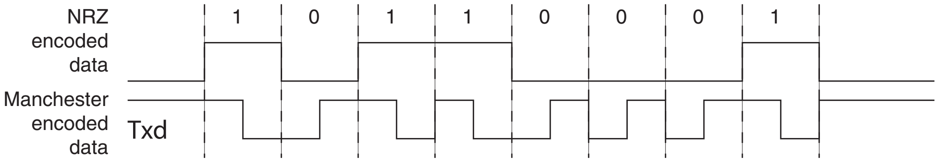
Figure 29-7. Transmitter Status


Baud Rate Clock
TXD
Start Bit
D0
D1
D2
D3
D4
D5
D6
D7
Parity Bit
Stop Bit
Start Bit
D0
D1
D2
D3
D4
D5
D6
D7
Parity Bit
Stop Bit
Write US_THR
TXRDY
TXEMPTY

29.6.3.2 Manchester Encoder

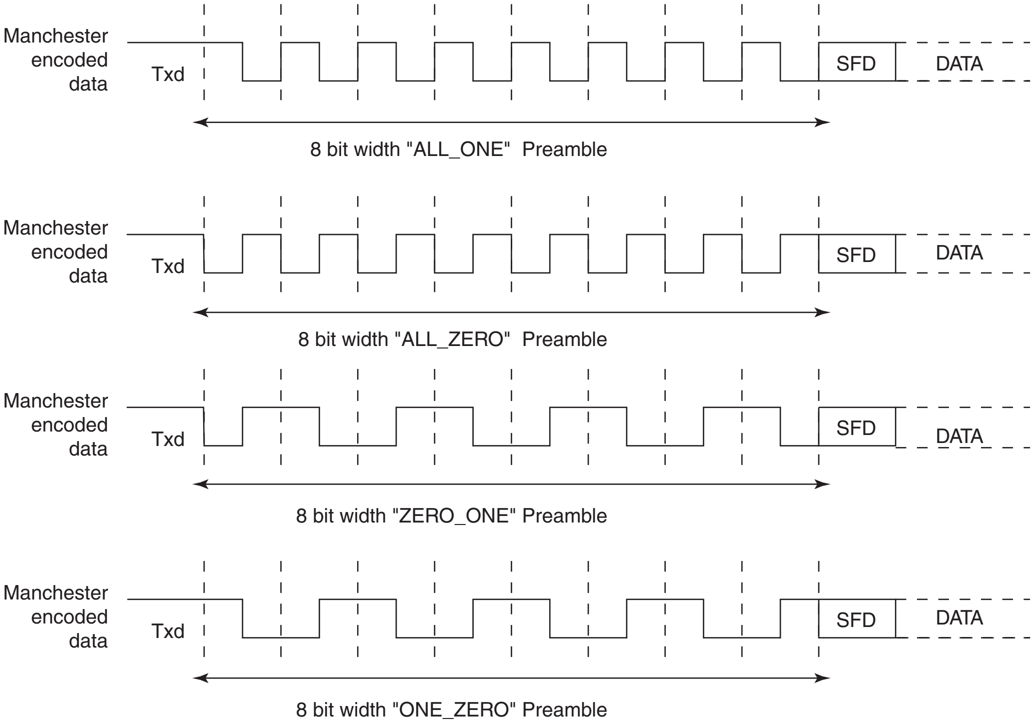
When the Manchester encoder is in use, characters transmitted through the USART are encoded based on biphase Manchester II format. To enable this mode, set the MAN field in the US_MR register to 1. Depending on polarity configuration, a logic level (zero or one), is transmitted as a coded signal one-to-zero or zero-to-one. Thus, a transition always occurs at the midpoint of each bit time. It consumes more bandwidth than the original NRZ signal (2x) but the receiver has more error control since the expected input must show a change at the center of a bit cell. An example of Manchester encoded sequence is: the byte 0xB1 or 10110001 encodes to 10 01 10 10 01 01 01 10, assuming the default polarity of the encoder. Figure 29-8 illustrates this coding scheme.

Figure 29-8. NRZ to Manchester Encoding


NRZ encoded data
Manchester encoded data
Txd
1
0
1
1
0
0
0
1

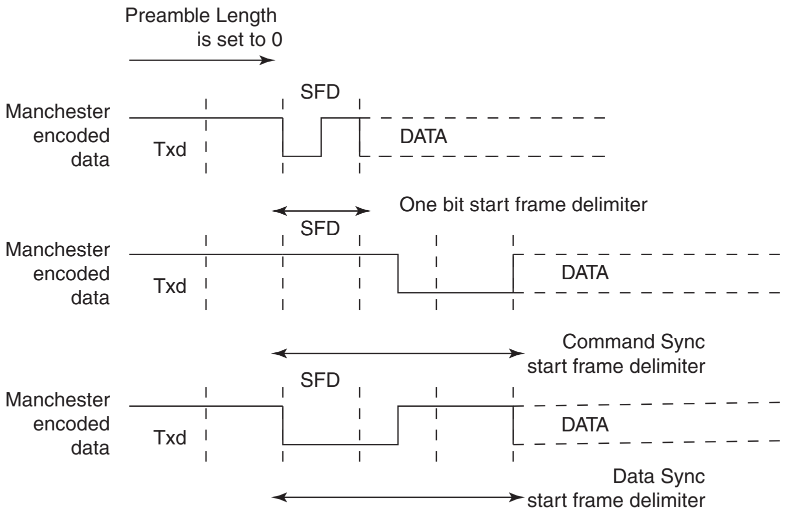
The Manchester encoded character can also be encapsulated by adding both a configurable preamble and a start frame delimiter pattern. Depending on the configuration, the preamble is a training sequence, composed of a pre-defined pattern with a programmable length from 1 to 15 bit times. If the preamble length is set to 0, the preamble waveform is not generated prior to any character. The preamble pattern is chosen among the following sequences: ALL_ONE, ALL_ZERO, ONE_ZERO or ZERO_ONE, writing the field TX_PP in the US_MAN register, the field TX_PL is used to configure the preamble length. Figure 29-9 illustrates and defines the valid patterns. To improve flexibility, the encoding scheme can be configured using the TX_MPOL field in the US_MAN register. If the TX_MPOL field is set to zero (default), a logic zero is encoded with a zero-to-one transition and a logic one is encoded with a one-to-zero transition. If the TX_MPOL field is set to one, a logic one is encoded with a one-to-zero transition and a logic zero is encoded with a zero-to-one transition.

Figure 29-9. Preamble Patterns, Default Polarity Assumed


Manchester encoded data
Txd
SFD
DATA
8 bit width "ALL_ONE" Preamble
Manchester encoded data
Txd
SFD
DATA
8 bit width "ALL_ZERO" Preamble
Manchester encoded data
Txd
SFD
DATA
8 bit width "ZERO_ONE" Preamble
Manchester encoded data
Txd
SFD
DATA
8 bit width "ONE_ZERO" Preamble

A start frame delimiter is to be configured using the ONEBIT field in the US_MR register. It consists of a user-defined pattern that indicates the beginning of a valid data. Figure 29-10 illustrates these patterns. If the start frame delimiter, also known as start bit, is one bit, (ONEBIT at 1), a logic zero is Manchester encoded and indicates that a new character is being sent serially on the line. If the start frame delimiter is a synchronization pattern also referred to as sync (ONEBIT at 0), a sequence of 3 bit times is sent serially on the line to indicate the start of a new character. The sync waveform is in itself an invalid Manchester waveform as the transition occurs at the middle of the second bit time. Two distinct sync patterns are used: the command sync and the data sync. The command sync has a logic one level for one and a half bit times, then a transition to logic zero for the second one and a half bit times. If the MODSYNC field in the US_MR register is set to 1, the next character is a command. If it is set to 0, the next character is a data. When direct memory access is used, the MODSYNC field can be immediately updated with a modified character located in memory. To enable this mode, VAR_SYNC field in US_MR register must be set to 1. In this case, the MODSYNC field in US_MR is bypassed and the sync configuration is held in the TXSYNH in the US_THR register. The USART character format is modified and includes sync information.

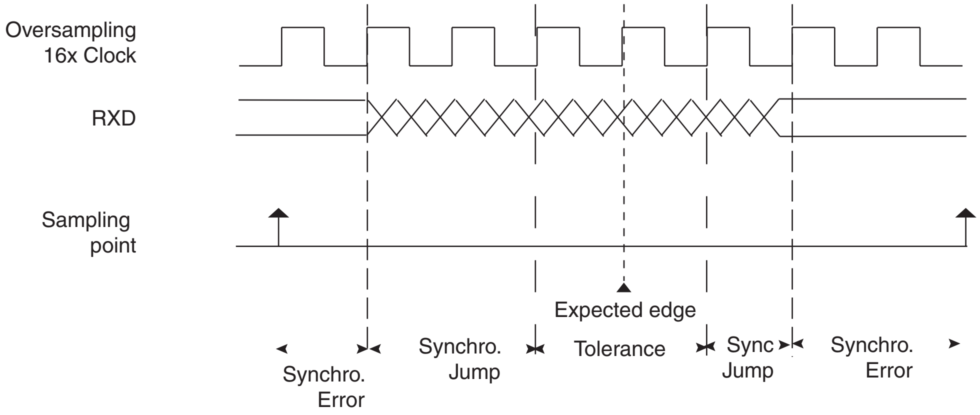
Figure 29-10. Start Frame Delimiter


Preamble Length
is set to 0
SFD
Manchester
encoded
data
Txd
DATA
One bit start frame delimiter
SFD
Manchester
encoded
data
Txd
DATA
Command Sync
start frame delimiter
SFD
Manchester
encoded
data
Txd
DATA
Data Sync
start frame delimiter

Drift Compensation

Drift compensation is available only in 16X oversampling mode. An hardware recovery system allows a larger clock drift. To enable the hardware system, the bit in the USART_MAN register must be set. If the RXD edge is one 16X clock cycle from the expected edge, this is considered as normal jitter and no corrective actions is taken. If the RXD event is between 4 and 2 clock cycles before the expected edge, then the current period is shortened by one clock cycle. If the RXD event is between 2 and 3 clock cycles after the expected edge, then the current period is lengthened by one clock cycle. These intervals are considered to be drift and so corrective actions are automatically taken.

Figure 29-11. Bit Resynchronization



29.6.3.3 *Asynchronous Receiver*

If the USART is programmed in asynchronous operating mode (SYNC = 0), the receiver oversamples the RXD input line. The oversampling is either 16 or 8 times the Baud Rate clock, depending on the OVER bit in the Mode Register (US_MR).

The receiver samples the RXD line. If the line is sampled during one half of a bit time at 0, a start bit is detected and data, parity and stop bits are successively sampled on the bit rate clock.

If the oversampling is 16, (OVER at 0), a start is detected at the eighth sample at 0. Then, data bits, parity bit and stop bit are sampled on each 16 sampling clock cycle. If the oversampling is 8 (OVER at 1), a start bit is detected at the fourth sample at 0. Then, data bits, parity bit and stop bit are sampled on each 8 sampling clock cycle.

The number of data bits, first bit sent and parity mode are selected by the same fields and bits as the transmitter, i.e. respectively CHRL, MODE9, MSBF and PAR. For the synchronization mechanism **only**, the number of stop bits has no effect on the receiver as it considers only one stop bit, regardless of the field NBSTOP, so that resynchronization between the receiver and the transmitter can occur. Moreover, as soon as the stop bit is sampled, the receiver starts looking for a new start bit so that resynchronization can also be accomplished when the transmitter is operating with one stop bit.

Figure 29-12 and Figure 29-13 illustrate start detection and character reception when USART operates in asynchronous mode.

Figure 29-12. Asynchronous Start Detection

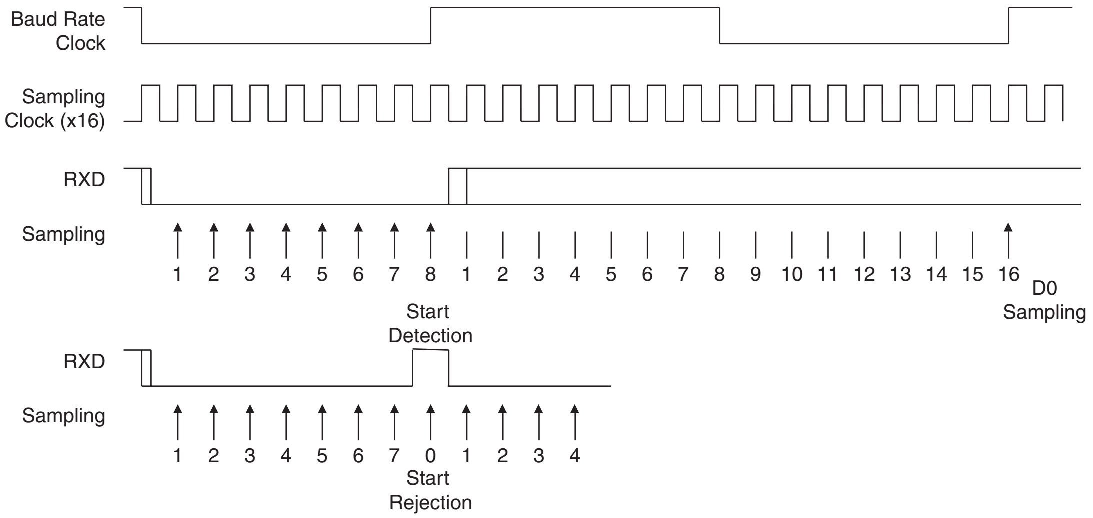
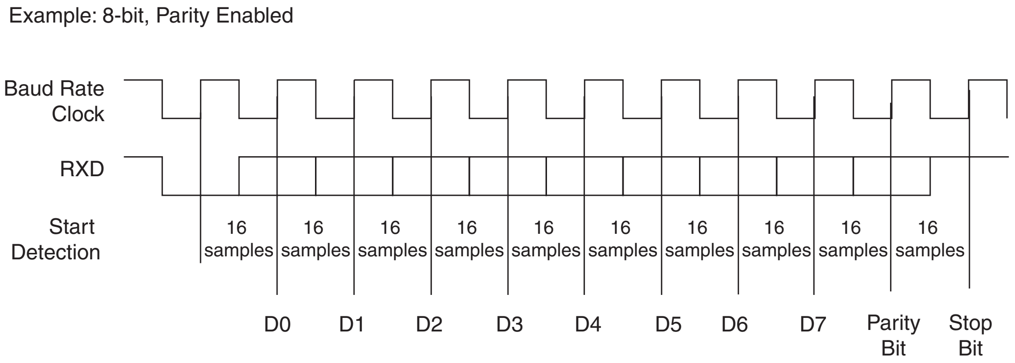

Baud Rate Clock
Sampling Clock (x16)
RXD
Sampling
1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Start Detection
D0 Sampling
RXD
Sampling
1 2 3 4 5 6 7 0 1 2 3 4
Start Rejection

Figure 29-13. Asynchronous Character Reception


Example: 8-bit, Parity Enabled
Baud Rate Clock
RXD
Start Detection
16 samples
16 samples
16 samples
16 samples
16 samples
16 samples
16 samples
16 samples
16 samples
16 samples
D0
D1
D2
D3
D4
D5
D6
D7
Parity Bit
Stop Bit

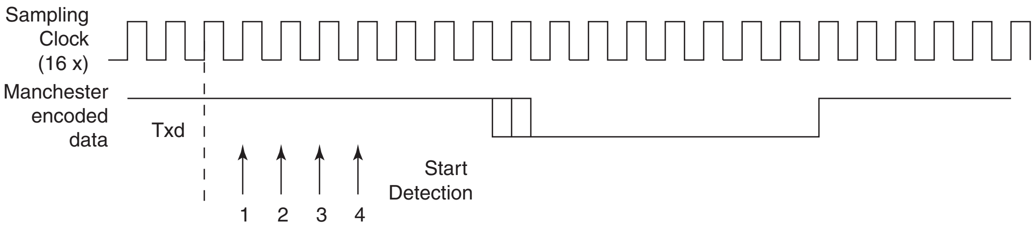
29.6.3.4 *Manchester Decoder*

When the MAN field in US_MR register is set to 1, the Manchester decoder is enabled. The decoder performs both preamble and start frame delimiter detection. One input line is dedicated to Manchester encoded input data.

An optional preamble sequence can be defined, its length is user-defined and totally independent of the emitter side. Use RX_PL in US_MAN register to configure the length of the preamble sequence. If the length is set to 0, no preamble is detected and the function is disabled. In addition, the polarity of the input stream is programmable with RX_MPOL field in US_MAN register. Depending on the desired application the preamble pattern matching is to be defined via the RX_PP field in US_MAN. See Figure 29-9 for available preamble patterns.

Unlike preamble, the start frame delimiter is shared between Manchester Encoder and Decoder. So, if ONEBIT field is set to 1, only a zero encoded Manchester can be detected as a valid start frame delimiter. If ONEBIT is set to 0, only a sync pattern is detected as a valid start frame delimiter. Decoder operates by detecting transition on incoming stream. If RXD is sampled during one quarter of a bit time at zero, a start bit is detected. See Figure 29-14.. The sample pulse rejection mechanism applies.

Figure 29-14. Asynchronous Start Bit Detection

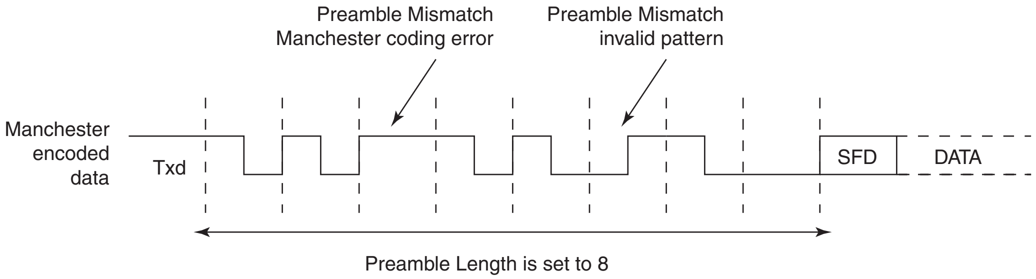


Sampling Clock (16 x)
Manchester encoded data
Txd
Start Detection
1
2
3
4

The receiver is activated and starts Preamble and Frame Delimiter detection, sampling the data at one quarter and then three quarters. If a valid preamble pattern or start frame delimiter is detected, the receiver continues decoding with the same synchronization. If the stream does not match a valid pattern or a valid start frame delimiter, the receiver re-synchronizes on the next valid edge.The minimum time threshold to estimate the bit value is three quarters of a bit time.

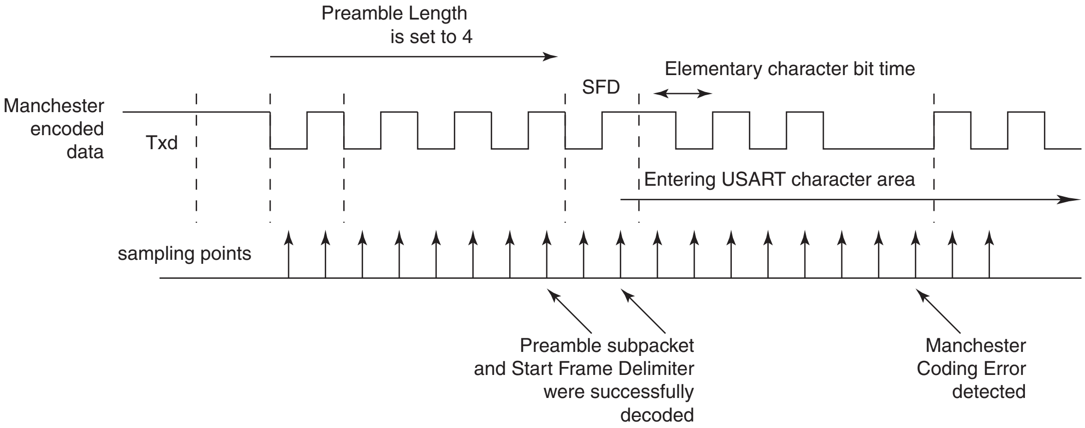
If a valid preamble (if used) followed with a valid start frame delimiter is detected, the incoming stream is decoded into NRZ data and passed to USART for processing. Figure 29-15 illustrates Manchester pattern mismatch. When incoming data stream is passed to the USART, the receiver is also able to detect Manchester code violation. A code violation is a lack of transition in the middle of a bit cell. In this case, MANE flag in US_CSR register is raised. It is cleared by writing the Control Register (US_CR) with the RSTSTA bit at 1. See Figure 29-16 for an example of Manchester error detection during data phase.

Figure 29-15. Preamble Pattern Mismatch



Preamble Mismatch Manchester coding error
Preamble Mismatch invalid pattern
Manchester encoded data
Txd
SFD
DATA
Preamble Length is set to 8

Figure 29-16. Manchester Error Flag



Preamble Length is set to 4
SFD
Elementary character bit time
Manchester encoded data
Txd
Entering USART character area
sampling points
Preamble subpacket and Start Frame Delimiter were successfully decoded
Manchester Coding Error detected

When the start frame delimiter is a sync pattern (ONEBIT field at 0), both command and data delimiter are supported. If a valid sync is detected, the received character is written as RXCHR field in the US_RHR register and the RXSYNH is updated. RXCHR is set to 1 when the received character is a command, and it is set to 0 if the received character is a data. This mechanism alleviates and simplifies the direct memory access as the character contains its own sync field in the same register.

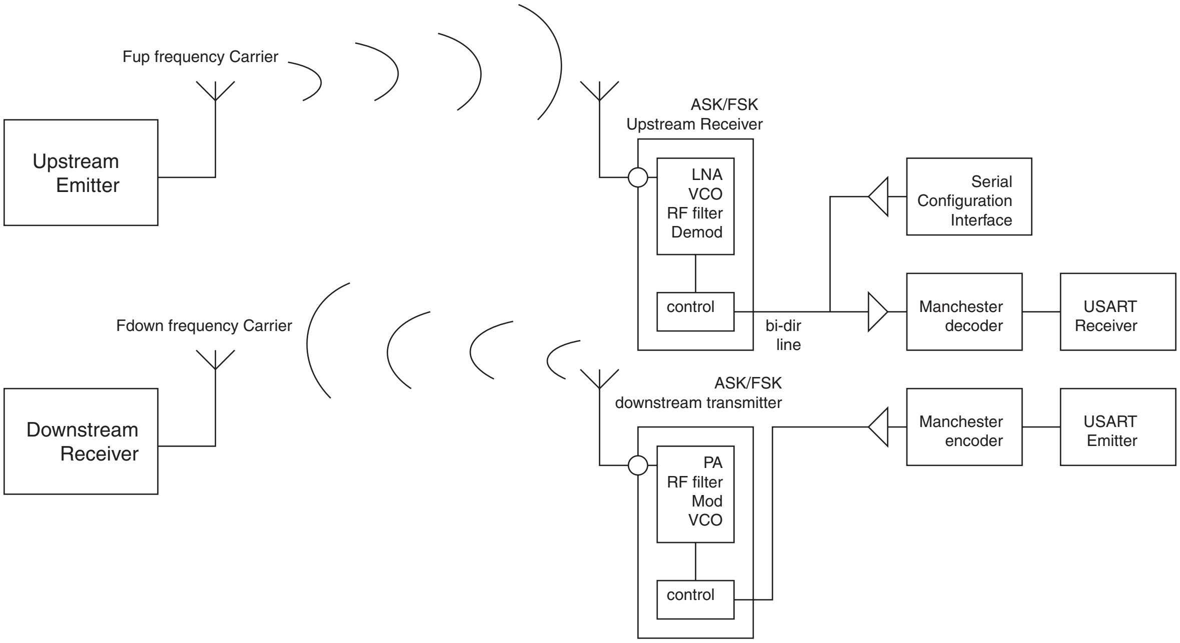
As the decoder is setup to be used in unipolar mode, the first bit of the frame has to be a zero-to-one transition.

29.6.3.5 Radio Interface: Manchester Encoded USART Application

This section describes low data rate RF transmission systems and their integration with a Manchester encoded USART. These systems are based on transmitter and receiver ICs that support ASK and FSK modulation schemes.

The goal is to perform full duplex radio transmission of characters using two different frequency carriers. See the configuration in Figure 29-17.

Figure 29-17. Manchester Encoded Characters RF Transmission


Fup frequency Carrier
Upstream Emitter
ASK/FSK Upstream Receiver
LNA
VCO
RF filter
Demod
control
Serial Configuration Interface
Manchester decoder
USART Receiver
bi-dir line
Fdown frequency Carrier
Downstream Receiver
ASK/FSK downstream transmitter
PA
RF filter
Mod
VCO
control
Manchester encoder
USART Emitter

The USART module is configured as a Manchester encoder/decoder. Looking at the downstream communication channel, Manchester encoded characters are serially sent to the RF emitter. This may also include a user defined preamble and a start frame delimiter. Mostly, preamble is used in the RF receiver to distinguish between a valid data from a transmitter and signals due to noise. The Manchester stream is then modulated. See Figure 29-18 for an example of ASK modulation scheme. When a logic one is sent to the ASK modulator, the power amplifier, referred to as PA, is enabled and transmits an RF signal at downstream frequency. When a logic zero is transmitted, the RF signal is turned off. If the FSK modulator is activated, two different frequencies are used to transmit data. When a logic 1 is sent, the modulator outputs an RF signal at frequency F0 and switches to F1 if the data sent is a 0. See Figure 29-19.

From the receiver side, another carrier frequency is used. The RF receiver performs a bit check operation examining demodulated data stream. If a valid pattern is detected, the receiver switches to receiving mode. The demodulated stream is sent to the Manchester decoder. Because of bit checking inside RF IC, the data transferred to the microcontroller is reduced by a user-defined number of bits. The Manchester preamble length is to be defined in accordance with the RF IC configuration.

Figure 29-18. ASK Modulator Output

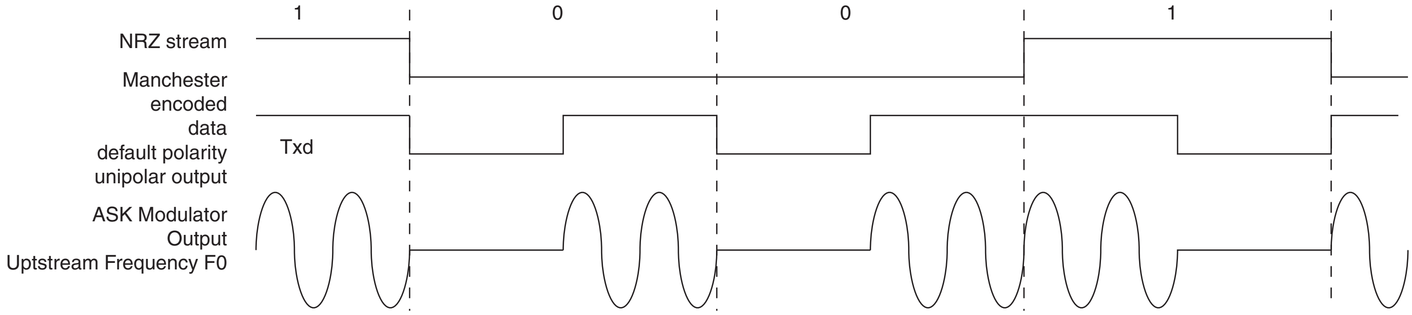
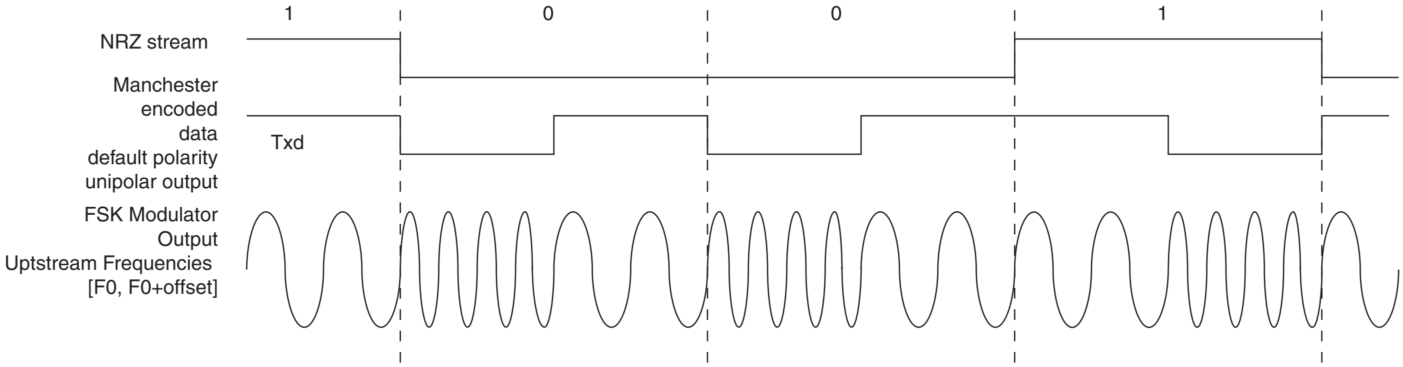

1
0
0
1
NRZ stream
Manchester encoded data default polarity unipolar output
Txd
ASK Modulator Output Uptstream Frequency F0

Figure 29-19. FSK Modulator Output


1
0
0
1
NRZ stream
Manchester encoded data default polarity unipolar output
Txd
FSK Modulator Output Uptstream Frequencies [F0, F0+offset]

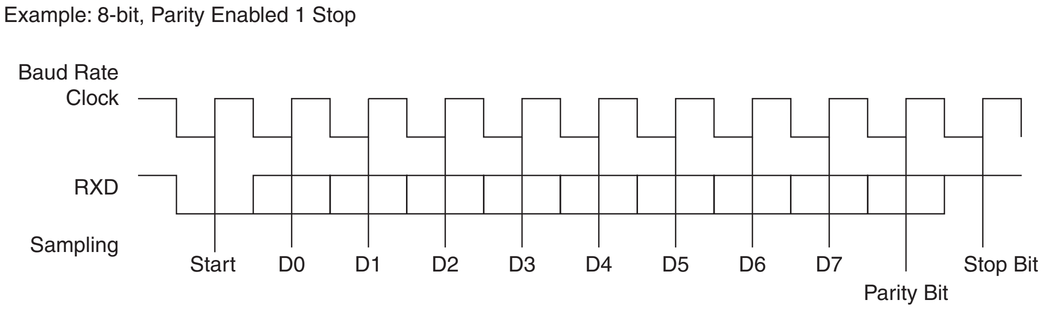
29.6.3.6 *Synchronous Receiver*

In synchronous mode (SYNC = 1), the receiver samples the RXD signal on each rising edge of the Baud Rate Clock. If a low level is detected, it is considered as a start. All data bits, the parity bit and the stop bits are sampled and the receiver waits for the next start bit. Synchronous mode operations provide a high speed transfer capability.

Configuration fields and bits are the same as in asynchronous mode.

Figure 29-20 illustrates a character reception in synchronous mode.

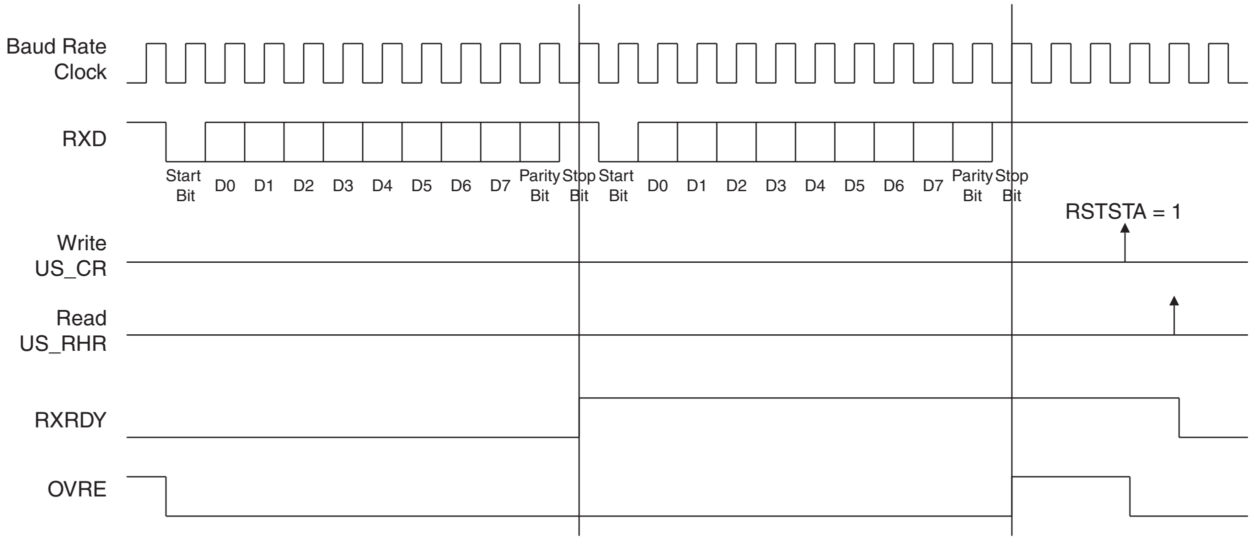
Figure 29-20. Synchronous Mode Character Reception


Example: 8-bit, Parity Enabled 1 Stop
Baud Rate Clock
RXD
Sampling
Start
D0
D1
D2
D3
D4
D5
D6
D7
Parity Bit
Stop Bit

29.6.3.7 Receiver Operations

When a character reception is completed, it is transferred to the Receive Holding Register (US_RHR) and the RXRDY bit in the Status Register (US_CSR) rises. If a character is completed while the RXRDY is set, the OVRE (Overrun Error) bit is set. The last character is transferred into US_RHR and overwrites the previous one. The OVRE bit is cleared by writing the Control Register (US_CR) with the RSTSTA (Reset Status) bit at 1.

Figure 29-21. Receiver Status


Baud Rate Clock
RXD
Start Bit
D0
D1
D2
D3
D4
D5
D6
D7
Parity Bit
Stop Bit
Start Bit
D0
D1
D2
D3
D4
D5
D6
D7
Parity Bit
Stop Bit
RSTSTA = 1
Write US_CR
Read US_RHR
RXRDY
OVRE

29.6.3.8 Parity

The USART supports five parity modes selected by programming the PAR field in the Mode Register (US_MR). The PAR field also enables the Multidrop mode, see "Multidrop Mode" on page 363. Even and odd parity bit generation and error detection are supported.

If even parity is selected, the parity generator of the transmitter drives the parity bit at 0 if a number of 1s in the character data bit is even, and at 1 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If odd parity is selected, the parity generator of the transmitter drives the parity bit at 1 if a number of 1s in the character data bit is even, and at 0 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If the mark parity is used, the parity generator of the transmitter drives the parity bit at 1 for all characters. The receiver parity checker reports an error if the parity bit is sampled at 0. If the space parity is used, the parity generator of the transmitter drives the parity bit at 0 for all characters. The receiver parity checker reports an error if the parity bit is sampled at 1. If parity is disabled, the transmitter does not generate any parity bit and the receiver does not report any parity error.

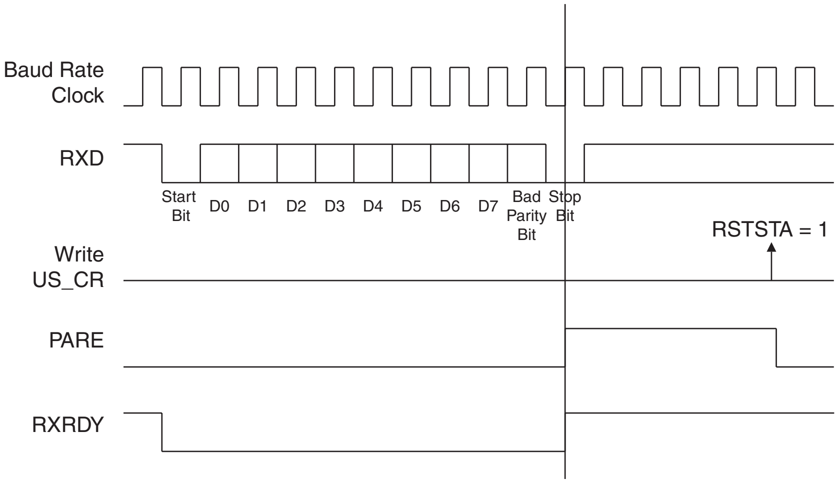
Table 29-6 shows an example of the parity bit for the character 0x41 (character ASCII "A") depending on the configuration of the USART. Because there are two bits at 1, 1 bit is added when a parity is odd, or 0 is added when a parity is even.

Table 29-6. Parity Bit Examples

Character	Hexa	Binary	Parity Bit	Parity Mode
A	0x41	0100 0001	1	Odd
A	0x41	0100 0001	0	Even
A	0x41	0100 0001	1	Mark
A	0x41	0100 0001	0	Space
A	0x41	0100 0001	None	None

When the receiver detects a parity error, it sets the PARE (Parity Error) bit in the Channel Status Register (US_CSR). The PARE bit can be cleared by writing the Control Register (US_CR) with the RSTSTA bit at 1. Figure 29-22 illustrates the parity bit status setting and clearing.

Figure 29-22. Parity Error



29.6.3.9 *Multidrop Mode*

If the PAR field in the Mode Register (US_MR) is programmed to the value 0x6 or 0x07, the USART runs in Multidrop Mode. This mode differentiates the data characters and the address characters. Data is transmitted with the parity bit at 0 and addresses are transmitted with the parity bit at 1.

If the USART is configured in multidrop mode, the receiver sets the PARE parity error bit when the parity bit is high and the transmitter is able to send a character with the parity bit high when the Control Register is written with the SENDA bit at 1.

To handle parity error, the PARE bit is cleared when the Control Register is written with the bit RSTSTA at 1.

The transmitter sends an address byte (parity bit set) when SENDA is written to US_CR. In this case, the next byte written to US_THR is transmitted as an address. Any character written in US_THR without having written the command SENDA is transmitted normally with the parity at 0.

29.6.3.10 *Transmitter Timeguard*

The timeguard feature enables the USART interface with slow remote devices.

The timeguard function enables the transmitter to insert an idle state on the TXD line between two characters. This idle state actually acts as a long stop bit.

The duration of the idle state is programmed in the TG field of the Transmitter Timeguard Register (US_TTGR). When this field is programmed at zero no timeguard is generated. Otherwise, the transmitter holds a high level on TXD after each transmitted byte during the number of bit periods programmed in TG in addition to the number of stop bits.

As illustrated in Figure 29-23, the behavior of TXRDY and TXEMPTY status bits is modified by the programming of a timeguard. TXRDY rises only when the start bit of the next character is sent, and thus remains at 0 during the timeguard transmission if a character has been written in US_THR. TXEMPTY remains low until the timeguard transmission is completed as the timeguard is part of the current character being transmitted.

Figure 29-23. Timeguard Operations

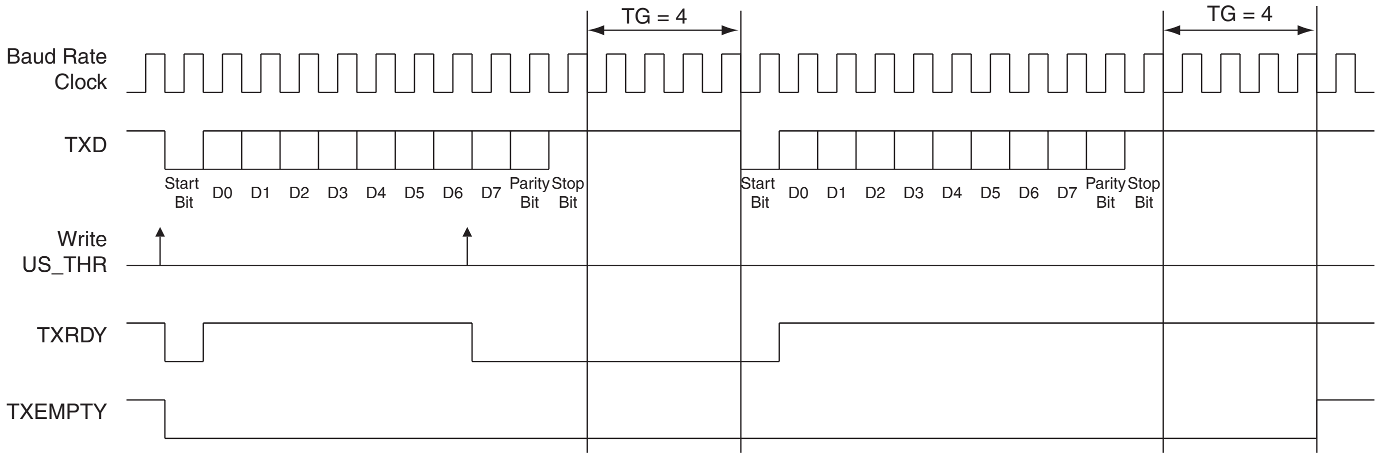


Table 29-7 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the Baud Rate.

Table 29-7. Maximum Timeguard Length Depending on Baud Rate

Baud Rate	Bit time	Timeguard
Bit/sec	µs	ms
1 200	833	212.50
9 600	104	26.56
14400	69.4	17.71
19200	52.1	13.28
28800	34.7	8.85
33400	29.9	7.63
56000	17.9	4.55
57600	17.4	4.43
115200	8.7	2.21

29.6.3.11 Receiver Time-out

The Receiver Time-out provides support in handling variable-length frames. This feature detects an idle condition on the RXD line. When a time-out is detected, the bit TIMEOUT in the Channel Status Register (US_CSR) rises and can generate an interrupt, thus indicating to the driver an end of frame.

The time-out delay period (during which the receiver waits for a new character) is programmed in the TO field of the Receiver Time-out Register (US_RTOR). If the TO field is programmed at 0, the Receiver Time-out is disabled and no time-out is detected. The TIMEOUT bit in US_CSR remains at 0. Otherwise, the receiver loads a 16-bit counter with the value programmed in TO. This counter is decremented at each bit period and reloaded each time a new character is received. If the counter reaches 0, the TIMEOUT bit in the Status Register rises. Then, the user can either:

- Stop the counter clock until a new character is received. This is performed by writing the Control Register (US_CR) with the STTTO (Start Time-out) bit at 1. In this case, the idle state on RXD before a new character is received will not provide a time-out. This prevents having to handle an interrupt before a character is received and allows waiting for the next idle state on RXD after a frame is received.

- Obtain an interrupt while no character is received. This is performed by writing US_CR with the RETTO (Reload and Start Time-out) bit at 1. If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

If STTTO is performed, the counter clock is stopped until a first character is received. The idle state on RXD before the start of the frame does not provide a time-out. This prevents having to obtain a periodic interrupt and enables a wait of the end of frame when the idle state on RXD is detected.

If RETTO is performed, the counter starts counting down immediately from the value TO. This enables generation of a periodic interrupt so that a user time-out can be handled, for example when no key is pressed on a keyboard.

Figure 29-24 shows the block diagram of the Receiver Time-out feature.

Figure 29-24. Receiver Time-out Block Diagram

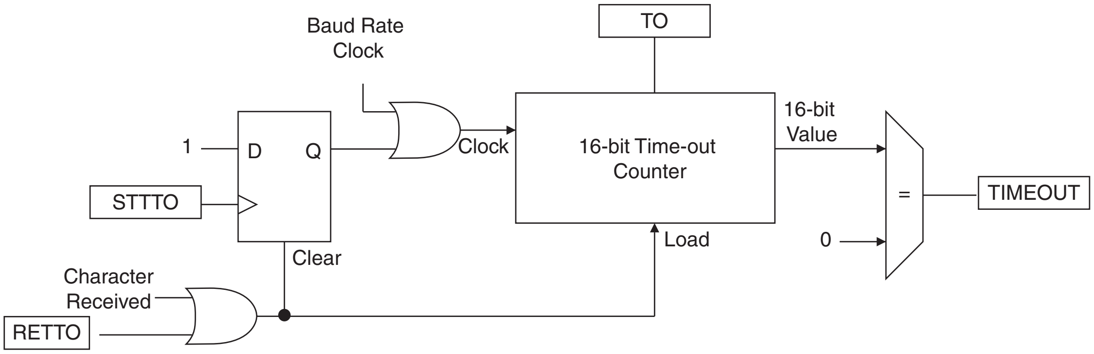


Table 29-8 gives the maximum time-out period for some standard baud rates.

Table 29-8. Maximum Time-out Period

Baud Rate	Bit Time	Time-out
bit/sec	µs	ms
600	1 667	109 225
1 200	833	54 613
2 400	417	27 306
4 800	208	13 653
9 600	104	6 827
14400	69	4 551
19200	52	3 413
28800	35	2 276
33400	30	1 962

Table 29-8. Maximum Time-out Period (Continued)

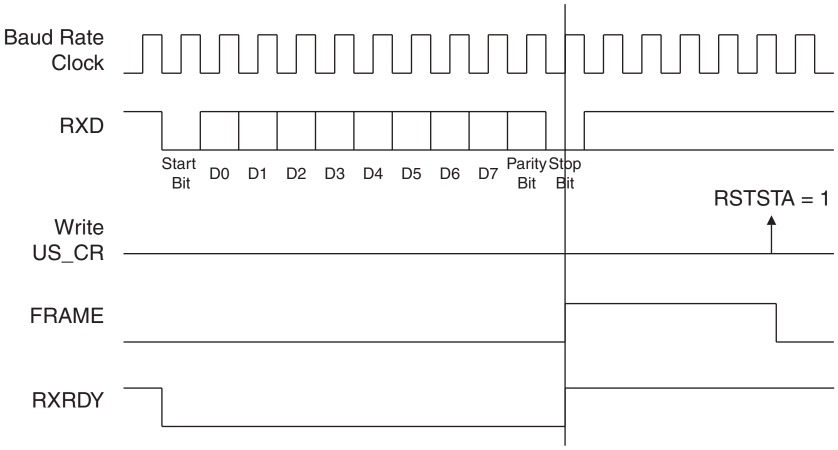
Baud Rate	Bit Time	Time-out
56000	18	1 170
57600	17	1 138
200000	5	328

29.6.3.12 *Framing Error*

The receiver is capable of detecting framing errors. A framing error happens when the stop bit of a received character is detected at level 0. This can occur if the receiver and the transmitter are fully desynchronized.

A framing error is reported on the FRAME bit of the Channel Status Register (US_CSR). The FRAME bit is asserted in the middle of the stop bit as soon as the framing error is detected. It is cleared by writing the Control Register (US_CR) with the RSTSTA bit at 1.

Figure 29-25. Framing Error Status



29.6.3.13 *Transmit Break*

The user can request the transmitter to generate a break condition on the TXD line. A break condition drives the TXD line low during at least one complete character. It appears the same as a 0x00 character sent with the parity and the stop bits at 0. However, the transmitter holds the TXD line at least during one character until the user requests the break condition to be removed.

A break is transmitted by writing the Control Register (US_CR) with the STTBRK bit at 1. This can be performed at any time, either while the transmitter is empty (no character in either the Shift Register or in US_THR) or when a character is being transmitted. If a break is requested while a character is being shifted out, the character is first completed before the TXD line is held low.

Once STTBRK command is requested further STTBRK commands are ignored until the end of the break is completed.

The break condition is removed by writing US_CR with the STPBRK bit at 1. If the STPBRK is requested before the end of the minimum break duration (one character, including start, data, parity and stop bits), the transmitter ensures that the break condition completes.

The transmitter considers the break as though it is a character, i.e. the STTBRK and STPBRK commands are taken into account only if the TXRDY bit in US_CSR is at 1 and the start of the break condition clears the TXRDY and TXEMPTY bits as if a character is processed.

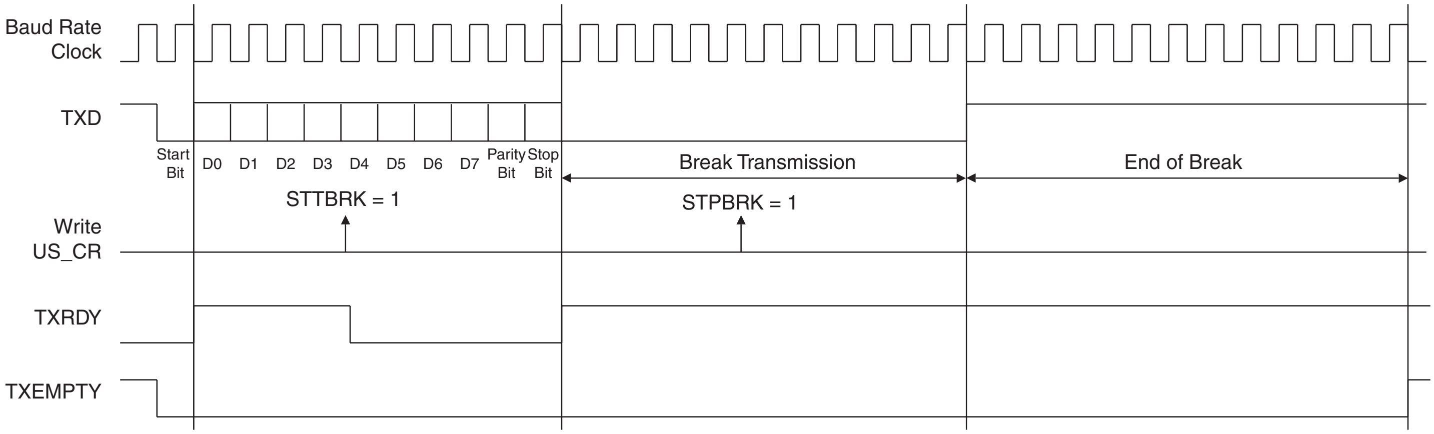
Writing US_CR with the both STTBRK and STPBRK bits at 1 can lead to an unpredictable result. All STPBRK commands requested without a previous STTBRK command are ignored. A byte written into the Transmit Holding Register while a break is pending, but not started, is ignored.

After the break condition, the transmitter returns the TXD line to 1 for a minimum of 12 bit times. Thus, the transmitter ensures that the remote receiver detects correctly the end of break and the start of the next character. If the timeguard is programmed with a value higher than 12, the TXD line is held high for the timeguard period.

After holding the TXD line for this period, the transmitter resumes normal operations.

Figure 29-26 illustrates the effect of both the Start Break (STTBRK) and Stop Break (STPBRK) commands on the TXD line.

Figure 29-26. Break Transmission



29.6.3.14 Receive Break

The receiver detects a break condition when all data, parity and stop bits are low. This corresponds to detecting a framing error with data at 0x00, but FRAME remains low.

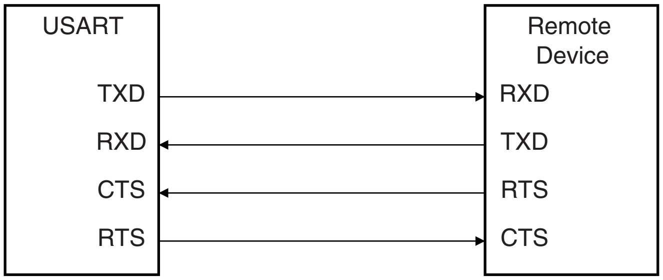
When the low stop bit is detected, the receiver asserts the RXBRK bit in US_CSR. This bit may be cleared by writing the Control Register (US_CR) with the bit RSTSTA at 1.

An end of receive break is detected by a high level for at least 2/16 of a bit period in asynchronous operating mode or one sample at high level in synchronous operating mode. The end of break detection also asserts the RXBRK bit.

29.6.3.15 Hardware Handshaking

The USART features a hardware handshaking out-of-band flow control. The RTS and CTS pins are used to connect with the remote device, as shown in Figure 29-27.

Figure 29-27. Connection with a Remote Device for Hardware Handshaking



Setting the USART to operate with hardware handshaking is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x2.

The USART behavior when hardware handshaking is enabled is the same as the behavior in standard synchronous or asynchronous mode, except that the receiver drives the RTS pin as described below and the level on the CTS pin modifies the behavior of the transmitter as described below. Using this mode requires using the PDC channel for reception. The transmitter can handle hardware handshaking in any case.

Figure 29-28 shows how the receiver operates if hardware handshaking is enabled. The RTS pin is driven high if the receiver is disabled and if the status RXBUFF (Receive Buffer Full) coming from the PDC channel is high. Normally, the remote device does not start transmitting while its CTS pin (driven by RTS) is high. As soon as the Receiver is enabled, the RTS falls, indicating to the remote device that it can start transmitting. Defining a new buffer to the PDC clears the status bit RXBUFF and, as a result, asserts the pin RTS low.

Figure 29-28. Receiver Behavior when Operating with Hardware Handshaking

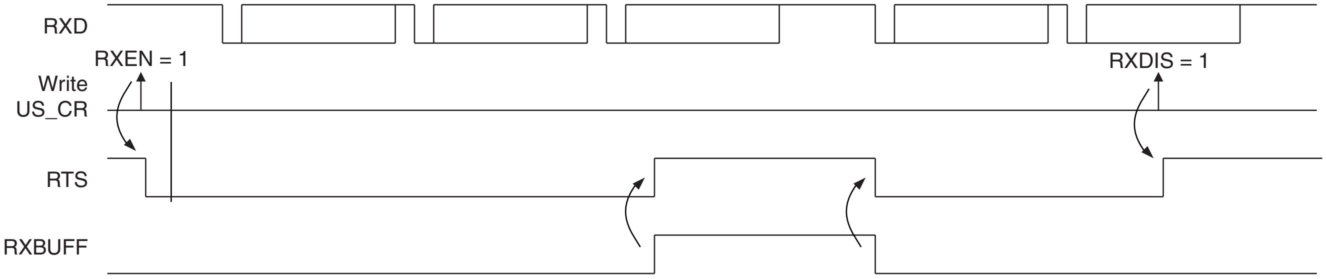
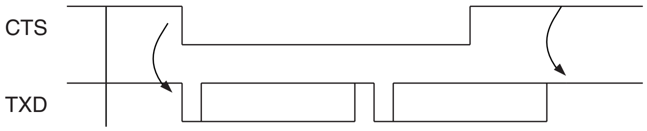


Figure 29-29 shows how the transmitter operates if hardware handshaking is enabled. The CTS pin disables the transmitter. If a character is being processing, the transmitter is disabled only after the completion of the current character and transmission of the next character happens as soon as the pin CTS falls.

Figure 29-29. Transmitter Behavior when Operating with Hardware Handshaking



29.6.4 ISO7816 Mode

The USART features an ISO7816-compatible operating mode. This mode permits interfacing with smart cards and Security Access Modules (SAM) communicating through an ISO7816 link. Both T = 0 and T = 1 protocols defined by the ISO7816 specification are supported.

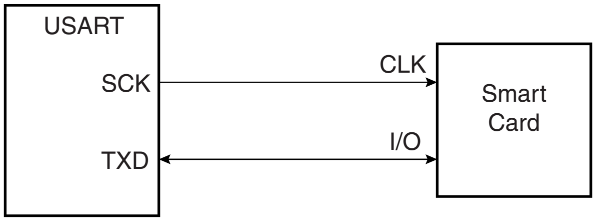
Setting the USART in ISO7816 mode is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x4 for protocol T = 0 and to the value 0x5 for protocol T = 1.

29.6.4.1 ISO7816 Mode Overview

The ISO7816 is a half duplex communication on only one bidirectional line. The baud rate is determined by a division of the clock provided to the remote device (see "Baud Rate Generator" on page 347).

The USART connects to a smart card as shown in Figure 29-30. The TXD line becomes bidirectional and the Baud Rate Generator feeds the ISO7816 clock on the SCK pin. As the TXD pin becomes bidirectional, its output remains driven by the output of the transmitter but only when the transmitter is active while its input is directed to the input of the receiver. The USART is considered as the master of the communication as it generates the clock.

Figure 29-30. Connection of a Smart Card to the USART



When operating in ISO7816, either in T = 0 or T = 1 modes, the character format is fixed. The configuration is 8 data bits, even parity and 1 or 2 stop bits, regardless of the values programmed in the CHRL, MODE9, PAR and CHMODE fields. MSBF can be used to transmit LSB or MSB first. Parity Bit (PAR) can be used to transmit in normal or inverse mode. Refer to "USART Mode Register" on page 381 and "PAR: Parity Type" on page 382.

The USART cannot operate concurrently in both receiver and transmitter modes as the communication is unidirectional at a time. It has to be configured according to the required mode by enabling or disabling either the receiver or the transmitter as desired. Enabling both the receiver and the transmitter at the same time in ISO7816 mode may lead to unpredictable results.

The ISO7816 specification defines an inverse transmission format. Data bits of the character must be transmitted on the I/O line at their negative value. The USART does not support this format and the user has to perform an exclusive OR on the data before writing it in the Transmit Holding Register (US_THR) or after reading it in the Receive Holding Register (US_RHR).

29.6.4.2 Protocol T = 0

In T = 0 protocol, a character is made up of one start bit, eight data bits, one parity bit and one guard time, which lasts two bit times. The transmitter shifts out the bits and does not drive the I/O line during the guard time.

If no parity error is detected, the I/O line remains at 1 during the guard time and the transmitter can continue with the transmission of the next character, as shown in Figure 29-31.

If a parity error is detected by the receiver, it drives the I/O line at 0 during the guard time, as shown in Figure 29-32. This error bit is also named NACK, for Non Acknowledge. In this case, the character lasts 1 bit time more, as the guard time length is the same and is added to the error bit time which lasts 1 bit time.

When the USART is the receiver and it detects an error, it does not load the erroneous character in the Receive Holding Register (US_RHR). It appropriately sets the PARE bit in the Status Register (US_SR) so that the software can handle the error.

Figure 29-31. T = 0 Protocol without Parity Error

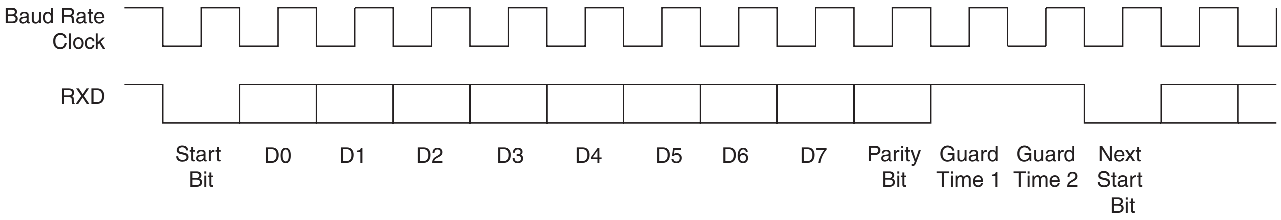
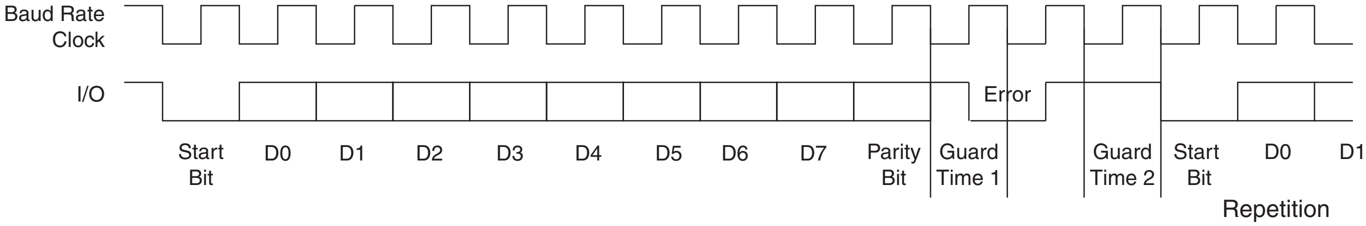

Baud Rate Clock
RXD
Start Bit
D0
D1
D2
D3
D4
D5
D6
D7
Parity Bit
Guard Time 1
Guard Time 2
Next Start Bit

Figure 29-32. T = 0 Protocol with Parity Error


Baud Rate Clock
I/O
Error
Start Bit
D0
D1
D2
D3
D4
D5
D6
D7
Parity Bit
Guard Time 1
Guard Time 2
Start Bit
D0
D1
Repetition

Receive Error Counter

The USART receiver also records the total number of errors. This can be read in the Number of Error (US_NER) register. The NB_ERRORS field can record up to 255 errors. Reading US_NER automatically clears the NB_ERRORS field.

Receive NACK Inhibit

The USART can also be configured to inhibit an error. This can be achieved by setting the INACK bit in the Mode Register (US_MR). If INACK is at 1, no error signal is driven on the I/O line even if a parity bit is detected, but the INACK bit is set in the Status Register (US_SR). The INACK bit can be cleared by writing the Control Register (US_CR) with the RSTNACK bit at 1.

Moreover, if INACK is set, the erroneous received character is stored in the Receive Holding Register, as if no error occurred. However, the RXRDY bit does not raise.

Transmit Character Repetition

When the USART is transmitting a character and gets a NACK, it can automatically repeat the character before moving on to the next one. Repetition is enabled by writing the MAX_ITERATION field in the Mode Register (US_MR) at a value higher than 0. Each character can be transmitted up to eight times; the first transmission plus seven repetitions.

If MAX_ITERATION does not equal zero, the USART repeats the character as many times as the value loaded in MAX_ITERATION.

When the USART repetition number reaches MAX_ITERATION, the ITERATION bit is set in the Channel Status Register (US_CSR). If the repetition of the character is acknowledged by the receiver, the repetitions are stopped and the iteration counter is cleared.

The ITERATION bit in US_CSR can be cleared by writing the Control Register with the RSIT bit at 1.

Disable Successive Receive NACK

The receiver can limit the number of successive NACKs sent back to the remote transmitter. This is programmed by setting the bit DSNACK in the Mode Register (US_MR). The maximum number of NACK transmitted is programmed in the MAX_ITERATION field. As soon as MAX_ITERATION is reached, the character is considered as correct, an acknowledge is sent on the line and the ITERATION bit in the Channel Status Register is set.

29.6.4.3 Protocol T = 1

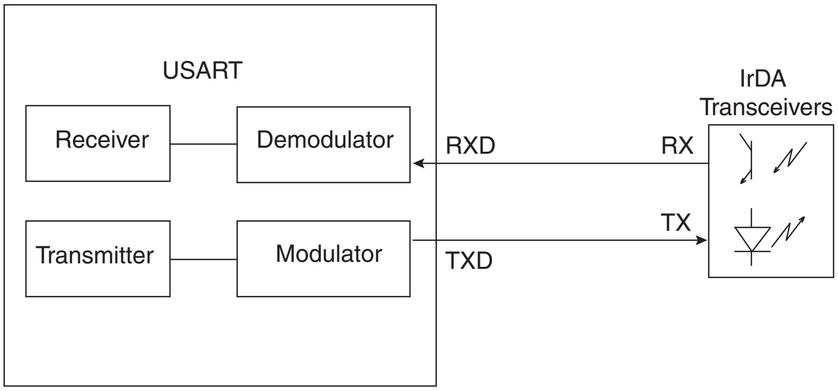
When operating in ISO7816 protocol T = 1, the transmission is similar to an asynchronous format with only one stop bit. The parity is generated when transmitting and checked when receiving. Parity error detection sets the PARE bit in the Channel Status Register (US_CSR).

29.6.5 IrDA Mode

The USART features an IrDA mode supplying half-duplex point-to-point wireless communication. It embeds the modulator and demodulator which allows a glueless connection to the infrared transceivers, as shown in Figure 29-33. The modulator and demodulator are compliant with the IrDA specification version 1.1 and support data transfer speeds ranging from 2.4 Kb/s to 115.2 Kb/s.

The USART IrDA mode is enabled by setting the USART_MODE field in the Mode Register (US_MR) to the value 0x8. The IrDA Filter Register (US_IF) allows configuring the demodulator filter. The USART transmitter and receiver operate in a normal asynchronous mode and all parameters are accessible. Note that the modulator and the demodulator are activated.

Figure 29-33. Connection to IrDA Transceivers



The receiver and the transmitter must be enabled or disabled according to the direction of the transmission to be managed.

29.6.5.1 *IrDA Modulation*

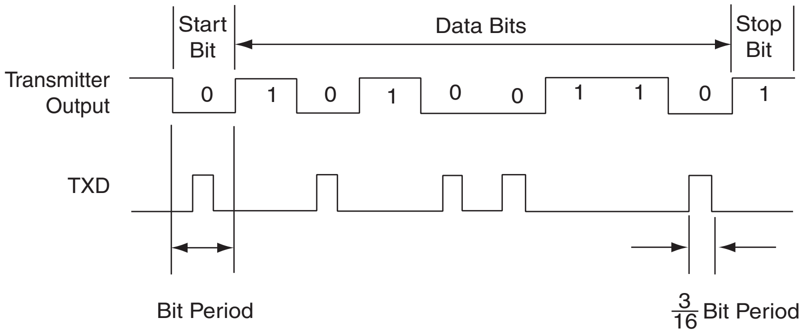
For baud rates up to and including 115.2 Kbits/sec, the RZI modulation scheme is used. “0” is represented by a light pulse of 3/16th of a bit time. Some examples of signal pulse duration are shown in Table 29-9.

Table 29-9. IrDA Pulse Duration

Baud Rate	Pulse Duration (3/16)
2.4 Kb/s	78.13 µs
9.6 Kb/s	19.53 µs
19.2 Kb/s	9.77 µs
38.4 Kb/s	4.88 µs
57.6 Kb/s	3.26 µs
115.2 Kb/s	1.63 µs

Figure 29-34 shows an example of character transmission.

Figure 29-34. IrDA Modulation



29.6.5.2 *IrDA Baud Rate*

Table 29-10 gives some examples of CD values, baud rate error and pulse duration. Note that the requirement on the maximum acceptable error of ±1.87% must be met.

Table 29-10. IrDA Baud Rate Error

Peripheral Clock	Baud Rate	CD	Baud Rate Error	Pulse Time
3 686 400	115 200	2	0.00%	1.63
20 000 000	115 200	11	1.38%	1.63
32 768 000	115 200	18	1.25%	1.63
40 000 000	115 200	22	1.38%	1.63
3 686 400	57 600	4	0.00%	3.26
20 000 000	57 600	22	1.38%	3.26
32 768 000	57 600	36	1.25%	3.26
40 000 000	57 600	43	0.93%	3.26
3 686 400	38 400	6	0.00%	4.88
20 000 000	38 400	33	1.38%	4.88

Table 29-10. IrDA Baud Rate Error (Continued)

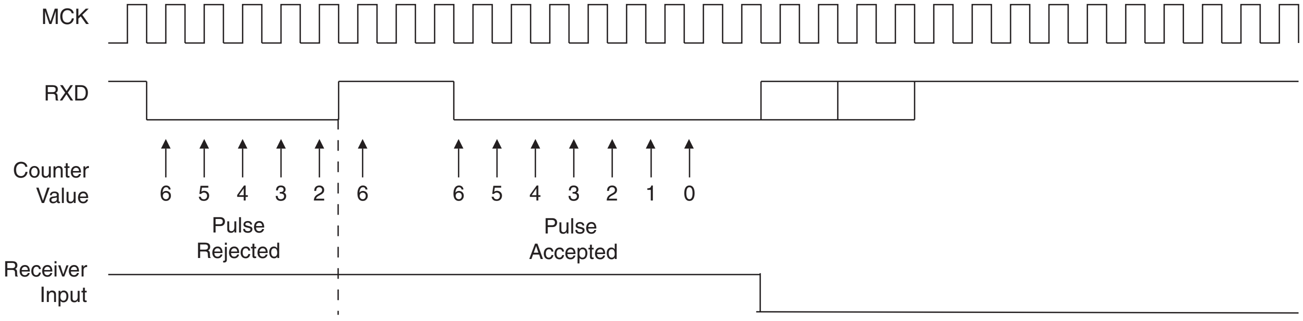
Peripheral Clock	Baud Rate	CD	Baud Rate Error	Pulse Time
32 768 000	38 400	53	0.63%	4.88
40 000 000	38 400	65	0.16%	4.88
3 686 400	19 200	12	0.00%	9.77
20 000 000	19 200	65	0.16%	9.77
32 768 000	19 200	107	0.31%	9.77
40 000 000	19 200	130	0.16%	9.77
3 686 400	9 600	24	0.00%	19.53
20 000 000	9 600	130	0.16%	19.53
32 768 000	9 600	213	0.16%	19.53
40 000 000	9 600	260	0.16%	19.53
3 686 400	2 400	96	0.00%	78.13
20 000 000	2 400	521	0.03%	78.13
32 768 000	2 400	853	0.04%	78.13

29.6.5.3 IrDA Demodulator

The demodulator is based on the IrDA Receive filter comprised of an 8-bit down counter which is loaded with the value programmed in US_IF. When a falling edge is detected on the RXD pin, the Filter Counter starts counting down at the Master Clock (MCK) speed. If a rising edge is detected on the RXD pin, the counter stops and is reloaded with US_IF. If no rising edge is detected when the counter reaches 0, the input of the receiver is driven low during one bit time.

Figure 29-35 illustrates the operations of the IrDA demodulator.

Figure 29-35. IrDA Demodulator Operations

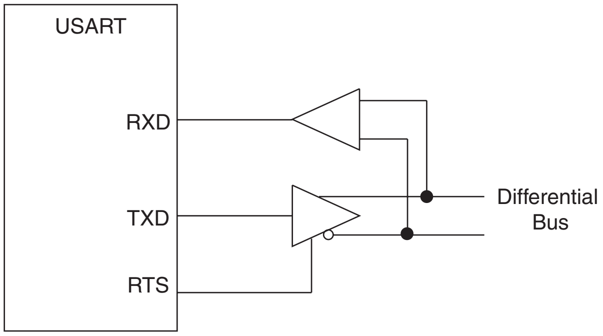


As the IrDA mode uses the same logic as the ISO7816, note that the FI_DI_RATIO field in US_FIDI must be set to a value higher than 0 in order to assure IrDA communications operate correctly.

29.6.6 RS485 Mode

The USART features the RS485 mode to enable line driver control. While operating in RS485 mode, the USART behaves as though in asynchronous or synchronous mode and configuration of all the parameters is possible. The difference is that the RTS pin is driven high when the transmitter is operating. The behavior of the RTS pin is controlled by the TXEMPTY bit. A typical connection of the USART to a RS485 bus is shown in Figure 29-36.

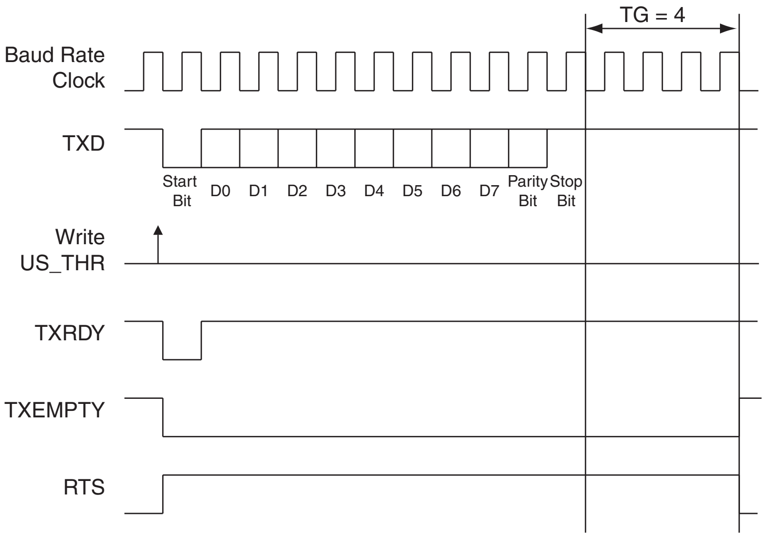
Figure 29-36. Typical Connection to a RS485 Bus


USART
RXD
TXD
RTS
Differential
Bus

The USART is set in RS485 mode by programming the USART_MODE field in the Mode Register (US_MR) to the value 0x1.

The RTS pin is at a level inverse to the TXEMPTY bit. Significantly, the RTS pin remains high when a timeguard is programmed so that the line can remain driven after the last character completion. Figure 29-37 gives an example of the RTS waveform during a character transmission when the timeguard is enabled.

Figure 29-37. Example of RTS Drive with Timeguard


TG = 4
Baud Rate
Clock
TXD
Start
Bit
D0
D1
D2
D3
D4
D5
D6
D7
Parity
Bit
Stop
Bit
Write
US_THR
TXRDY
TXEMPTY
RTS

29.6.7 Modem Mode

The USART features modem mode, which enables control of the signals: DTR (Data Terminal Ready), DSR (Data Set Ready), RTS (Request to Send), CTS (Clear to Send), DCD (Data Carrier Detect) and RI (Ring Indicator). While operating in modem mode, the USART behaves as a DTE (Data Terminal Equipment) as it drives DTR and RTS and can detect level change on DSR, DCD, CTS and RI.

Setting the USART in modem mode is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x3. While operating in modem mode the USART behaves as though in asynchronous mode and all the parameter configurations are available.

Table 29-11 gives the correspondence of the USART signals with modem connection standards.

Table 29-11. Circuit References

USART Pin	V24	CCITT	Direction
TXD	2	103	From terminal to modem
RTS	4	105	From terminal to modem
DTR	20	108.2	From terminal to modem
RXD	3	104	From modem to terminal
CTS	5	106	From terminal to modem
DSR	6	107	From terminal to modem
DCD	8	109	From terminal to modem
RI	22	125	From terminal to modem

The control of the DTR output pin is performed by writing the Control Register (US_CR) with the DTRDIS and DTREN bits respectively at 1. The disable command forces the corresponding pin to its inactive level, i.e. high. The enable command forces the corresponding pin to its active level, i.e. low. RTS output pin is automatically controlled in this mode

The level changes are detected on the RI, DSR, DCD and CTS pins. If an input change is detected, the RIIC, DSRIC, DCDIC and CTSIC bits in the Channel Status Register (US_CSR) are set respectively and can trigger an interrupt. The status is automatically cleared when US_CSR is read. Furthermore, the CTS automatically disables the transmitter when it is detected at its inactive state. If a character is being transmitted when the CTS rises, the character transmission is completed before the transmitter is actually disabled.

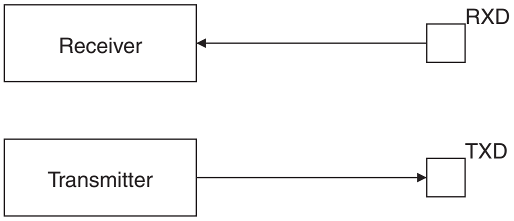
29.6.8 Test Modes

The USART can be programmed to operate in three different test modes. The internal loopback capability allows on-board diagnostics. In the loopback mode the USART interface pins are disconnected or not and reconfigured for loopback internally or externally.

29.6.8.1 Normal Mode

Normal mode connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

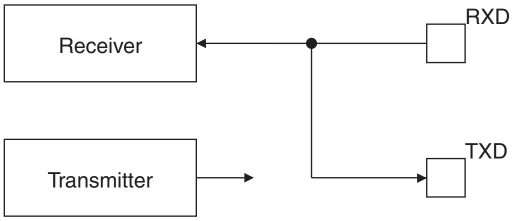
Figure 29-38. Normal Mode Configuration



29.6.8.2 *Automatic Echo Mode*

Automatic echo mode allows bit-by-bit retransmission. When a bit is received on the RXD pin, it is sent to the TXD pin, as shown in Figure 29-39. Programming the transmitter has no effect on the TXD pin. The RXD pin is still connected to the receiver input, thus the receiver remains active.

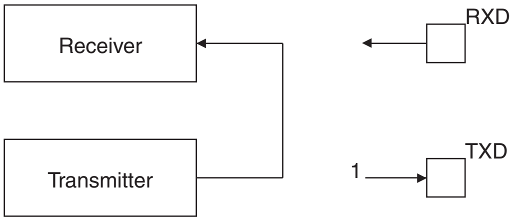
Figure 29-39. Automatic Echo Mode Configuration



29.6.8.3 *Local Loopback Mode*

Local loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in Figure 29-40. The TXD and RXD pins are not used. The RXD pin has no effect on the receiver and the TXD pin is continuously driven high, as in idle state.

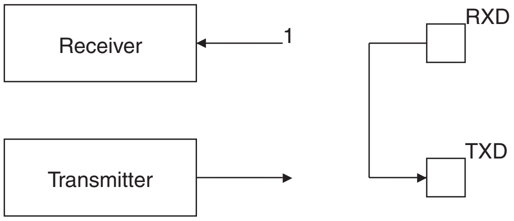
Figure 29-40. Local Loopback Mode Configuration



29.6.8.4 *Remote Loopback Mode*

Remote loopback mode directly connects the RXD pin to the TXD pin, as shown in Figure 29-41. The transmitter and the receiver are disabled and have no effect. This mode allows bit-by-bit retransmission.

Figure 29-41. Remote Loopback Mode Configuration


Receiver
1
RXD
Transmitter
TXD

29.7 USART User Interface

Table 29-12. USART Memory Map

Offset	Register	Name	Access	Reset State
0x0000	Control Register	US_CR	Write-only	–
0x0004	Mode Register	US_MR	Read/Write	–
0x0008	Interrupt Enable Register	US_IER	Write-only	–
0x000C	Interrupt Disable Register	US_IDR	Write-only	–
0x0010	Interrupt Mask Register	US_IMR	Read-only	0x0
0x0014	Channel Status Register	US_CSR	Read-only	–
0x0018	Receiver Holding Register	US_RHR	Read-only	0x0
0x001C	Transmitter Holding Register	US_THR	Write-only	–
0x0020	Baud Rate Generator Register	US_BRGR	Read/Write	0x0
0x0024	Receiver Time-out Register	US_RTOR	Read/Write	0x0
0x0028	Transmitter Timeguard Register	US_TTGR	Read/Write	0x0
0x2C - 0x3C	Reserved	–	–	–
0x0040	FI DI Ratio Register	US_FIDI	Read/Write	0x174
0x0044	Number of Errors Register	US_NER	Read-only	–
0x0048	Reserved	–	–	–
0x004C	IrDA Filter Register	US_IF	Read/Write	0x0
0x0050	Manchester Encoder Decode Register	US_MAN	Read/Write	0x30011004
0x5C - 0xFC	Reserved	–	–	–
0x100 - 0x128	Reserved for PDC Registers	–	–	–

29.7.1 USART Control Register

Name: US_CR

Access Type: Write-only

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

- **RSTRX: Reset Receiver**

0: No effect.

1: Resets the receiver.

- **RSTTX: Reset Transmitter**

0: No effect.

1: Resets the transmitter.

- **RXEN: Receiver Enable**

0: No effect.

1: Enables the receiver, if RXDIS is 0.

- **RXDIS: Receiver Disable**

0: No effect.

1: Disables the receiver.

- **TXEN: Transmitter Enable**

0: No effect.

1: Enables the transmitter if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0: No effect.

1: Disables the transmitter.

- **RSTSTA: Reset Status Bits**

0: No effect.

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

- **STTBRK: Start Break**

0: No effect.

1: Starts transmission of a break after the characters present in US_THR and the Transmit Shift Register have been transmitted. No effect if a break is already being transmitted.

- **STPBRK: Stop Break**

0: No effect.

1: Stops transmission of the break after a minimum of one character length and transmits a high level during 12-bit periods. No effect if no break is being transmitted.

- **STTTO: Start Time-out**

0: No effect.

1: Starts waiting for a character before clocking the time-out counter. Resets the status bit TIMEOUT in US_CSR.

- **SENDA: Send Address**

0: No effect.

1: In Multidrop Mode only, the next character written to the US_THR is sent with the address bit set.

- **RSTIT: Reset Iterations**

0: No effect.

1: Resets ITERATION in US_CSR. No effect if the ISO7816 is not enabled.

- **RSTNACK: Reset Non Acknowledge**

0: No effect

1: Resets NACK in US_CSR.

- **RETTO: Rearm Time-out**

0: No effect

1: Restart Time-out

- **DTREN: Data Terminal Ready Enable**

0: No effect.

1: Drives the pin DTR at 0.

- **DTRDIS: Data Terminal Ready Disable**

0: No effect.

1: Drives the pin DTR to 1.

- **RTSEN: Request to Send Enable**

0: No effect.

1: Drives the pin RTS to 0.

- **RTSDIS: Request to Send Disable**

0: No effect.

1: Drives the pin RTS to 1.

29.7.2 USART Mode Register

Name: US_MR

Access Type: Read/Write

31	30	29	28	27	26	25	24
ONEBIT	MODSYNC–	MAN	FILTER	–	MAX_ITERATION		

23	22	21	20	19	18	17	16
–	VAR_SYNC	DSNACK	INACK	OVER	CLKO	MODE9	MSBF

15	14	13	12	11	10	9	8
CHMODE		NBSTOP		PAR			SYNC

7	6	5	4	3	2	1	0
CHRL		USCLKS		USART_MODE			

- **USART_MODE**

Table 29-13.

USART_MODE				Mode of the USART
0	0	0	0	Normal
0	0	0	1	RS485
0	0	1	0	Hardware Handshaking
0	0	1	1	Modem
0	1	0	0	IS07816 Protocol: T = 0
0	1	0	1	Reserved
0	1	1	0	IS07816 Protocol: T = 1
0	1	1	1	Reserved
1	0	0	0	IrDA
1	1	x	x	Reserved

- **USCLKS: Clock Selection**

Table 29-14.

USCLKS		Selected Clock
0	0	MCK
0	1	MCK/DIV (DIV = 8)
1	0	Reserved
1	1	SCK

- **CHRL: Character Length.**

Table 29-15.

CHRL		Character Length
0	0	5 bits
0	1	6 bits
1	0	7 bits
1	1	8 bits

- **SYNC: Synchronous Mode Select**

0: USART operates in Asynchronous Mode.

1: USART operates in Synchronous Mode.

- **PAR: Parity Type**

Table 29-16.

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	Multidrop mode

- **NBSTOP: Number of Stop Bits**

Table 29-17.

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**

Table 29-18.

CHMODE		Mode Description
0	0	Normal Mode
0	1	Automatic Echo. Receiver input is connected to the TXD pin.
1	0	Local Loopback. Transmitter output is connected to the Receiver Input..
1	1	Remote Loopback. RXD pin is internally connected to the TXD pin.

- **MSBF: Bit Order**

0: Least Significant Bit is sent/received first.

1: Most Significant Bit is sent/received first.

- **MODE9: 9-bit Character Length**

0: CHRL defines character length.

1: 9-bit character length.

- **CLKO: Clock Output Select**

0: The USART does not drive the SCK pin.

1: The USART drives the SCK pin if USCLKS does not select the external clock SCK.

- **OVER: Oversampling Mode**

0: 16x Oversampling.

1: 8x Oversampling.

- **INACK: Inhibit Non Acknowledge**

0: The NACK is generated.

1: The NACK is not generated.

- **DSNACK: Disable Successive NACK**

0: NACK is sent on the ISO line as soon as a parity error occurs in the received character (unless INACK is set).

1: Successive parity errors are counted up to the value specified in the MAX_ITERATION field. These parity errors generate a NACK on the ISO line. As soon as this value is reached, no additional NACK is sent on the ISO line. The flag ITERATION is asserted.

- **VAR_SYNC: Variable Synchronization of Command/Data Sync Start Frame Delimiter**

0: User defined configuration of command or data sync field depending on SYNC value.

1: The sync field is updated when a character is written into US_THR register.

- **MAX_ITERATION**

Defines the maximum number of iterations in mode ISO7816, protocol T= 0.

- **FILTER: Infrared Receive Line Filter**

0: The USART does not filter the receive line.

1: The USART filters the receive line using a three-sample filter (1/16-bit clock) (2 over 3 majority).

- **MAN: Manchester Encoder/Decoder Enable**

0: Manchester Encoder/Decoder are disabled.

1: Manchester Encoder/Decoder are enabled.

- **MODSYNC: Manchester Synchronization Mode**

0:The Manchester Start bit is a 0 to 1 transition

1: The Manchester Start bit is a 1 to 0 transition.

- **ONEBIT: Start Frame Delimiter Selector**

0: Start Frame delimiter is COMMAND or DATA SYNC.

1: Start Frame delimiter is One Bit.

29.7.3 USART Interrupt Enable Register

Name: US_IER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	MANE	CTSIC	DCDIC	DSRIC	RIIC
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: RXRDY Interrupt Enable**

- **TXRDY: TXRDY Interrupt Enable**

- **RXBRK: Receiver Break Interrupt Enable**

- **ENDRX: End of Receive Transfer Interrupt Enable**

- **ENDTX: End of Transmit Interrupt Enable**

- **OVRE: Overrun Error Interrupt Enable**

- **FRAME: Framing Error Interrupt Enable**

- **PARE: Parity Error Interrupt Enable**

- **TIMEOUT: Time-out Interrupt Enable**

- **TXEMPTY: TXEMPTY Interrupt Enable**

- **ITERATION: Iteration Interrupt Enable**

- **TXBUFE: Buffer Empty Interrupt Enable**

- **RXBUFF: Buffer Full Interrupt Enable**

- **NACK: Non Acknowledge Interrupt Enable**

- **RIIC: Ring Indicator Input Change Enable**

- **DSRIC: Data Set Ready Input Change Enable**

- **DCDIC: Data Carrier Detect Input Change Interrupt Enable**

- **CTSIC: Clear to Send Input Change Interrupt Enable**

- **MANE: Manchester Error Interrupt Enable**

0: No effect.

1: Enables the corresponding interrupt.

29.7.4 USART Interrupt Disable Register

Name: US_IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	MANE	CTSIC	DCDIC	DSRIC	RIIC
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: RXRDY Interrupt Disable**

- **TXRDY: TXRDY Interrupt Disable**

- **RXBRK: Receiver Break Interrupt Disable**

- **ENDRX: End of Receive Transfer Interrupt Disable**

- **ENDTX: End of Transmit Interrupt Disable**

- **OVRE: Overrun Error Interrupt Disable**

- **FRAME: Framing Error Interrupt Disable**

- **PARE: Parity Error Interrupt Disable**

- **TIMEOUT: Time-out Interrupt Disable**

- **TXEMPTY: TXEMPTY Interrupt Disable**

- **ITERATION: Iteration Interrupt Disable**

- **TXBUFE: Buffer Empty Interrupt Disable**

- **RXBUFF: Buffer Full Interrupt Disable**

- **NACK: Non Acknowledge Interrupt Disable**

- **RIIC: Ring Indicator Input Change Disable**

- **DSRIC: Data Set Ready Input Change Disable**

- **DCDIC: Data Carrier Detect Input Change Interrupt Disable**

- **CTSIC: Clear to Send Input Change Interrupt Disable**

- **MANE: Manchester Error Interrupt Disable**

0: No effect.

1: Disables the corresponding interrupt.

29.7.5 USART Interrupt Mask Register

Name: US_IMR

Access Type: Read-only

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

23	22	21	20	19	18	17	16
–	–	–	MANE	CTSIC	DCDIC	DSRIC	RIIC

15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: RXRDY Interrupt Mask**

- **TXRDY: TXRDY Interrupt Mask**

- **RXBRK: Receiver Break Interrupt Mask**

- **ENDRX: End of Receive Transfer Interrupt Mask**

- **ENDTX: End of Transmit Interrupt Mask**

- **OVRE: Overrun Error Interrupt Mask**

- **FRAME: Framing Error Interrupt Mask**

- **PARE: Parity Error Interrupt Mask**

- **TIMEOUT: Time-out Interrupt Mask**

- **TXEMPTY: TXEMPTY Interrupt Mask**

- **ITERATION: Iteration Interrupt Mask**

- **TXBUFE: Buffer Empty Interrupt Mask**

- **RXBUFF: Buffer Full Interrupt Mask**

- **NACK: Non Acknowledge Interrupt Mask**

- **RIIC: Ring Indicator Input Change Mask**

- **DSRIC: Data Set Ready Input Change Mask**

- **DCDIC: Data Carrier Detect Input Change Interrupt Mask**

- **CTSIC: Clear to Send Input Change Interrupt Mask**

- **MANE: Manchester Error Interrupt Mask**

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

29.7.6 USART Channel Status Register

Name: US_CSR

Access Type: Read-only

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

23	22	21	20	19	18	17	16
CTS	DCD	DSR	RI	CTSIC	DCDIC	DSRIC	RIIC

15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: Receiver Ready**

0: No complete character has been received since the last read of US_RHR or the receiver is disabled. If characters were being received when the receiver was disabled, RXRDY changes to 1 when the receiver is enabled.

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

- **TXRDY: Transmitter Ready**

0: A character is in the US_THR waiting to be transferred to the Transmit Shift Register, or an STTBRK command has been requested, or the transmitter is disabled. As soon as the transmitter is enabled, TXRDY becomes 1.

1: There is no character in the US_THR.

- **RXBRK: Break Received/End of Break**

0: No Break received or End of Break detected since the last RSTSTA.

1: Break Received or End of Break detected since the last RSTSTA.

- **ENDRX: End of Receiver Transfer**

0: The End of Transfer signal from the Receive PDC channel is inactive.

1: The End of Transfer signal from the Receive PDC channel is active.

- **ENDTX: End of Transmitter Transfer**

0: The End of Transfer signal from the Transmit PDC channel is inactive.

1: The End of Transfer signal from the Transmit PDC channel is active.

- **OVRE: Overrun Error**

0: No overrun error has occurred since the last RSTSTA.

1: At least one overrun error has occurred since the last RSTSTA.

- **FRAME: Framing Error**

0: No stop bit has been detected low since the last RSTSTA.

1: At least one stop bit has been detected low since the last RSTSTA.

- **PARE: Parity Error**

0: No parity error has been detected since the last RSTSTA.

1: At least one parity error has been detected since the last RSTSTA.

- **TIMEOUT: Receiver Time-out**

0: There has not been a time-out since the last Start Time-out command (STTTO in US_CR) or the Time-out Register is 0.

1: There has been a time-out since the last Start Time-out command (STTTO in US_CR).

- **TXEMPTY: Transmitter Empty**

0: There are characters in either US_THR or the Transmit Shift Register, or the transmitter is disabled.

1: There are no characters in US_THR, nor in the Transmit Shift Register.

- **ITERATION: Max number of Repetitions Reached**

0: Maximum number of repetitions has not been reached since the last RSIT.

1: Maximum number of repetitions has been reached since the last RSIT.

- **TXBUFE: Transmission Buffer Empty**

0: The signal Buffer Empty from the Transmit PDC channel is inactive.

1: The signal Buffer Empty from the Transmit PDC channel is active.

- **RXBUFF: Reception Buffer Full**

0: The signal Buffer Full from the Receive PDC channel is inactive.

1: The signal Buffer Full from the Receive PDC channel is active.

- **NACK: Non Acknowledge**

0: No Non Acknowledge has not been detected since the last RSTNACK.

1: At least one Non Acknowledge has been detected since the last RSTNACK.

- **RIIC: Ring Indicator Input Change Flag**

0: No input change has been detected on the RI pin since the last read of US_CSR.

1: At least one input change has been detected on the RI pin since the last read of US_CSR.

- **DSRIC: Data Set Ready Input Change Flag**

0: No input change has been detected on the DSR pin since the last read of US_CSR.

1: At least one input change has been detected on the DSR pin since the last read of US_CSR.

- **DCDIC: Data Carrier Detect Input Change Flag**

0: No input change has been detected on the DCD pin since the last read of US_CSR.

1: At least one input change has been detected on the DCD pin since the last read of US_CSR.

- **CTSIC: Clear to Send Input Change Flag**

0: No input change has been detected on the CTS pin since the last read of US_CSR.

1: At least one input change has been detected on the CTS pin since the last read of US_CSR.

- **RI: Image of RI Input**

0: RI is at 0.

1: RI is at 1.

- **DSR: Image of DSR Input**

0: DSR is at 0

1: DSR is at 1.

- **DCD: Image of DCD Input**

0: DCD is at 0.

1: DCD is at 1.

- **CTS: Image of CTS Input**

0: CTS is at 0.

1: CTS is at 1.

- **MANERR: Manchester Error**

0: No Manchester error has been detected since the last RSTSTA.

1: At least one Manchester error has been detected since the last RSTSTA.

29.7.7 USART Receive Holding Register

Name: US_RHR

Access Type: Read-only

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

- **RXCHR: Received Character**

Last character received if RXRDY is set.

- **RXSYNH: Received Sync**

0: Last Character received is a Data.

1: Last Character received is a Command.

29.7.8 USART Transmit Holding Register

Name: US_THR

Access Type: Write-only

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

- **TXSYNH: Sync Field to be transmitted**

0: The next character sent is encoded as a data. Start Frame Delimiter is DATA SYNC.

1: The next character sent is encoded as a command. Start Frame Delimiter is COMMAND SYNC.

29.7.9 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
–	–	–	–	–	FP		

15	14	13	12	11	10	9	8
CD							

7	6	5	4	3	2	1	0
CD							

- **CD: Clock Divider**

Table 29-19.

CD	**USART_MODE ≠ ISO7816**			USART_MODE = ISO7816
	SYNC = 0		SYNC = 1	
	OVER = 0	**OVER = 1**		
0	Baud Rate Clock Disabled			
1 to 65535	Baud Rate = Selected Clock/16/CD	Baud Rate = Selected Clock/8/CD	Baud Rate = Selected Clock /CD	Baud Rate = Selected Clock/CD/FI_DI_RATIO

- **FP: Fractional Part**

0: Fractional divider is disabled.

1 - 7: Baudrate resolution, defined by FP x 1/8.

29.7.10 USART Receiver Time-out Register

Name: US_RTOR

Access Type: Read/Write

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

- **TO: Time-out Value**

0: The Receiver Time-out is disabled.

1 - 65535: The Receiver Time-out is enabled and the Time-out delay is TO x Bit Period.

29.7.11 USART Transmitter Timeguard Register

Name: US_TTGR

Access Type: Read/Write

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

- **TG: Timeguard Value**

0: The Transmitter Timeguard is disabled.

1 - 255: The Transmitter timeguard is enabled and the timeguard delay is TG x Bit Period.

29.7.12 USART FI DI RATIO Register

Name: US_FIDI

Access Type: Read/Write

Reset Value : 0x174

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	FI_DI_RATIO		

7	6	5	4	3	2	1	0
FI_DI_RATIO							

- **FI_DI_RATIO: FI Over DI Ratio Value**

0: If ISO7816 mode is selected, the Baud Rate Generator generates no signal.

1 - 2047: If ISO7816 mode is selected, the Baud Rate is the clock provided on SCK divided by FI_DI_RATIO.

29.7.13 USART Number of Errors Register

Name: US_NER

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
NB_ERRORS							

- **NB_ERRORS: Number of Errors**

Total number of errors that occurred during an ISO7816 transfer. This register automatically clears when read.

29.7.14 USART Manchester Configuration Register

Name: US_MAN

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	DRIFT	–	RX_MPOL	–	–	RX_PP	

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

15	14	13	12	11	10	9	8
–	–	–	TX_MPOL	–	–	TX_PP	

7	6	5	4	3	2	1	0
–	–	–	–	TX_PL			

- **TX_PL: Transmitter Preamble Length**

0: The Transmitter Preamble pattern generation is disabled

1 - 15: The Preamble Length is TX_PL x Bit Period

- **TX_PP: Transmitter Preamble Pattern**

Table 29-20.

TX_PP		Preamble Pattern default polarity assumed (TX_MPOL field not set)
0	0	ALL_ONE
0	1	ALL_ZERO
1	0	ZERO_ONE
1	1	ONE_ZERO

- **TX_MPOL: Transmitter Manchester Polarity**

0: Logic Zero is coded as a zero-to-one transition, Logic One is coded as a one-to-zero transition.

1: Logic Zero is coded as a one-to-zero transition, Logic One is coded as a zero-to-one transition.

- **RX_PL: Receiver Preamble Length**

0: The receiver preamble pattern detection is disabled

1 - 15: The detected preamble length is RX_PL x Bit Period

- **RX_PP: Receiver Preamble Pattern detected**

Table 29-21.

RX_PP		Preamble Pattern default polarity assumed (RX_MPOL field not set)
0	0	ALL_ONE
0	1	ALL_ZERO
1	0	ZERO_ONE
1	1	ONE_ZERO

- **RX_MPOL: Receiver Manchester Polarity**

0: Logic Zero is coded as a zero-to-one transition, Logic One is coded as a one-to-zero transition.

1: Logic Zero is coded as a one-to-zero transition, Logic One is coded as a zero-to-one transition.

- **DRIFT: Drift compensation**

0: The USART can not recover from an important clock drift

1: The USART can recover from clock drift. The 16X clock mode must be enabled.

29.7.15 USART IrDA FILTER Register

Name: US_IF

Access Type: Read/Write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
IRDA_FILTER							

- **IRDA_FILTER: IrDA Filter**

Sets the filter of the IrDA demodulator.

30. Timer/Counter (TC)

30.1 Description

The Timer Counter (TC) includes three identical 16-bit Timer Counter channels.

Each channel can be independently programmed to perform a wide range of functions including frequency measurement, event counting, interval measurement, pulse generation, delay timing and pulse width modulation.

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

The Timer Counter block has two global registers which act upon all three TC channels.

The Block Control Register allows the three channels to be started simultaneously with the same instruction.

The Block Mode Register defines the external clock inputs for each channel, allowing them to be chained.

Table 30-1 gives the assignment of the device Timer Counter clock inputs common to Timer Counter 0 to 2

Table 30-1. Timer Counter Clock Assignment

Name	Definition
TIMER_CLOCK1	MCK/2
TIMER_CLOCK2	MCK/8
TIMER_CLOCK3	MCK/32
TIMER_CLOCK4	MCK/128
TIMER_CLOCK5	SCLK

30.2 Block Diagram

Figure 30-1. Timer Counter Block Diagram

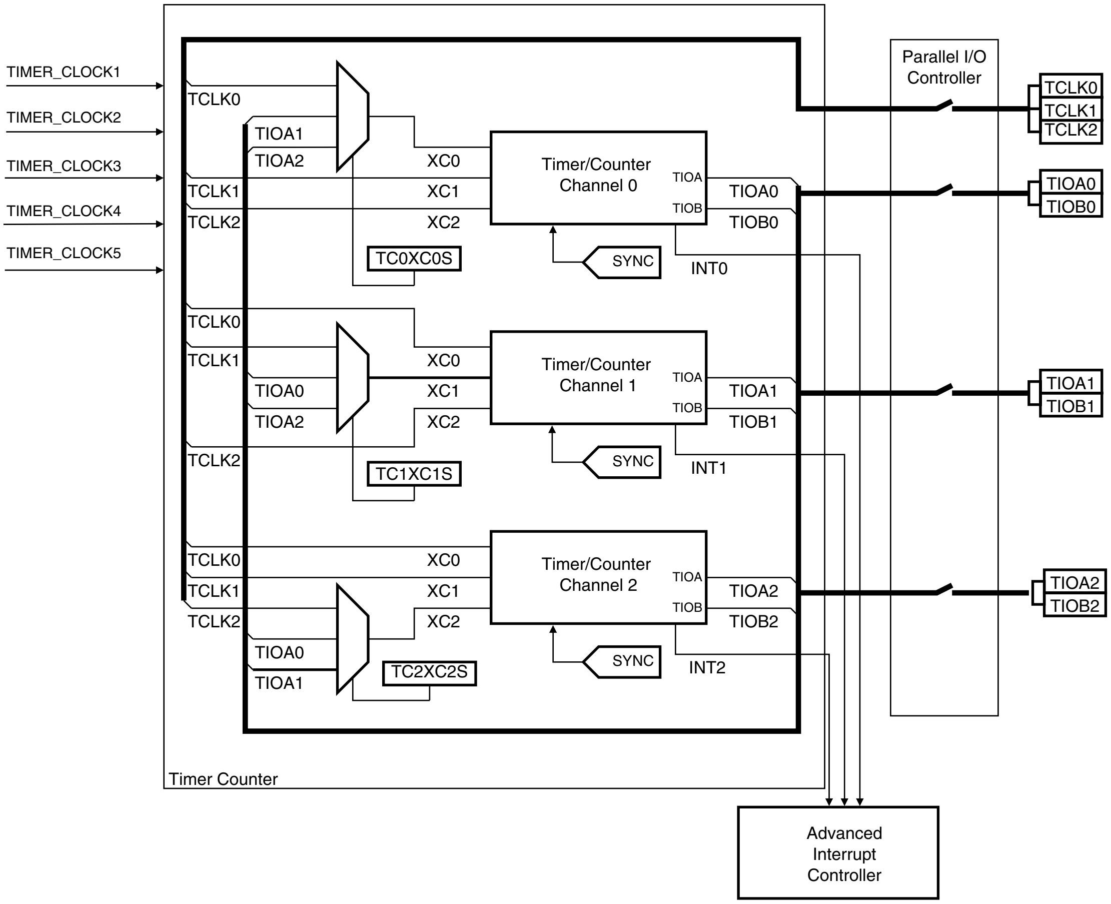

TIMER_CLOCK1
TIMER_CLOCK2
TIMER_CLOCK3
TIMER_CLOCK4
TIMER_CLOCK5
TCLK0
TCLK1
TCLK2
TIOA1
TIOA2
XC0
XC1
XC2
TC0XC0S
Timer/Counter Channel 0
TIOA
TIOB
TIOA0
TIOB0
SYNC
INT0
TIOA0
TIOA2
TC1XC1S
Timer/Counter Channel 1
TIOA1
TIOB1
INT1
TIOA1
TC2XC2S
Timer/Counter Channel 2
TIOA2
TIOB2
INT2
Timer Counter
Parallel I/O Controller
Advanced Interrupt Controller

Table 30-2. Signal Name Description

Block/Channel	Signal Name	Description
Channel Signal	XC0, XC1, XC2	External Clock Inputs
	TIOA	Capture Mode: Timer Counter Input Waveform Mode: Timer Counter Output
	TIOB	Capture Mode: Timer Counter Input Waveform Mode: Timer Counter Input/Output
	INT	Interrupt Signal Output
	SYNC	Synchronization Input Signal

30.3 Pin Name List

Table 30-3. TC pin list

Pin Name	Description	Type
TCLK0-TCLK2	External Clock Input	Input
TIOA0-TIOA2	I/O Line A	I/O
TIOB0-TIOB2	I/O Line B	I/O

30.4 Product Dependencies

30.4.1 I/O Lines

The pins used for interfacing the compliant external devices are multiplexed with PIO lines. The programmer must first program the PIOA controller to select the appropriate TC alternate functions.

30.4.2 Power Management

The TC is clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the Timer Counter clock.

30.4.3 Interrupt

The TC has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling the TC interrupt requires programming the AIC before configuring the TC.

30.5 Functional Description

30.5.1 TC Description

The three channels of the Timer Counter are independent and identical in operation. The registers for channel programming are listed in Table 30-5 on page 413.

30.5.2 16-bit Counter

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

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

30.5.3 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 internal I/O signals TIOA0, TIOA1 or TIOA2 for chaining by programming the TC_BMR (Block Mode). See Figure 30-2 on page 401.

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

- Internal clock signals: TIMER_CLOCK1, TIMER_CLOCK2, TIMER_CLOCK3, TIMER_CLOCK4, TIMER_CLOCK5
- External clock signals: XC0, XC1 or XC2

This selection is made by the TCCLKS bits in the TC Channel Mode Register.

The selected clock can be inverted with the CLKI bit in TC_CMR. 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). See Figure 30-3 on page 401

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

Figure 30-2. Clock Chaining Selection

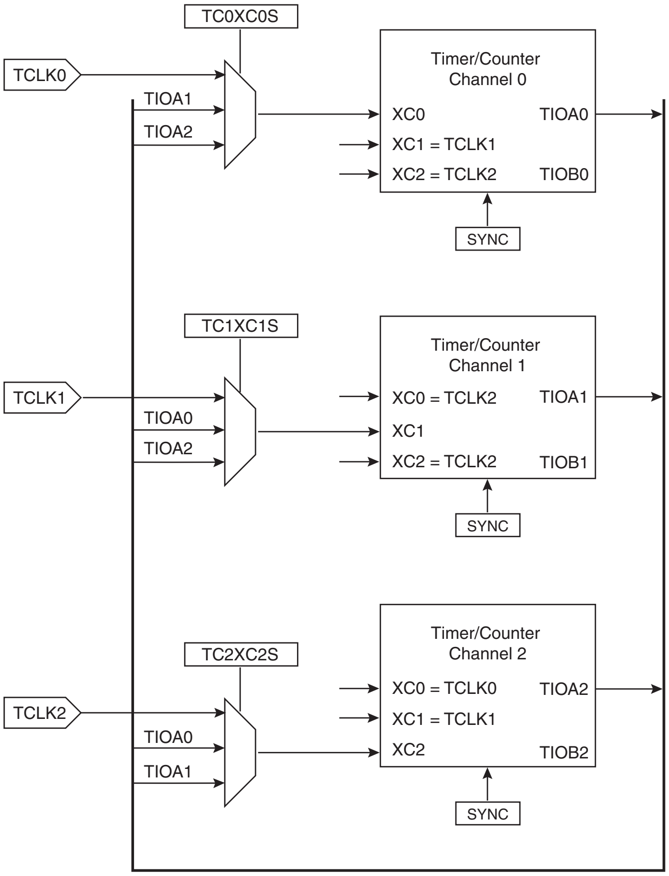
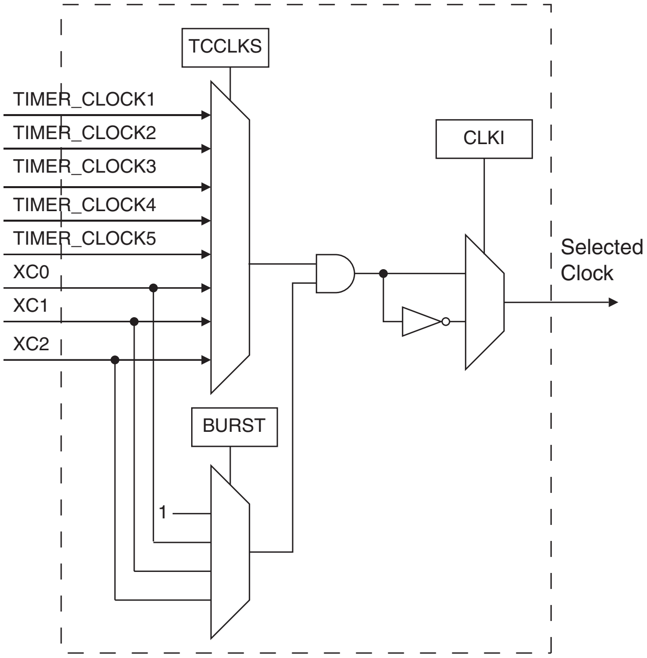

TC0XC0S
TCLK0
TIOA1
TIOA2
Timer/Counter Channel 0
XC0
XC1 = TCLK1
XC2 = TCLK2
TIOA0
TIOB0
SYNC
TC1XC1S
TCLK1
TIOA0
TIOA2
Timer/Counter Channel 1
XC0 = TCLK2
XC1
XC2 = TCLK2
TIOA1
TIOB1
SYNC
TC2XC2S
TCLK2
TIOA0
TIOA1
Timer/Counter Channel 2
XC0 = TCLK0
XC1 = TCLK1
XC2
TIOA2
TIOB2
SYNC

Figure 30-3. Clock Selection

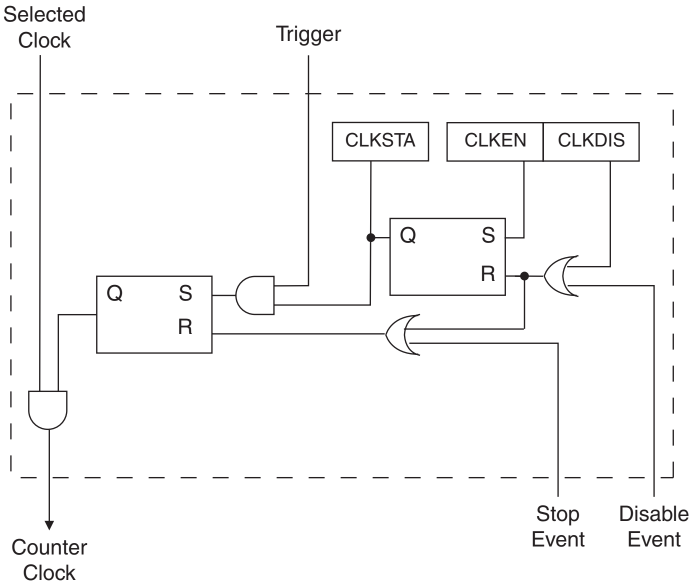

TCCLKS
TIMER_CLOCK1
TIMER_CLOCK2
TIMER_CLOCK3
TIMER_CLOCK4
TIMER_CLOCK5
XC0
XC1
XC2
CLKI
Selected Clock
BURST
1

30.5.4 Clock Control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See Figure 30-4.

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

Figure 30-4. Clock Control



30.5.5 TC Operating Modes

Each channel can independently operate in two different modes:

- Capture Mode provides measurement on signals.
- Waveform Mode provides wave generation.

The TC Operating Mode is programmed with the WAVE bit in the TC Channel Mode Register.

In Capture Mode, TIOA and TIOB are configured as inputs.

In Waveform Mode, TIOA is always configured to be an output and TIOB is an output if it is not selected to be the external trigger.

30.5.6 Trigger

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

The following triggers are common to both modes:

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

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

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

Regardless of the trigger used, it will be taken into account at the following active edge of the selected clock. This means that the counter value can be read differently from zero just after a trigger, especially when a low frequency signal is selected as the clock.

30.5.7 Capture Operating Mode

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

Capture Mode allows the TC channel to perform measurements such as pulse timing, frequency, period, duty cycle and phase on TIOA and TIOB signals which are considered as inputs.

Figure 30-5 shows the configuration of the TC channel when programmed in Capture Mode.

30.5.8 Capture Registers A and B

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

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

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

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

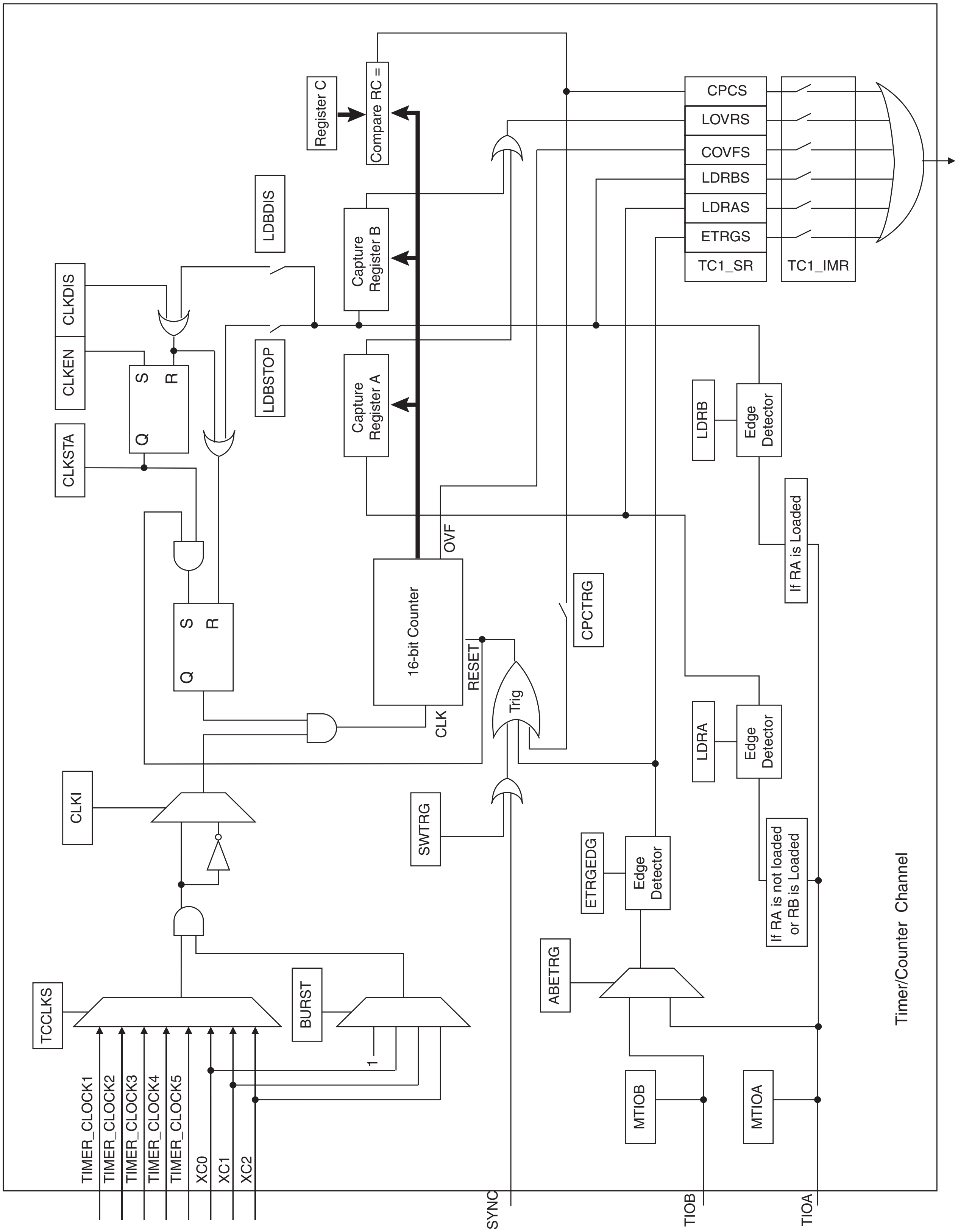
Loading RA or RB before the read of the last value loaded sets the Overrun Error Flag (LOVRS) in TC_SR (Status Register). In this case, the old value is overwritten.

30.5.9 Trigger Conditions

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

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

Figure 30-5. Capture Mode


TIMER_CLOCK1
TIMER_CLOCK2
TIMER_CLOCK3
TIMER_CLOCK4
TIMER_CLOCK5
XC0
XC1
XC2
TCCLKS
BURST
1
CLKI
CLKSTA
CLKEN
CLKDIS
S
R
Q
S
R
Q
LDBSTOP
LDBDIS
Capture Register A
Capture Register B
Register C
Compare RC =
16-bit Counter
CLK
RESET
OVF
Trig
SWTRG
SYNC
CPCTRG
ETRGEDG
Edge Detector
ABETRG
MTIOB
MTIOA
TIOB
TIOA
LDRA
Edge Detector
LDRB
Edge Detector
If RA is Loaded
If RA is not loaded or RB is Loaded
CPCS
LOVRS
COVFS
LDRBS
LDRAS
ETRGS
TC1_SR
TC1_IMR
INT
Timer/Counter Channel

30.5.10 Waveform Operating Mode

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

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

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

Figure 30-6 shows the configuration of the TC channel when programmed in Waveform Operating Mode.

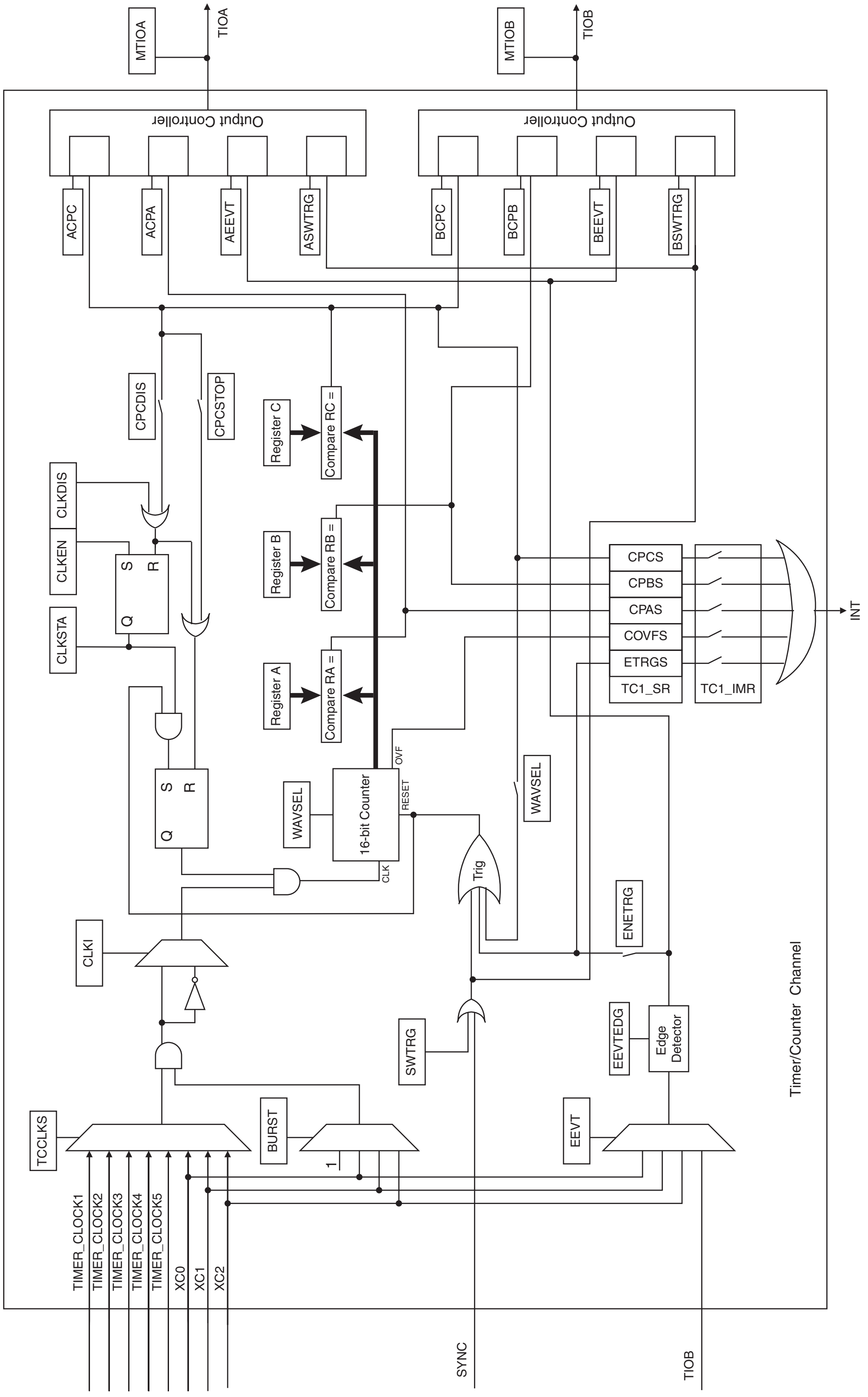
30.5.11 Waveform Selection

Depending on the WAVSEL parameter in TC_CMR (Channel Mode Register), the behavior of TC_CV varies.

With any selection, RA, RB and RC can all be used as compare registers.

RA Compare is used to control the TIOA output, RB Compare is used to control the TIOB output (if correctly configured) and RC Compare is used to control TIOA and/or TIOB outputs.

Figure 30-6. Waveform Mode


MTIOA
TIOA
MTIOB
TIOB
Output Controller
Output Controller
ACPC
ACPA
AEEVT
ASWTRG
BCPC
BCPB
BEEVT
BSWTRG
CPCDIS
CPCSTOP
Register C
Compare RC =
Register B
Compare RB =
Register A
Compare RA =
CLKDIS
CLKEN
CLKSTA
S
R
Q
S
R
Q
CPCS
CPBS
CPAS
COVFS
ETRGS
TC1_SR
TC1_IMR
INT
OVF
RESET
CLK
16-bit Counter
WAVSEL
WAVSEL
Trig
ENETRG
CLKI
SWTRG
EEVTEDG
Edge Detector
Timer/Counter Channel
TCCLKS
BURST
1
EEVT
TIMER_CLOCK1
TIMER_CLOCK2
TIMER_CLOCK3
TIMER_CLOCK4
TIMER_CLOCK5
XC0
XC1
XC2
SYNC
TIOB

30.5.11.1 WAVSEL = 00

When WAVSEL = 00, the value of TC_CV is incremented from 0 to 0xFFFF. Once 0xFFFF has been reached, the value of TC_CV is reset. Incrementation of TC_CV starts again and the cycle continues. See Figure 30-7.

An external event trigger or a software trigger can reset the value of TC_CV. It is important to note that the trigger may occur at any time. See Figure 30-8.

RC Compare cannot be programmed to generate a trigger in this configuration. At the same time, RC Compare can stop the counter clock (CPCSTOP = 1 in TC_CMR) and/or disable the counter clock (CPCDIS = 1 in TC_CMR).

Figure 30-7. WAVSEL= 00 without trigger

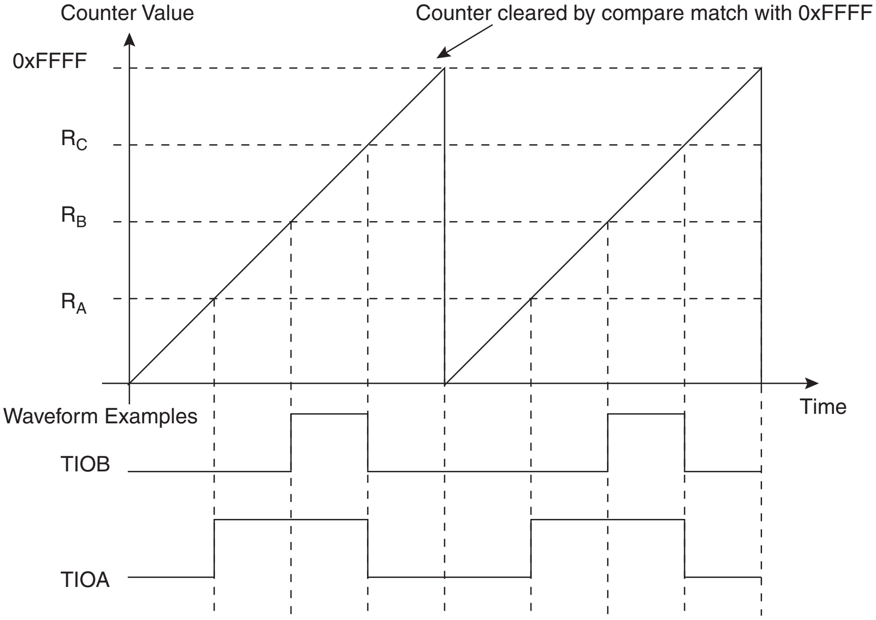
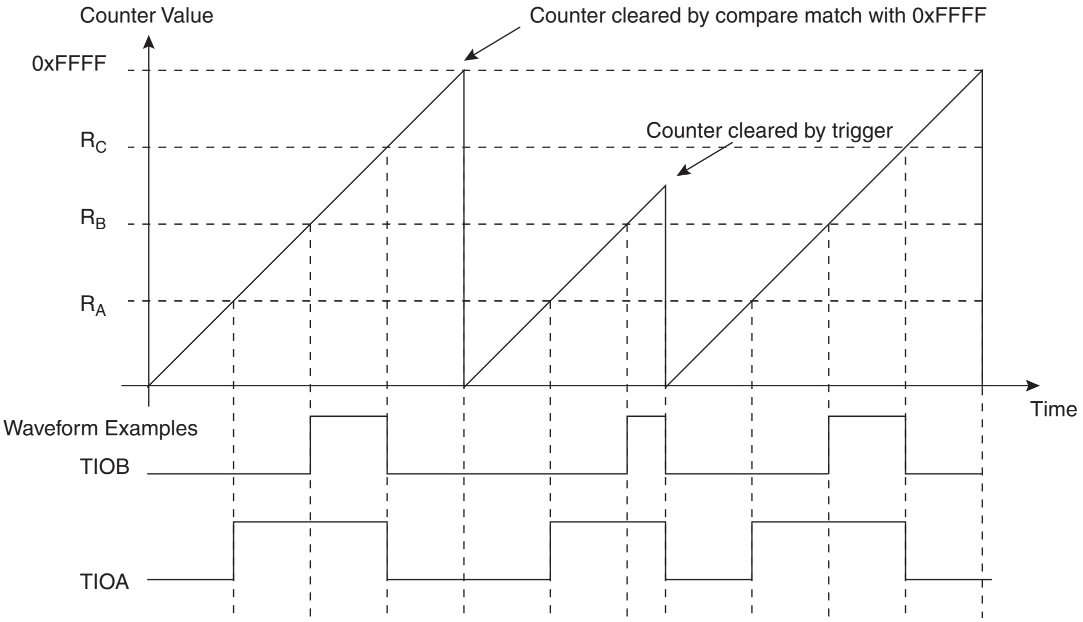

Counter Value
Counter cleared by compare match with 0xFFFF
0xFFFF
R_C
R_B
R_A
Time
Waveform Examples
TIOB
TIOA

Figure 30-8. WAVSEL= 00 with trigger


Counter Value
Counter cleared by compare match with 0xFFFF
0xFFFF
R_C
Counter cleared by trigger
R_B
R_A
Time
Waveform Examples
TIOB
TIOA

30.5.11.2 WAVSEL = 10

When WAVSEL = 10, the value of TC_CV is incremented from 0 to the value of RC, then automatically reset on a RC Compare. Once the value of TC_CV has been reset, it is then incremented and so on. See Figure 30-9.

It is important to note that TC_CV can be reset at any time by an external event or a software trigger if both are programmed correctly. See Figure 30-10.

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

Figure 30-9. WAVSEL = 10 Without Trigger

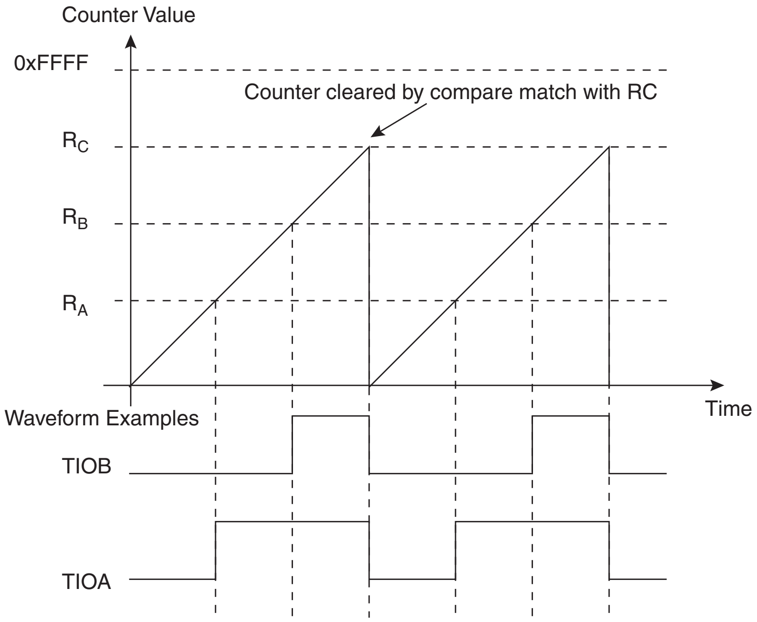
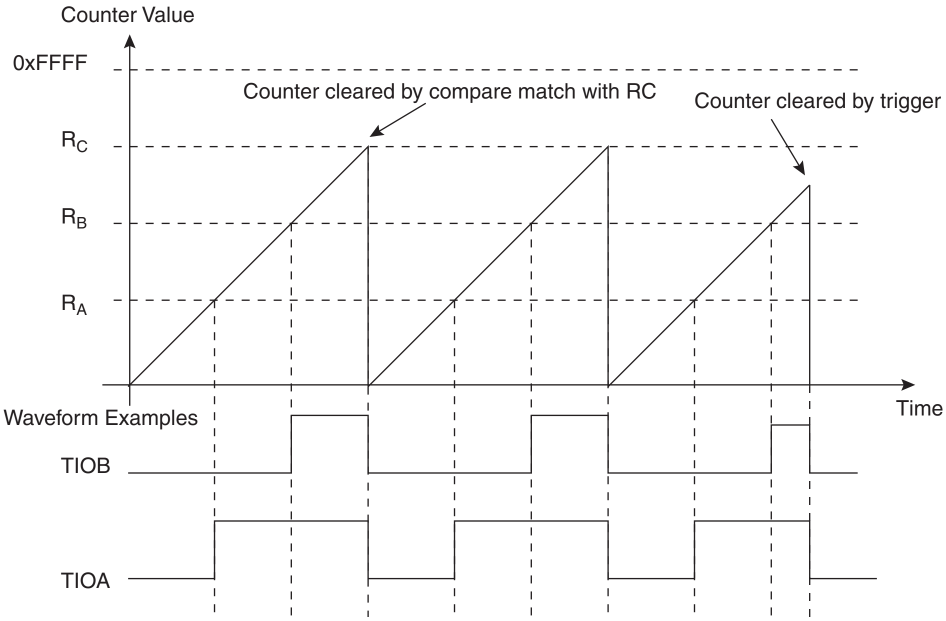

Counter Value
0xFFFF
Counter cleared by compare match with RC
R_C
R_B
R_A
Time
Waveform Examples
TIOB
TIOA

Figure 30-10. WAVSEL = 10 With Trigger


Counter Value
0xFFFF
Counter cleared by compare match with RC
Counter cleared by trigger
R_C
R_B
R_A
Waveform Examples
Time
TIOB
TIOA

30.5.11.3 WAVSEL = 01

When WAVSEL = 01, the value of TC_CV is incremented from 0 to 0xFFFF. Once 0xFFFF is reached, the value of TC_CV is decremented to 0, then re-incremented to 0xFFFF and so on. See Figure 30-11.

A trigger such as an external event or a software trigger can modify TC_CV at any time. If a trigger occurs while TC_CV is incrementing, TC_CV then decrements. If a trigger is received while TC_CV is decrementing, TC_CV then increments. See Figure 30-12.

RC Compare cannot be programmed to generate a trigger in this configuration.

At the same time, RC Compare can stop the counter clock (CPCSTOP = 1) and/or disable the counter clock (CPCDIS = 1).

Figure 30-11. WAVSEL = 01 Without Trigger

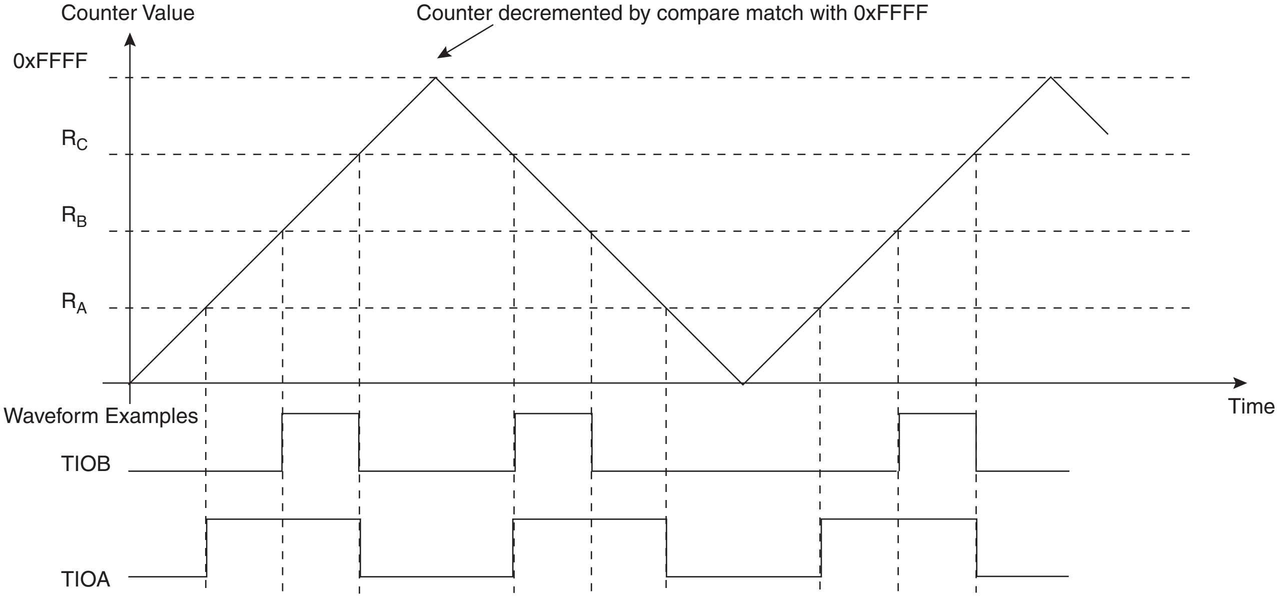
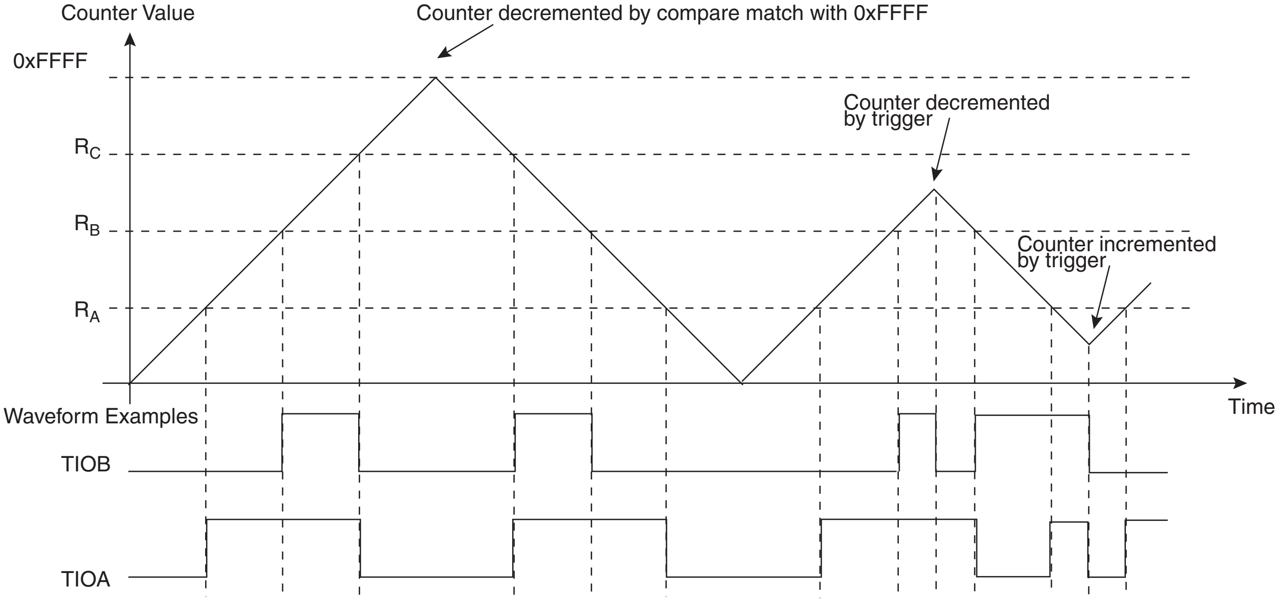

Counter Value
Counter decremented by compare match with 0xFFFF
0xFFFF
R_C
R_B
R_A
Time
Waveform Examples
TIOB
TIOA

Figure 30-12. WAVSEL = 01 With Trigger


Counter Value
Counter decremented by compare match with 0xFFFF
0xFFFF
R_C
R_B
R_A
Counter decremented by trigger
Counter incremented by trigger
Time
Waveform Examples
TIOB
TIOA

30.5.11.4 WAVSEL = 11

When WAVSEL = 11, the value of TC_CV is incremented from 0 to RC. Once RC is reached, the value of TC_CV is decremented to 0, then re-incremented to RC and so on. See Figure 30-13.

A trigger such as an external event or a software trigger can modify TC_CV at any time. If a trigger occurs while TC_CV is incrementing, TC_CV then decrements. If a trigger is received while TC_CV is decrementing, TC_CV then increments. See Figure 30-14.

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

Figure 30-13. WAVSEL = 11 Without Trigger

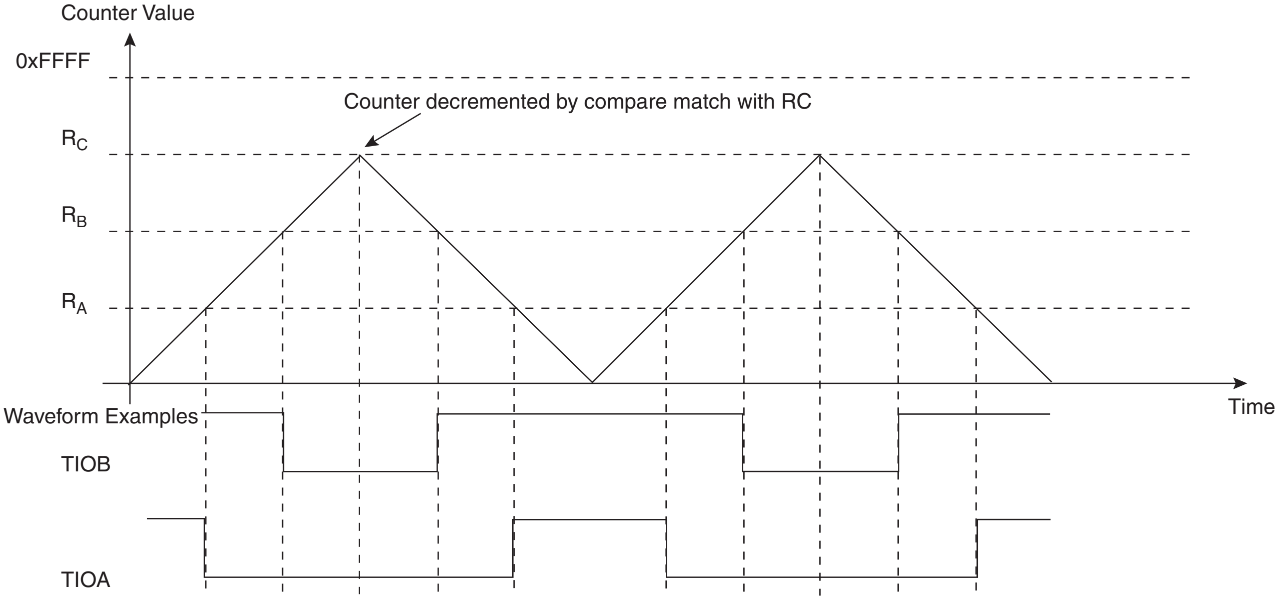
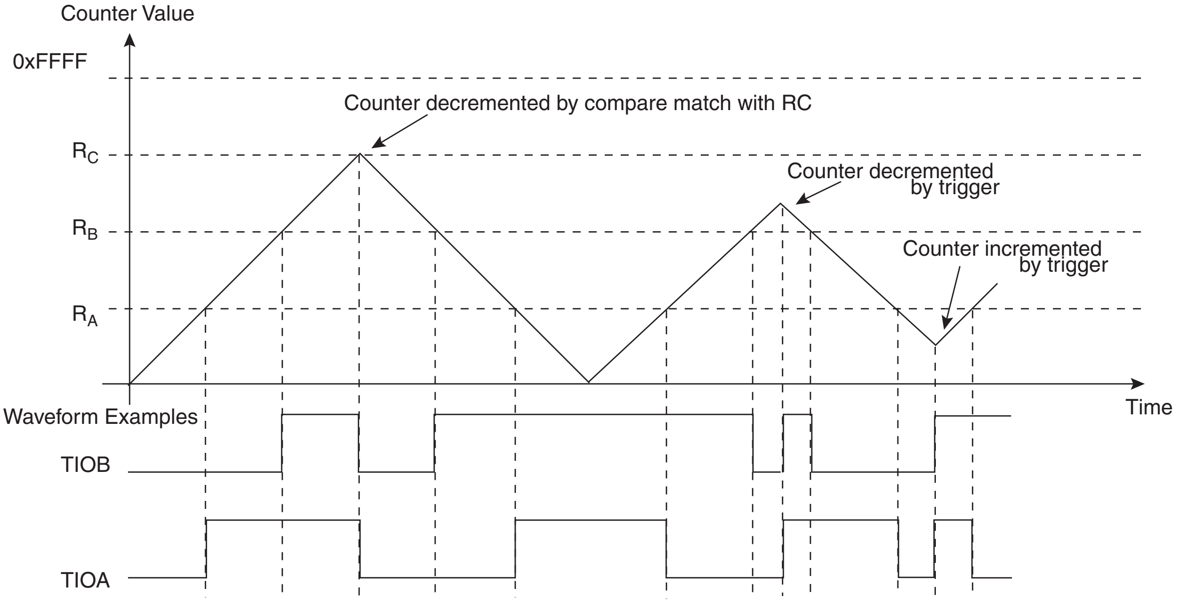

Counter Value
0xFFFF
Counter decremented by compare match with RC
R_C
R_B
R_A
Time
Waveform Examples
TIOB
TIOA

Figure 30-14. WAVSEL = 11 With Trigger


Counter Value
0xFFFF
Counter decremented by compare match with RC
R_C
Counter decremented by trigger
R_B
Counter incremented by trigger
R_A
Time
Waveform Examples
TIOB
TIOA

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

If TIOB is defined as an external event signal (EEVT = 0), TIOB is no longer used as an output and the compare register B is not used to generate waveforms and subsequently no IRQs. In this case 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 and the software trigger are also available as triggers. RC Compare can also be used as a trigger depending on the parameter WAVSEL.

30.5.13 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.

30.6 Timer Counter (TC) User Interface

Table 30-4. TC Global Memory Map

Offset	Channel/Register	Name	Access	Reset Value
0x00	TC Channel 0	See Table 30-5		
0x40	TC Channel 1	See Table 30-5		
0x80	TC Channel 2	See Table 30-5		
0xC0	TC Block Control Register	TC_BCR	Write-only	–
0xC4	TC Block Mode Register	TC_BMR	Read/Write	0

TC_BCR (Block Control Register) and TC_BMR (Block Mode Register) control the whole TC block. TC channels are controlled by the registers listed in Table 30-5. The offset of each of the channel registers in Table 30-5 is in relation to the offset of the corresponding channel as mentioned in Table 30-5.

Table 30-5. TC Channel Memory Map

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

Notes: 1. Read-only if WAVE = 0

30.6.1 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.

30.6.2 TC Block Mode Register

Register Name: TC_BMR

Access Type: Read/Write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	TC2XC2S		TCXC1S		TC0XC0S	

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

TC0XC0S		Signal Connected to XC0
0	0	TCLK0
0	1	none
1	0	TIOA1
1	1	TIOA2

- **TC1XC1S: External Clock Signal 1 Selection**

TC1XC1S		Signal Connected to XC1
0	0	TCLK1
0	1	none
1	0	TIOA0
1	1	TIOA2

- **TC2XC2S: External Clock Signal 2 Selection**

TC2XC2S		Signal Connected to XC2
0	0	TCLK2
0	1	none
1	0	TIOA0
1	1	TIOA1

30.6.3 TC Channel Control Register

Register Name: TC_CCR

Access Type: Write-only

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

- **CLKEN: Counter Clock Enable Command**

0 = No effect.

1 = Enables the clock if CLKDIS is not 1.

- **CLKDIS: Counter Clock Disable Command**

0 = No effect.

1 = Disables the clock.

- **SWTRG: Software Trigger Command**

0 = No effect.

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

30.6.4 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	TIMER_CLOCK1
0	0	1	TIMER_CLOCK2
0	1	0	TIMER_CLOCK3
0	1	1	TIMER_CLOCK4
1	0	0	TIMER_CLOCK5
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI: Clock Invert**

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

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

- **BURST: Burst Signal Selection**

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

- **LDBSTOP: Counter Clock Stopped with RB Loading**

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

1 = Counter clock is stopped when RB loading occurs.

- **LDBDIS: Counter Clock Disable with RB Loading**

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

1 = Counter clock is disabled when RB loading occurs.

- **ETRGEDG: External Trigger Edge Selection**

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

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

0 = TIOB is used as an external trigger.

1 = TIOA is used as an external trigger.

- **CPCTRG: RC Compare Trigger Enable**

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

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

- **WAVE**

0 = Capture Mode is enabled.

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

- **LDRA: RA Loading Selection**

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

- **LDRB: RB Loading Selection**

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

30.6.5 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	WAVSEL		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	TIMER_CLOCK1
0	0	1	TIMER_CLOCK2
0	1	0	TIMER_CLOCK3
0	1	1	TIMER_CLOCK4
1	0	0	TIMER_CLOCK5
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI: Clock Invert**

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

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

- **BURST: Burst Signal Selection**

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

- **CPCSTOP: Counter Clock Stopped with RC Compare**

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

1 = Counter clock is stopped when counter reaches RC.

- **CPCDIS: Counter Clock Disable with RC Compare**

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

1 = Counter clock is disabled when counter reaches RC.

- **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 and subsequently no IRQs.*

- **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.

- **WAVSEL: Waveform Selection**

WAVSEL		Effect
0	0	UP mode without automatic trigger on RC Compare
1	0	UP mode with automatic trigger on RC Compare
0	1	UPDOWN mode without automatic trigger on RC Compare
1	1	UPDOWN mode with automatic trigger on RC Compare

- **WAVE = 1**

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

1 = Waveform Mode is enabled.

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

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

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

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

- **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

30.6.6 TC Counter Value Register

Register Name: TC_CV

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
CV							

7	6	5	4	3	2	1	0
CV							

- **CV: Counter Value**

CV contains the counter value in real time.

30.6.7 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.

30.6.8 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.

30.6.9 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.

30.6.10 TC Status Register

Register Name: TC_SR

Access Type: Read-only

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

23	22	21	20	19	18	17	16
–	–	–	–	–	MTIOB	MTIOA	CLKSTA

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

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

- **COVFS: Counter Overflow Status**

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

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

- **LOVRS: Load Overrun Status**

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

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

- **CPAS: RA Compare Status**

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

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

- **CPBS: RB Compare Status**

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

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

- **CPCS: RC Compare Status**

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

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

- **LDRAS: RA Loading Status**

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

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

- **LDRBS: RB Loading Status**

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

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

- **ETRGS: External Trigger Status**

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

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

- **CLKSTA: Clock Enabling Status**

0 = Clock is disabled.

1 = Clock is enabled.

- **MTIOA: TIOA Mirror**

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

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

- **MTIOB: TIOB Mirror**

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

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

30.6.11 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.

30.6.12 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.

30.6.13 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.

31. USB Device Port (UDP)

31.1 Description

The USB Device Port (UDP) is compliant with the Universal Serial Bus (USB) V2.0 full-speed device specification.

Each endpoint can be configured in one of several USB transfer types. It can be associated with one or two banks of a dual-port RAM used to store the current data payload. If two banks are used, one DPR bank is read or written by the processor, while the other is read or written by the USB device peripheral. This feature is mandatory for isochronous endpoints. Thus the device maintains the maximum bandwidth (1M bytes/s) by working with endpoints with two banks of DPR.

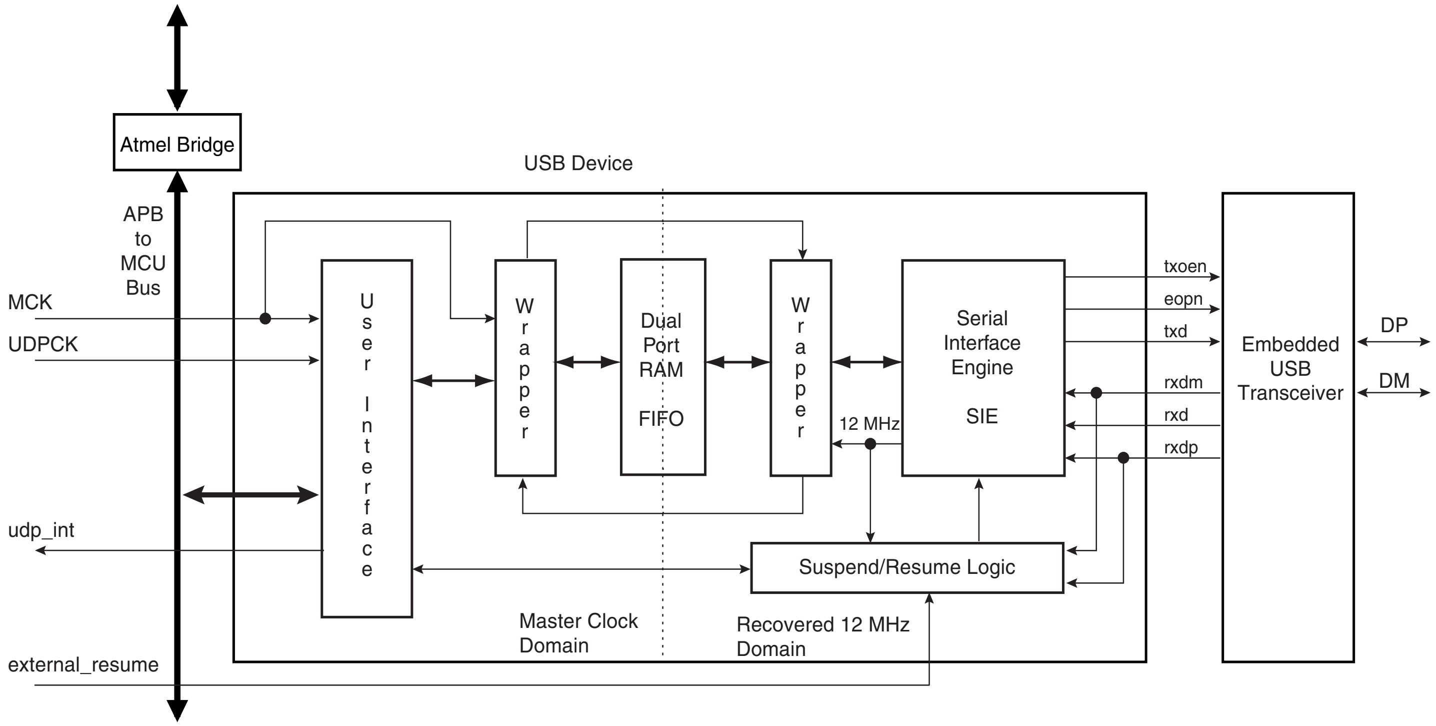
Table 31-1. USB Endpoint Description

Endpoint Number	Mnemonic	Dual-Bank	Max. Endpoint Size	Endpoint Type
0	EP0	No	8	Control/Bulk/Interrupt
1	EP1	Yes	64	Bulk/Iso/Interrupt
2	EP2	Yes	64	Bulk/Iso/Interrupt
3	EP3	No	64	Control/Bulk/Interrupt
4	EP4	Yes	256	Bulk/Iso/Interrupt
5	EP5	Yes	256	Bulk/Iso/Interrupt

Suspend and resume are automatically detected by the USB device, which notifies the processor by raising an interrupt. Depending on the product, an external signal can be used to send a wake up to the USB host controller.

31.2 Block Diagram

Figure 31-1. Block Diagram


Atmel Bridge
APB to MCU Bus
MCK
UDPCK
udp_int
external_resume
USB Device
User Interface
Wrapper
Dual Port RAM FIFO
Wrapper
12 MHz
Serial Interface Engine SIE
Suspend/Resume Logic
Master Clock Domain
Recovered 12 MHz Domain
txoen
eopn
txd
rxdm
rxd
rxdp
Embedded USB Transceiver
DP
DM

Access to the UDP is via the APB bus interface. Read and write to the data FIFO are done by reading and writing 8-bit values to APB registers.

The UDP peripheral requires two clocks: one peripheral clock used by the MCK domain and a 48 MHz clock used by the 12 MHz domain.

A USB 2.0 full-speed pad is embedded and controlled by the Serial Interface Engine (SIE).

The signal external_resume is optional. It allows the UDP peripheral to wake up once in system mode. The host is then notified that the device asks for a resume. This optional feature must be also negotiated with the host during the enumeration.

31.3 Product Dependencies

For further details on the USB Device hardware implementation, see the specific Product Properties document.

The USB physical transceiver is integrated into the product. The bidirectional differential signals DP and DM are available from the product boundary.

One I/O line may be used by the application to check that VBUS is still available from the host. Self-powered devices may use this entry to be notified that the host has been powered off. In this case, the pullup on DP must be disabled in order to prevent feeding current to the host. The application should disconnect the transceiver, then remove the pullup.

31.3.1 I/O Lines

DP and DM are not controlled by any PIO controllers. The embedded USB physical transceiver is controlled by the USB device peripheral.

To reserve an I/O line to check VBUS, the programmer must first program the PIO controller to assign this I/O in input PIO mode.

31.3.2 Power Management

The USB device peripheral requires a 48 MHz clock. This clock must be generated by a PLL with an accuracy of ± 0.25%.

Thus, the USB device receives two clocks from the Power Management Controller (PMC): the master clock, MCK, used to drive the peripheral user interface, and the UDPCK, used to interface with the bus USB signals (recovered 12 MHz domain).

WARNING: The UDP peripheral clock in the Power Management Controller (PMC) must be enabled before any read/write operations to the UDP registers including the UDP_TXVC register.

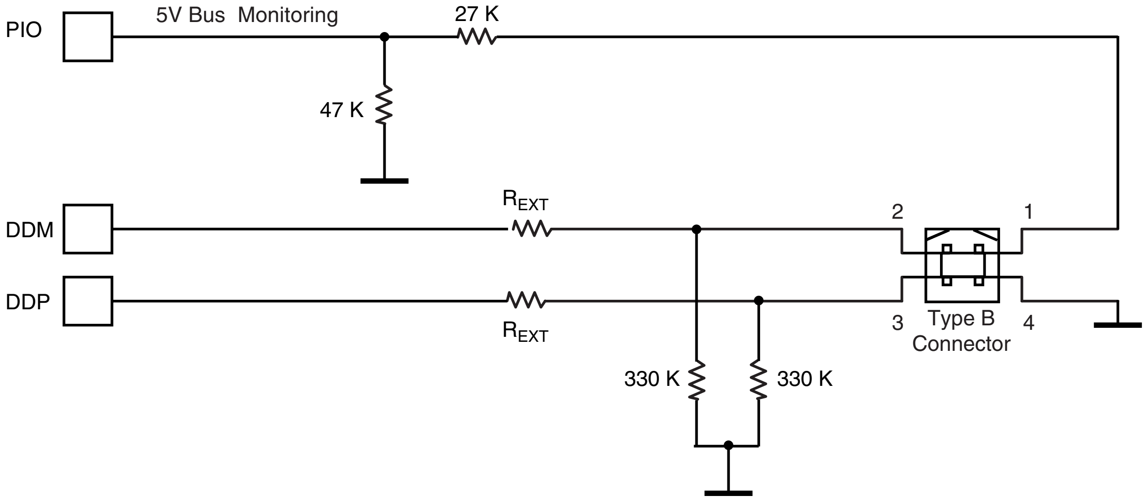
31.3.3 Interrupt

The USB device interface has an interrupt line connected to the Advanced Interrupt Controller (AIC).

Handling the USB device interrupt requires programming the AIC before configuring the UDP.

31.4 Typical Connection

Figure 31-2. Board Schematic to Interface Device Peripheral



31.4.1 USB Device Transceiver

The USB device transceiver is embedded in the product. A few discrete components are required as follows:

- the application detects all device states as defined in chapter 9 of the USB specification;
  - VBUS monitoring
- to reduce power consumption the host is disconnected
- for line termination.

31.4.2 VBUS Monitoring

VBUS monitoring is required to detect host connection. VBUS monitoring is done using a standard PIO with internal pullup disabled. When the host is switched off, it should be considered as a disconnect, the pullup must be disabled in order to prevent powering the host through the pull-up resistor.

When the host is disconnected and the transceiver is enabled, then DDP and DDM are floating. This may lead to over consumption. A solution is to connect 330 KΩ pulldowns on DP and DM. These pulldowns do not alter DDP and DDM signal integrity.

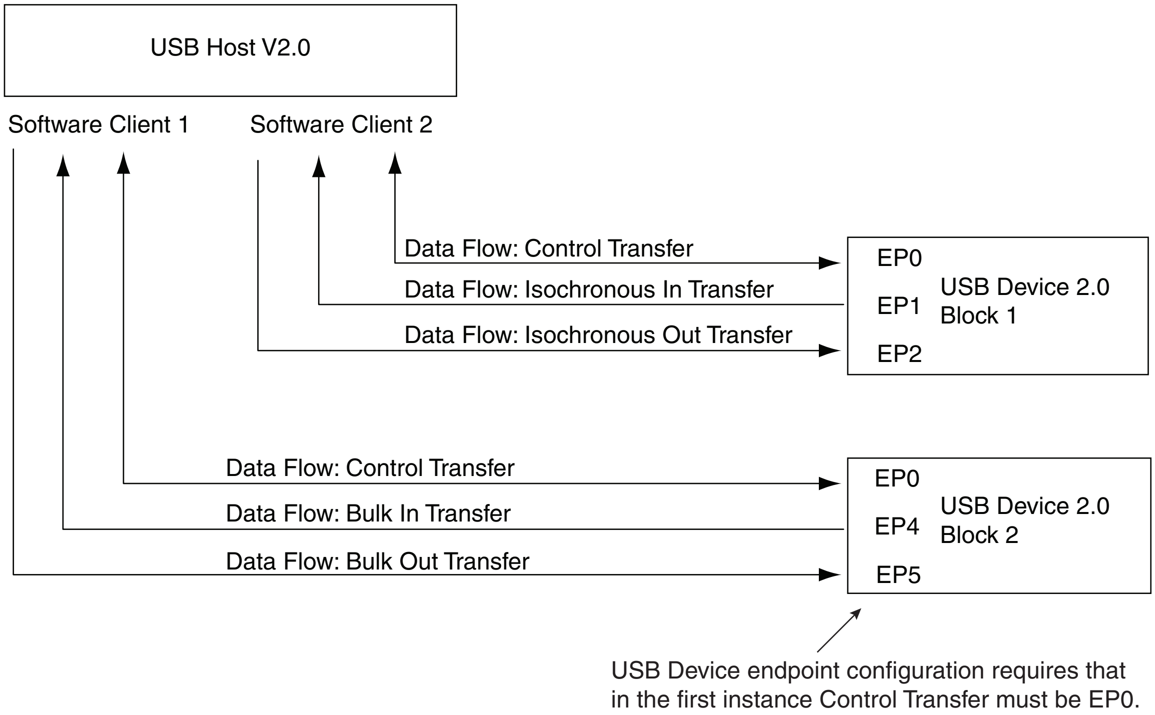
A termination serial resistor must be connected to DP and DM. The resistor value is defined in the electrical specification of the product (R_{EXT}).

31.5 Functional Description

31.5.1 USB V2.0 Full-speed Introduction

The USB V2.0 full-speed provides communication services between host and attached USB devices. Each device is offered with a collection of communication flows (pipes) associated with each endpoint. Software on the host communicates with a USB device through a set of communication flows.

Figure 31-3. Example of USB V2.0 Full-speed Communication Control


USB Host V2.0
Software Client 1
Software Client 2
Data Flow: Control Transfer
Data Flow: Isochronous In Transfer
Data Flow: Isochronous Out Transfer
EP0
EP1
EP2
USB Device 2.0
Block 1
Data Flow: Control Transfer
Data Flow: Bulk In Transfer
Data Flow: Bulk Out Transfer
EP0
EP4
EP5
USB Device 2.0
Block 2

USB Device endpoint configuration requires that in the first instance Control Transfer must be EP0.

The Control Transfer endpoint EP0 is always used when a USB device is first configured (USB v. 2.0 specifications).

31.5.1.1 USB V2.0 Full-speed Transfer Types

A communication flow is carried over one of four transfer types defined by the USB device.

Table 31-2. USB Communication Flow

Transfer	Direction	Bandwidth	Supported Endpoint Size	Error Detection	Retrying
Control	Bidirectional	Not guaranteed	8, 16, 32, 64	Yes	Automatic
Isochronous	Unidirectional	Guaranteed	256	Yes	No
Interrupt	Unidirectional	Not guaranteed	≤64	Yes	Yes
Bulk	Unidirectional	Not guaranteed	8, 16, 32, 64	Yes	Yes

31.5.1.2 USB Bus Transactions

Each transfer results in one or more transactions over the USB bus. There are three kinds of transactions flowing across the bus in packets:

1. Setup Transaction
2. Data IN Transaction
3. Data OUT Transaction

31.5.1.3 USB Transfer Event Definitions

As indicated below, transfers are sequential events carried out on the USB bus.

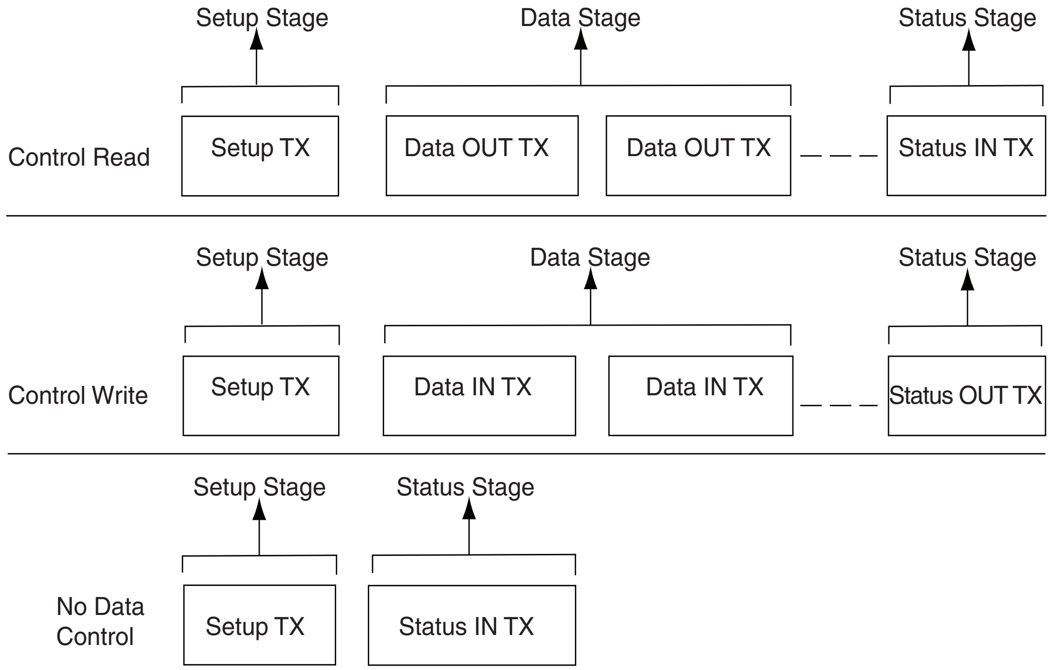
Table 31-3. USB Transfer Events

Control Transfers[(1)] [(3)]	• Setup transaction > Data IN transactions > Status OUT transaction • Setup transaction > Data OUT transactions > Status IN transaction • Setup transaction > Status IN transaction
Interrupt IN Transfer (device toward host)	• Data IN transaction > Data IN transaction
Interrupt OUT Transfer (host toward device)	• Data OUT transaction > Data OUT transaction
Isochronous IN Transfer[(2)] (device toward host)	• Data IN transaction > Data IN transaction
Isochronous OUT Transfer[(2)] (host toward device)	• Data OUT transaction > Data OUT transaction
Bulk IN Transfer (device toward host)	• Data IN transaction > Data IN transaction
Bulk OUT Transfer (host toward device)	• Data OUT transaction > Data OUT transaction

Notes:
1. Control transfer must use endpoints with no ping-pong attributes.
2. Isochronous transfers must use endpoints with ping-pong attributes.
3. Control transfers can be aborted using a stall handshake.

A status transaction is a special type of host-to-device transaction used only in a control transfer. The control transfer must be performed using endpoints with no ping-pong attributes. According to the control sequence (read or write), the USB device sends or receives a status transaction.

Figure 31-4. Control Read and Write Sequences


Setup Stage
Data Stage
Status Stage
Control Read
Setup TX
Data OUT TX
Data OUT TX
Status IN TX
Setup Stage
Data Stage
Status Stage
Control Write
Setup TX
Data IN TX
Data IN TX
Status OUT TX
Setup Stage
Status Stage
No Data Control
Setup TX
Status IN TX

Notes:
1. During the Status IN stage, the host waits for a zero length packet (Data IN transaction with no data) from the device using DATA1 PID. Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev. 2.0,* for more information on the protocol layer.
2. During the Status OUT stage, the host emits a zero length packet to the device (Data OUT transaction with no data).

31.5.2 Handling Transactions with USB V2.0 Device Peripheral

31.5.2.1 Setup Transaction

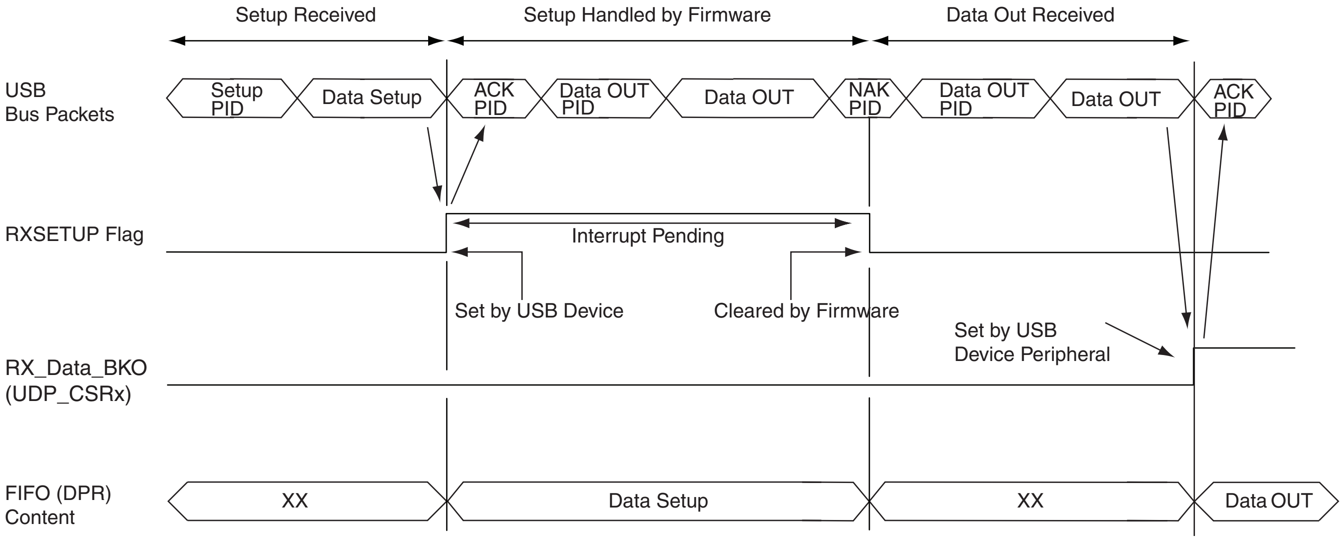
Setup is a special type of host-to-device transaction used during control transfers. Control transfers must be performed using endpoints with no ping-pong attributes. A setup transaction needs to be handled as soon as possible by the firmware. It is used to transmit requests from the host to the device. These requests are then handled by the USB device and may require more arguments. The arguments are sent to the device by a Data OUT transaction which follows the setup transaction. These requests may also return data. The data is carried out to the host by the next Data IN transaction which follows the setup transaction. A status transaction ends the control transfer.

When a setup transfer is received by the USB endpoint:

- The USB device automatically acknowledges the setup packet
- RXSETUP is set in the UDP_ CSRx register
- An endpoint interrupt is generated while the RXSETUP is not cleared. This interrupt is carried out to the microcontroller if interrupts are enabled for this endpoint.

Thus, firmware must detect the RXSETUP polling the UDP_ CSRx or catching an interrupt, read the setup packet in the FIFO, then clear the RXSETUP. RXSETUP cannot be cleared before the setup packet has been read in the FIFO. Otherwise, the USB device would accept the next Data OUT transfer and overwrite the setup packet in the FIFO.

Figure 31-5. Setup Transaction Followed by a Data OUT Transaction



Setup Received
Setup Handled by Firmware
Data Out Received
USB Bus Packets
Setup PID
Data Setup
ACK PID
Data OUT PID
Data OUT
NAK PID
Data OUT PID
Data OUT
ACK PID
RXSETUP Flag
Interrupt Pending
Set by USB Device
Cleared by Firmware
Set by USB Device Peripheral
RX_Data_BKO (UDP_CSRx)
FIFO (DPR) Content
XX
Data Setup
XX
Data OUT

31.5.2.2 Data IN Transaction

Data IN transactions are used in control, isochronous, bulk and interrupt transfers and conduct the transfer of data from the device to the host. Data IN transactions in isochronous transfer must be done using endpoints with ping-pong attributes.

Using Endpoints Without Ping-pong Attributes

To perform a Data IN transaction using a non ping-pong endpoint:

1. The application checks if it is possible to write in the FIFO by polling TXPKTRDY in the endpoint's UDP_ CSRx register (TXPKTRDY must be cleared).
2. The application writes the first packet of data to be sent in the endpoint's FIFO, writing zero or more byte values in the endpoint's UDP_ FDRx register,
3. The application notifies the USB peripheral it has finished by setting the TXPKTRDY in the endpoint's UDP_ CSRx register.
4. The application is notified that the endpoint's FIFO has been released by the USB device when TXCOMP in the endpoint's UDP_ CSRx register has been set. Then an interrupt for the corresponding endpoint is pending while TXCOMP is set.
5. The microcontroller writes the second packet of data to be sent in the endpoint's FIFO, writing zero or more byte values in the endpoint's UDP_ FDRx register,
6. The microcontroller notifies the USB peripheral it has finished by setting the TXPKTRDY in the endpoint's UDP_ CSRx register.
7. The application clears the TXCOMP in the endpoint's UDP_ CSRx.

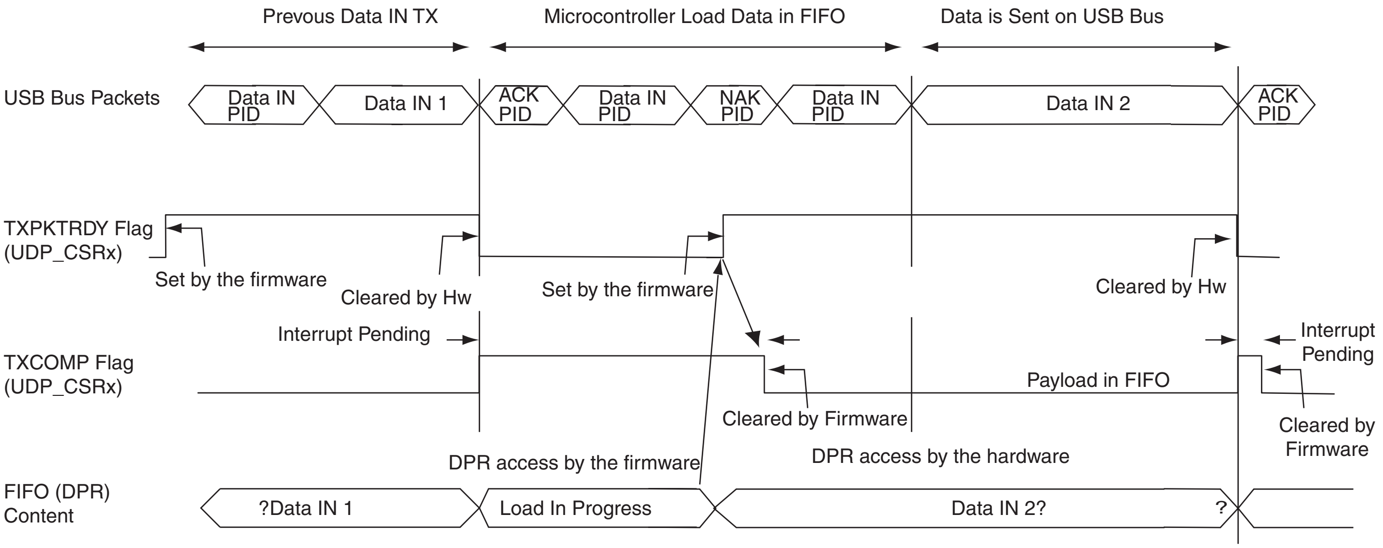
After the last packet has been sent, the application must clear TXCOMP once this has been set.

TXCOMP is set by the USB device when it has received an ACK PID signal for the Data IN packet. An interrupt is pending while TXCOMP is set.

Warning: TX_COMP must be cleared after TX_PKTRDY has been set.

Note: Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev 2.0,* for more information on the Data IN protocol layer.

Figure 31-6. Data IN Transfer for Non Ping-pong Endpoint

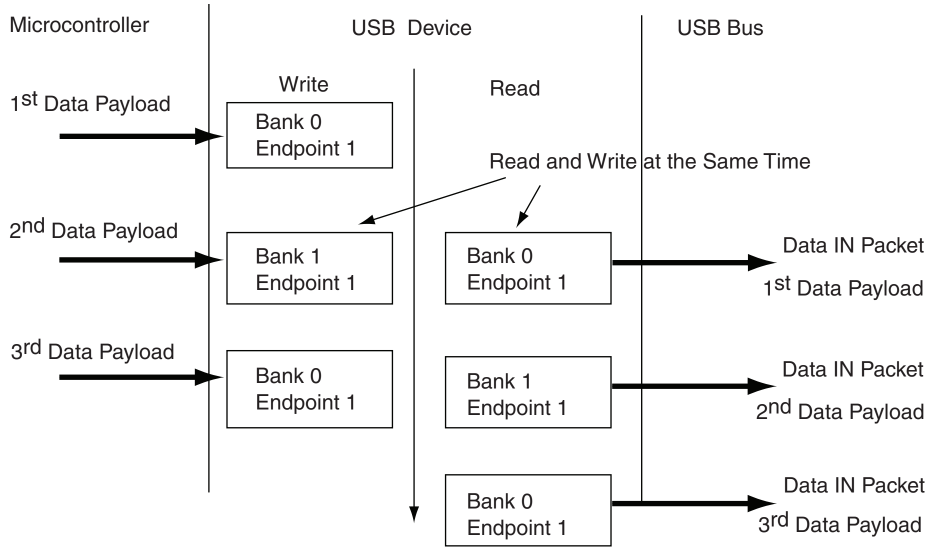


Prevous Data IN TX
Microcontroller Load Data in FIFO
Data is Sent on USB Bus
USB Bus Packets
Data IN PID
Data IN 1
ACK PID
Data IN PID
NAK PID
Data IN PID
Data IN 2
ACK PID
TXPKTRDY Flag (UDP_CSRx)
Set by the firmware
Cleared by Hw
Set by the firmware
Cleared by Hw
Interrupt Pending
Interrupt Pending
TXCOMP Flag (UDP_CSRx)
Payload in FIFO
Cleared by Firmware
Cleared by Firmware
DPR access by the firmware
DPR access by the hardware
FIFO (DPR) Content
?Data IN 1
Load In Progress
Data IN 2?
?

Using Endpoints With Ping-pong Attribute

The use of an endpoint with ping-pong attributes is necessary during isochronous transfer. This also allows handling the maximum bandwidth defined in the USB specification during bulk transfer. To be able to guarantee a constant or the maximum bandwidth, the microcontroller must prepare the next data payload to be sent while the current one is being sent by the USB device. Thus two banks of memory are used. While one is available for the microcontroller, the other one is locked by the USB device.

Figure 31-7. Bank Swapping Data IN Transfer for Ping-pong Endpoints

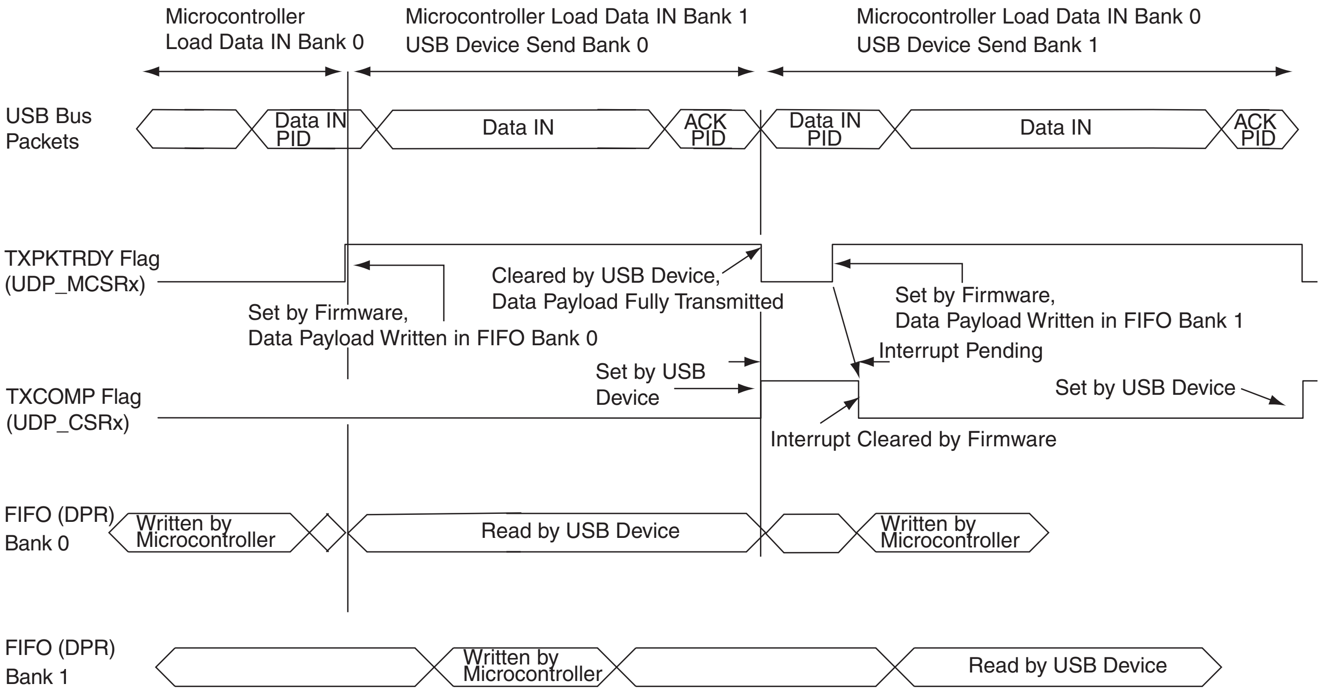


Microcontroller
USB Device
USB Bus
Write
Read
1st Data Payload
Bank 0 Endpoint 1
Read and Write at the Same Time
2nd Data Payload
Bank 1 Endpoint 1
Bank 0 Endpoint 1
Data IN Packet
1st Data Payload
3rd Data Payload
Bank 0 Endpoint 1
Bank 1 Endpoint 1
Data IN Packet
2nd Data Payload
Bank 0 Endpoint 1
Data IN Packet
3rd Data Payload

When using a ping-pong endpoint, the following procedures are required to perform Data IN transactions:

1. The microcontroller checks if it is possible to write in the FIFO by polling TXPKTRDY to be cleared in the endpoint's UDP_ CSRx register.
2. The microcontroller writes the first data payload to be sent in the FIFO (Bank 0), writing zero or more byte values in the endpoint's UDP_ FDRx register.
3. The microcontroller notifies the USB peripheral it has finished writing in Bank 0 of the FIFO by setting the TXPKTRDY in the endpoint's UDP_ CSRx register.
4. Without waiting for TXPKTRDY to be cleared, the microcontroller writes the second data payload to be sent in the FIFO (Bank 1), writing zero or more byte values in the endpoint's UDP_ FDRx register.
5. The microcontroller is notified that the first Bank has been released by the USB device when TXCOMP in the endpoint's UDP_ CSRx register is set. An interrupt is pending while TXCOMP is being set.
6. Once the microcontroller has received TXCOMP for the first Bank, it notifies the USB device that it has prepared the second Bank to be sent rising TXPKTRDY in the endpoint's UDP_ CSRx register.
7. At this step, Bank 0 is available and the microcontroller can prepare a third data payload to be sent.

Figure 31-8. Data IN Transfer for Ping-pong Endpoint


Microcontroller
Load Data IN Bank 0
Microcontroller Load Data IN Bank 1
USB Device Send Bank 0
Microcontroller Load Data IN Bank 0
USB Device Send Bank 1
USB Bus
Packets
Data IN
PID
Data IN
ACK
PID
Data IN
PID
Data IN
ACK
PID
TXPKTRDY Flag
(UDP_MCSRx)
Cleared by USB Device,
Data Payload Fully Transmitted
Set by Firmware,
Data Payload Written in FIFO Bank 0
Set by Firmware,
Data Payload Written in FIFO Bank 1
Interrupt Pending
Set by USB
Device
Set by USB Device
TXCOMP Flag
(UDP_CSRx)
Interrupt Cleared by Firmware
FIFO (DPR)
Bank 0
Written by
Microcontroller
Read by USB Device
Written by
Microcontroller
FIFO (DPR)
Bank 1
Written by
Microcontroller
Read by USB Device

Warning: There is software critical path due to the fact that once the second bank is filled, the driver has to wait for TX_COMP to set TX_PKTRDY. If the delay between receiving TX_COMP is set and TX_PKTRDY is set is too long, some Data IN packets may be NACKed, reducing the bandwidth.

Warning: TX_COMP must be cleared after TX_PKTRDY has been set.

31.5.2.3 Data OUT Transaction

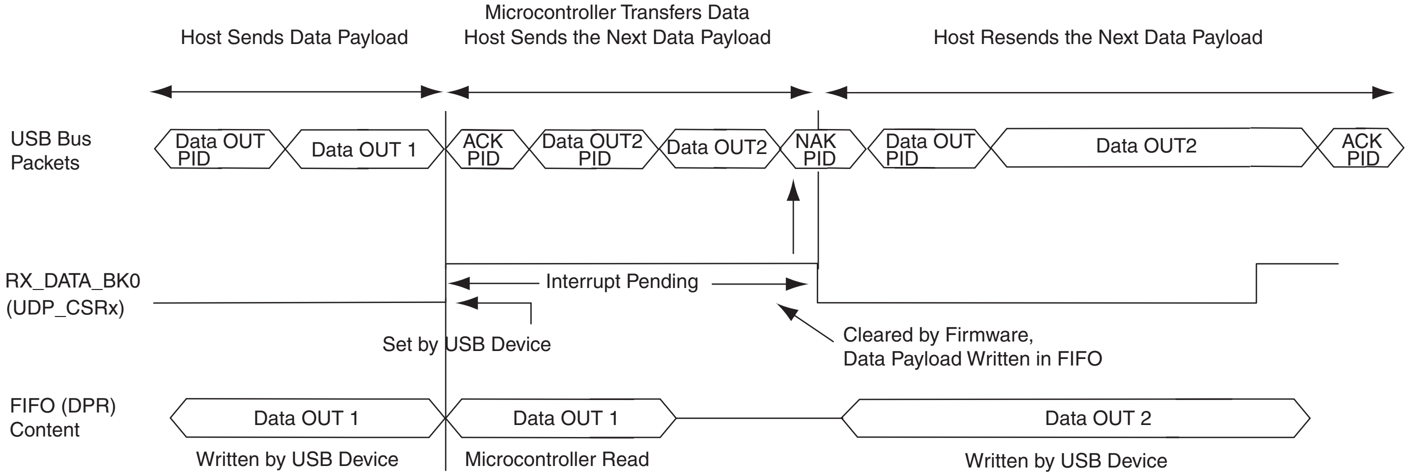
Data OUT transactions are used in control, isochronous, bulk and interrupt transfers and conduct the transfer of data from the host to the device. Data OUT transactions in isochronous transfers must be done using endpoints with ping-pong attributes.

Data OUT Transaction Without Ping-pong Attributes

To perform a Data OUT transaction, using a non ping-pong endpoint:

1. The host generates a Data OUT packet.
2. This packet is received by the USB device endpoint. While the FIFO associated to this endpoint is being used by the microcontroller, a NAK PID is returned to the host. Once the FIFO is available, data are written to the FIFO by the USB device and an ACK is automatically carried out to the host.
3. The microcontroller is notified that the USB device has received a data payload polling RX_DATA_BK0 in the endpoint's UDP_ CSRx register. An interrupt is pending for this endpoint while RX_DATA_BK0 is set.
4. The number of bytes available in the FIFO is made available by reading RXBYTECNT in the endpoint's UDP_ CSRx register.
5. The microcontroller carries out data received from the endpoint's memory to its memory. Data received is available by reading the endpoint's UDP_ FDRx register.
6. The microcontroller notifies the USB device that it has finished the transfer by clearing RX_DATA_BK0 in the endpoint's UDP_ CSRx register.
7. A new Data OUT packet can be accepted by the USB device.

Figure 31-9. Data OUT Transfer for Non Ping-pong EndpointsAn interrupt is pending while the flag RX_DATA_BK0 is

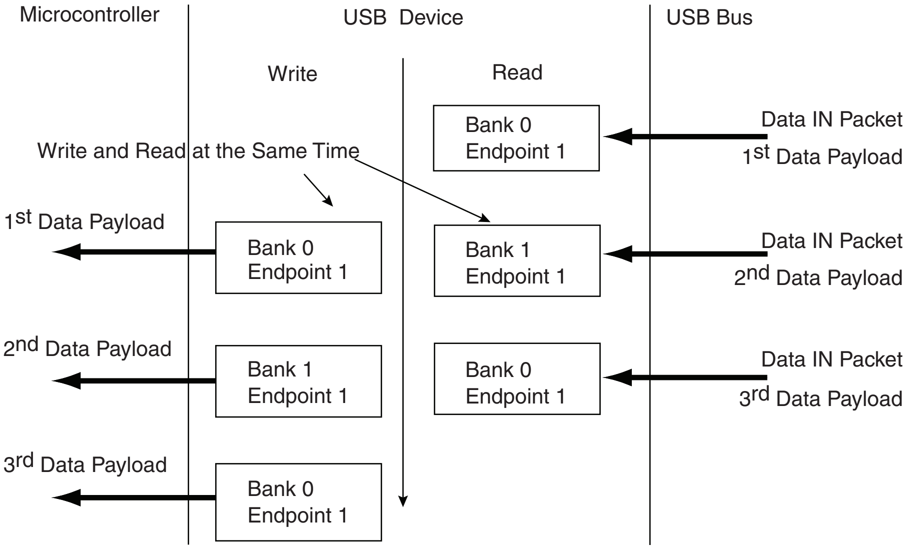

Host Sends Data Payload
Microcontroller Transfers Data
Host Sends the Next Data Payload
Host Resends the Next Data Payload
USB Bus Packets
Data OUT PID
Data OUT 1
ACK PID
Data OUT2 PID
Data OUT2
NAK PID
Data OUT PID
Data OUT2
ACK PID
RX_DATA_BK0 (UDP_CSRx)
Interrupt Pending
Set by USB Device
Cleared by Firmware, Data Payload Written in FIFO
FIFO (DPR) Content
Data OUT 1
Data OUT 1
Data OUT 2
Written by USB Device
Microcontroller Read
Written by USB Device

set. Memory transfer between the USB device, the FIFO and microcontroller memory can not be done after RX_DATA_BK0 has been cleared. Otherwise, the USB device would accept the next Data OUT transfer and overwrite the current Data OUT packet in the FIFO.

Using Endpoints With Ping-pong Attributes

During isochronous transfer, using an endpoint with ping-pong attributes is obligatory. To be able to guarantee a constant bandwidth, the microcontroller must read the previous data payload sent by the host, while the current data payload is received by the USB device. Thus two banks of memory are used. While one is available for the microcontroller, the other one is locked by the USB device.

Figure 31-10. Bank Swapping in Data OUT Transfers for Ping-pong EndpointsWhen using a ping-pong endpoint, the fol-

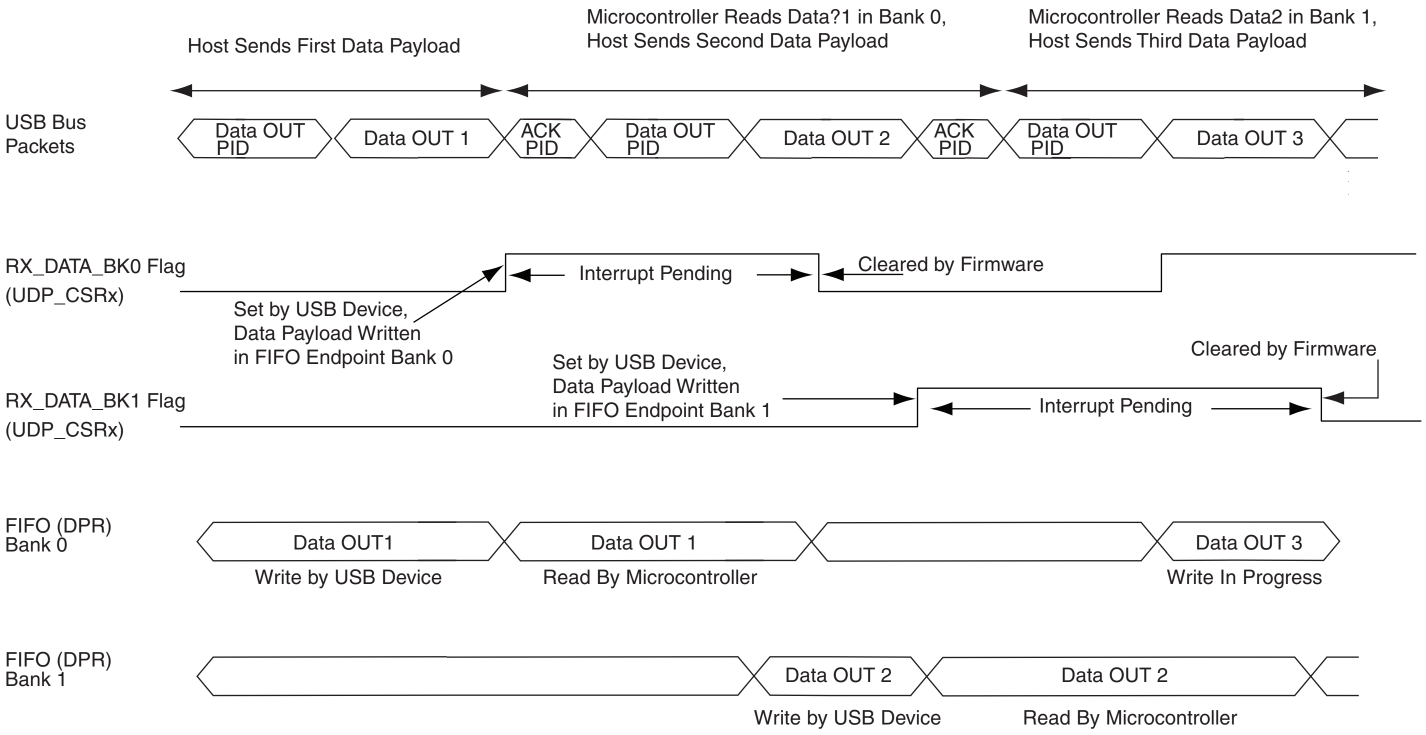

Microcontroller
USB Device
USB Bus
Write
Read
Bank 0
Endpoint 1
Data IN Packet
1st Data Payload
Write and Read at the Same Time
1st Data Payload
Bank 0
Endpoint 1
Bank 1
Endpoint 1
Data IN Packet
2nd Data Payload
2nd Data Payload
Bank 1
Endpoint 1
Bank 0
Endpoint 1
Data IN Packet
3rd Data Payload
3rd Data Payload
Bank 0
Endpoint 1

lowing procedures are required to perform Data OUT transactions:

1. The host generates a Data OUT packet.
2. This packet is received by the USB device endpoint. It is written in the endpoint's FIFO Bank 0.
3. The USB device sends an ACK PID packet to the host. The host can immediately send a second Data OUT packet. It is accepted by the device and copied to FIFO Bank 1.
4. The microcontroller is notified that the USB device has received a data payload, polling RX_DATA_BK0 in the endpoint's UDP_ CSRx register. An interrupt is pending for this endpoint while RX_DATA_BK0 is set.
5. The number of bytes available in the FIFO is made available by reading RXBYTECNT in the endpoint's UDP_ CSRx register.
6. The microcontroller transfers out data received from the endpoint's memory to the microcontroller's memory. Data received is made available by reading the endpoint's UDP_ FDRx register.
7. The microcontroller notifies the USB peripheral device that it has finished the transfer by clearing RX_DATA_BK0 in the endpoint's UDP_ CSRx register.
8. A third Data OUT packet can be accepted by the USB peripheral device and copied in the FIFO Bank 0.
9. If a second Data OUT packet has been received, the microcontroller is notified by the flag RX_DATA_BK1 set in the endpoint's UDP_ CSRx register. An interrupt is pending for this endpoint while RX_DATA_BK1 is set.
10. The microcontroller transfers out data received from the endpoint's memory to the microcontroller's memory. Data received is available by reading the endpoint's UDP_ FDRx register.

11. The microcontroller notifies the USB device it has finished the transfer by clearing RX_DATA_BK1 in the endpoint's UDP_ CSRx register.
12. A fourth Data OUT packet can be accepted by the USB device and copied in the FIFO Bank 0.

Figure 31-11. Data OUT Transfer for Ping-pong EndpointAn interrupt is pending while the RX_DATA_BK0 or


Host Sends First Data Payload
Microcontroller Reads Data?1 in Bank 0,
Host Sends Second Data Payload
Microcontroller Reads Data2 in Bank 1,
Host Sends Third Data Payload
USB Bus
Packets
Data OUT PID
Data OUT 1
ACK PID
Data OUT PID
Data OUT 2
ACK PID
Data OUT PID
Data OUT 3
RX_DATA_BK0 Flag
(UDP_CSRx)
Interrupt Pending
Cleared by Firmware
Set by USB Device,
Data Payload Written
in FIFO Endpoint Bank 0
Cleared by Firmware
Set by USB Device,
Data Payload Written
in FIFO Endpoint Bank 1
RX_DATA_BK1 Flag
(UDP_CSRx)
Interrupt Pending
FIFO (DPR)
Bank 0
Data OUT1
Data OUT 1
Data OUT 3
Write by USB Device
Read By Microcontroller
Write In Progress
FIFO (DPR)
Bank 1
Data OUT 2
Data OUT 2
Write by USB Device
Read By Microcontroller

RX_DATA_BK1 flag is set.

Warning: When RX_DATA_BK0 and RX_DATA_BK1 are both set, there is no way to determine which one to clear first. Thus the software must keep an internal counter to be sure to clear alternatively RX_DATA_BK0 then RX_DATA_BK1. This situation may occur when the software application is busy elsewhere and the two banks are filled by the USB host. Once the application comes back to the USB driver, the two flags are set.

Stall Handshake

A stall handshake can be used in one of two distinct occasions. (For more information on the stall handshake, refer to Chapter 8 of the *Universal Serial Bus Specification, Rev 2.0.*)

- A functional stall is used when the halt feature associated with the endpoint is set. (Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev 2.0,* for more information on the halt feature.)
- To abort the current request, a protocol stall is used, but uniquely with control transfer.

The following procedure generates a stall packet:

1. The microcontroller sets the FORCESTALL flag in the UDP_ CSRx endpoint's register.
2. The host receives the stall packet.

3. The microcontroller is notified that the device has sent the stall by polling the STALLSENT to be set. An endpoint interrupt is pending while STALLSENT is set. The microcontroller must clear STALLSENT to clear the interrupt.

When a setup transaction is received after a stall handshake, STALLSENT must be cleared in order to prevent interrupts due to STALLSENT being set.

Figure 31-12. Stall Handshake (Data IN Transfer)

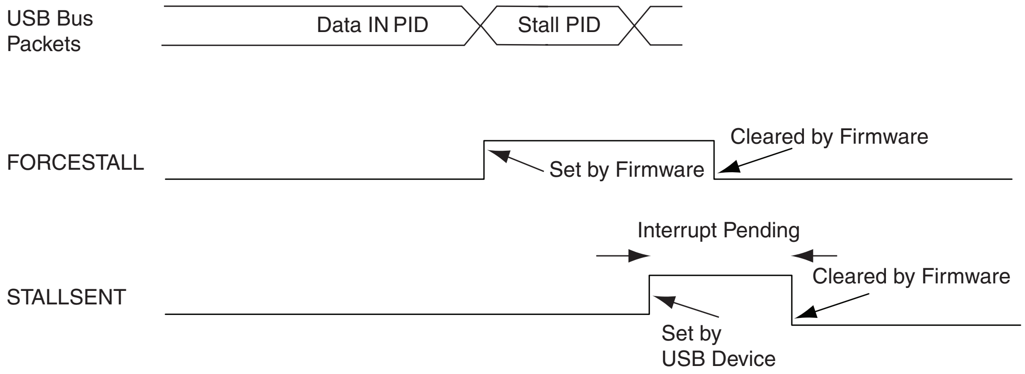
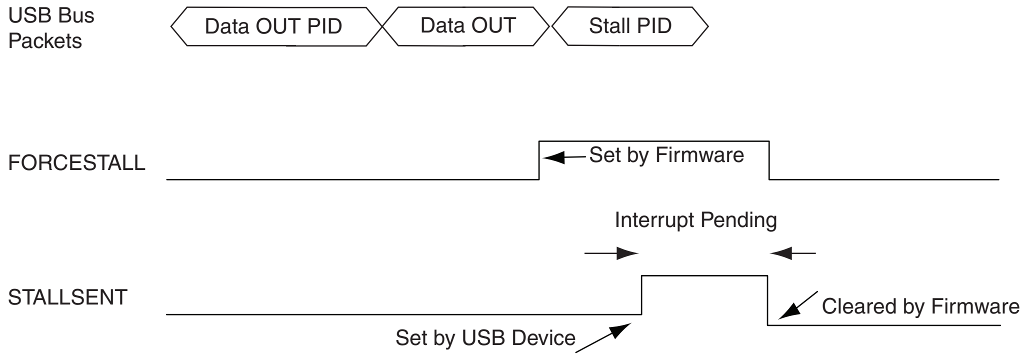

USB Bus Packets
Data IN PID
Stall PID
FORCESTALL
Cleared by Firmware
Set by Firmware
Interrupt Pending
STALLSENT
Cleared by Firmware
Set by USB Device

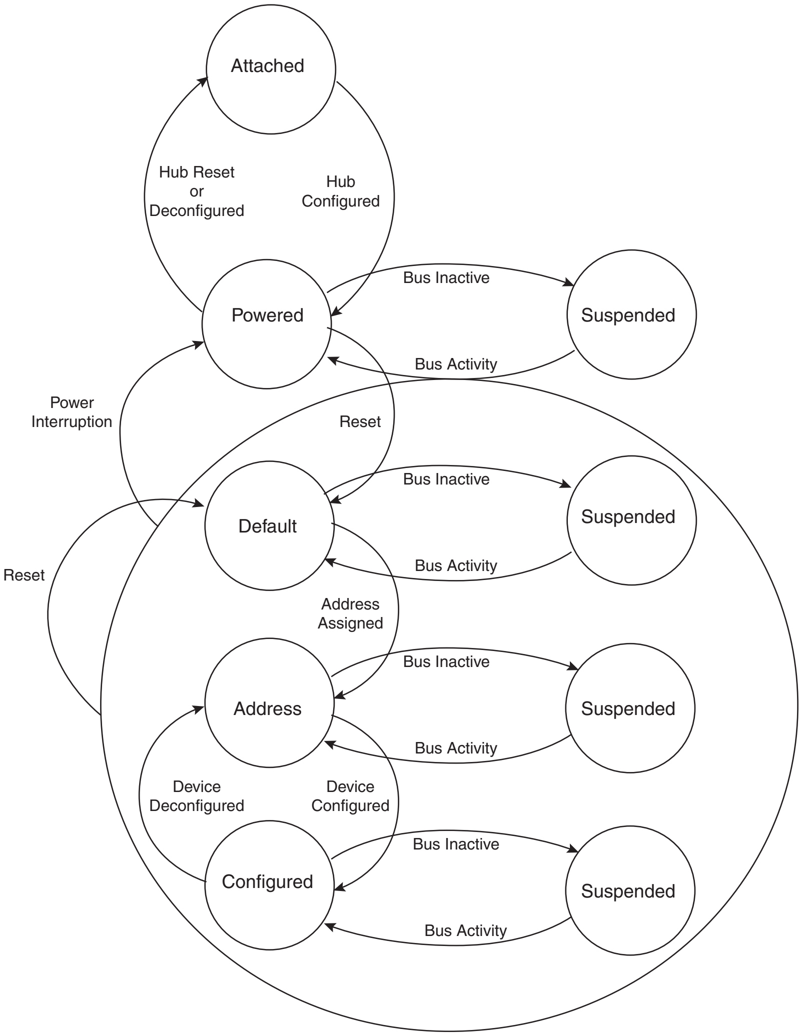
Figure 31-13. Stall Handshake (Data OUT Transfer)


USB Bus Packets
Data OUT PID
Data OUT
Stall PID
FORCESTALL
Set by Firmware
Interrupt Pending
STALLSENT
Cleared by Firmware
Set by USB Device

31.5.3 Controlling Device States

A USB device has several possible states. Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev 2.0*.

Figure 31-14. USB Device State Diagram


Attached
Hub Reset or Deconfigured
Hub Configured
Powered
Bus Inactive
Bus Activity
Suspended
Power Interruption
Reset
Default
Bus Inactive
Bus Activity
Suspended
Reset
Address Assigned
Address
Bus Inactive
Bus Activity
Suspended
Device Deconfigured
Device Configured
Configured
Bus Inactive
Bus Activity
Suspended

Movement from one state to another depends on the USB bus state or on standard requests sent through control transactions via the default endpoint (endpoint 0).

After a period of bus inactivity, the USB device enters Suspend Mode. Accepting Suspend/Resume requests from the USB host is mandatory. Constraints in Suspend Mode are very strict for bus-powered applications; devices may not consume more than 500 μA on the USB bus.

While in Suspend Mode, the host may wake up a device by sending a resume signal (bus activity) or a USB device may send a wake up request to the host, e.g., waking up a PC by moving a USB mouse.

The wake up feature is not mandatory for all devices and must be negotiated with the host.

Not Powered State

Self powered devices can detect 5V VBUS using a PIO as described in the typical connection section. When the device is not connected to a host, device power consumption can be reduced by disabling MCK for the UDP, disabling UDPCK and disabling the transceiver. DDP and DDM lines are pulled down by 330 KΩ resistors.

31.5.3.1 *Entering Attached State*

When no device is connected, the USB DP and DM signals are tied to GND by 15 KΩ pull-down resistors integrated in the hub downstream ports. When a device is attached to a hub downstream port, the device connects a 1.5 KΩ pull-up resistor on DP. The USB bus line goes into IDLE state, DP is pulled up by the device 1.5 KΩ resistor to 3.3V and DM is pulled down by the 15 KΩ resistor of the host. To enable integrated pullup, the UDP_PUP_ON bit in the MATRIX_USBPCR Bus Matrix register must be set.

After pullup connection, the device enters the powered state. In this state, the UDPCK and MCK must be enabled in the Power Management Controller. The transceiver can remain disabled.

31.5.3.2 *From Powered State to Default State*

After its connection to a USB host, the USB device waits for an end-of-bus reset. The unmaskable flag ENDBUSRES is set in the register UDP_ISR and an interrupt is triggered.

Once the ENDBUSRES interrupt has been triggered, the device enters Default State. In this state, the UDP software must:

- Enable the default endpoint, setting the EPEDS flag in the UDP_CSR[0] register and, optionally, enabling the interrupt for endpoint 0 by writing 1 to the UDP_IER register. The enumeration then begins by a control transfer.
- Configure the interrupt mask register which has been reset by the USB reset detection
- Enable the transceiver clearing the TXVDIS flag in the UDP_TXVC register.

In this state UDPCK and MCK must be enabled.

Warning: Each time an ENDBUSRES interrupt is triggered, the Interrupt Mask Register and UDP_CSR registers have been reset.

31.5.3.3 *From Default State to Address State*

After a set address standard device request, the USB host peripheral enters the address state.

Warning: Before the device enters in address state, it must achieve the Status IN transaction of the control transfer, i.e., the UDP device sets its new address once the TXCOMP flag in the UDP_CSR[0] register has been received and cleared.

To move to address state, the driver software sets the FADDEN flag in the UDP_GLB_STAT register, sets its new address, and sets the FEN bit in the UDP_FADDR register.

31.5.3.4 *From Address State to Configured State*

Once a valid Set Configuration standard request has been received and acknowledged, the device enables endpoints corresponding to the current configuration. This is done by setting the EPEDS and EPTYPE fields in the UDP_CSRx registers and, optionally, enabling corresponding interrupts in the UDP_IER register.

31.5.3.5 Entering in Suspend State

When a Suspend (no bus activity on the USB bus) is detected, the RXSUSP signal in the UDP_ISR register is set. This triggers an interrupt if the corresponding bit is set in the UDP_IMR register.This flag is cleared by writing to the UDP_ICR register. Then the device enters Suspend Mode.

In this state bus powered devices must drain less than 500uA from the 5V VBUS. As an example, the microcontroller switches to slow clock, disables the PLL and main oscillator, and goes into Idle Mode. It may also switch off other devices on the board.

The USB device peripheral clocks can be switched off. Resume event is asynchronously detected. MCK and UDPCK can be switched off in the Power Management controller and the USB transceiver can be disabled by setting the TXVDIS field in the UDP_TXVC register.

Warning: Read, write operations to the UDP registers are allowed only if MCK is enabled for the UDP peripheral. Switching off MCK for the UDP peripheral must be one of the last operations after writing to the UDP_TXVC and acknowledging the RXSUSP.

31.5.3.6 Receiving a Host Resume

In suspend mode, a resume event on the USB bus line is detected asynchronously, transceiver and clocks are disabled (however the pullup shall not be removed).

Once the resume is detected on the bus, the WAKEUP signal in the UDP_ISR is set. It may generate an interrupt if the corresponding bit in the UDP_IMR register is set. This interrupt may be used to wake up the core, enable PLL and main oscillators and configure clocks.

Warning: Read, write operations to the UDP registers are allowed only if MCK is enabled for the UDP peripheral. MCK for the UDP must be enabled before clearing the WAKEUP bit in the UDP_ICR register and clearing TXVDIS in the UDP_TXVC register.

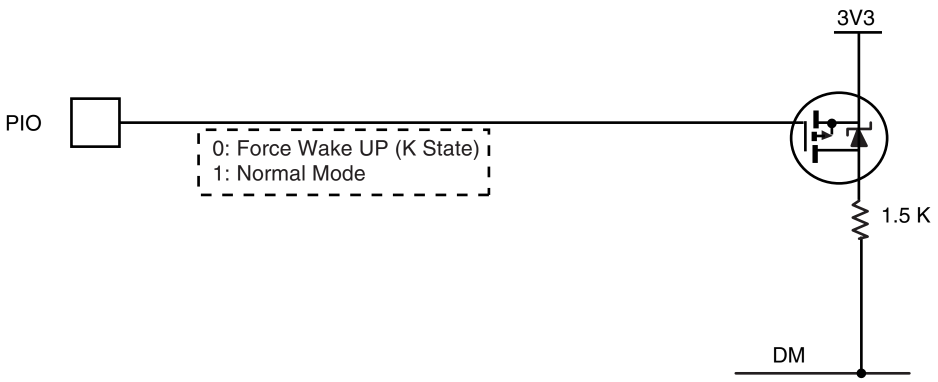
31.5.3.7 Sending a Device Remote Wakeup

In Suspend state it is possible to wake up the host sending an external resume.

- The device must wait at least 5 ms after being entered in suspend before sending an external resume.
- The device has 10 ms from the moment it starts to drain current and it forces a K state to resume the host.
- The device must force a K state from 1 to 15 ms to resume the host

To force a K state to the bus (DM at 3.3V and DP tied to GND), it is possible to use a transistor to connect a pullup on DM. The K state is obtained by disabling the pullup on DP and enabling the pullup on DM. This should be under the control of the application.

Figure 31-15. Board Schematic to Drive a K State


3V3
PIO
0: Force Wake UP (K State)
1: Normal Mode
1.5 K
DM

31.6 USB Device Port (UDP) User Interface

WARNING: The UDP peripheral clock in the Power Management Controller (PMC) must be enabled before any read/write operations to the UDP registers including the UDP_TXVC register.

Table 31-4. UDP Memory Map

Offset	Register	Name	Access	Reset State
0x000	Frame Number Register	UDP_ FRM_NUM	Read	0x0000_0000
0x004	Global State Register	UDP_ GLB_STAT	Read/Write	0x0000_0000
0x008	Function Address Register	UDP_ FADDR	Read/Write	0x0000_0100
0x00C	Reserved	–	–	–
0x010	Interrupt Enable Register	UDP_ IER	Write	
0x014	Interrupt Disable Register	UDP_ IDR	Write	
0x018	Interrupt Mask Register	UDP_ IMR	Read	0x0000_1200
0x01C	Interrupt Status Register	UDP_ ISR	Read	0x0000_XX00
0x020	Interrupt Clear Register	UDP_ ICR	Write	
0x024	Reserved	–	–	–
0x028	Reset Endpoint Register	UDP_ RST_EP	Read/Write	
0x02C	Reserved	–	–	–
0x030	Endpoint 0 Control and Status Register	UDP_CSR0	Read/Write	0x0000_0000
. . .	. . .			
See Note: (1)	Endpoint 5 Control and Status Register	UDP_CSR5	Read/Write	0x0000_0000
0x050	Endpoint 0 FIFO Data Register	UDP_ FDR0	Read/Write	0x0000_0000
. . .	. . .			
See Note: (2)	Endpoint 5 FIFO Data Register	UDP_ FDR5	Read/Write	0x0000_0000
0x070	Reserved	–	–	–
0x074	Transceiver Control Register	UDP_ TXVC (3)	Read/Write	0x0000_0100
0x078 - 0xFC	Reserved	–	–	–

Notes: 1. The addresses of the UDP_ CSRx registers are calculated as: 0x030 + 4(Endpoint Number - 1).
2. The addresses of the UDP_ FDRx registers are calculated as: 0x050 + 4(Endpoint Number - 1).
3. See Warning above the "UDP Memory Map" on this page.

31.6.1 UDP Frame Number Register

Register Name: UDP_ FRM_NUM

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	FRM_NUM		

7	6	5	4	3	2	1	0
FRM_NUM							

- **FRM_NUM[10:0]: Frame Number as Defined in the Packet Field Formats**

This 11-bit value is incremented by the host on a per frame basis. This value is updated at each start of frame.

Value Updated at the SOF_EOP (Start of Frame End of Packet).

- **FRM_ERR: Frame Error**

This bit is set at SOF_EOP when the SOF packet is received containing an error.

This bit is reset upon receipt of SOF_PID.

- **FRM_OK: Frame OK**

This bit is set at SOF_EOP when the SOF packet is received without any error.

This bit is reset upon receipt of SOF_PID (Packet Identification).

In the Interrupt Status Register, the SOF interrupt is updated upon receiving SOF_PID. This bit is set without waiting for EOP.

Note: In the 8-bit Register Interface, FRM_OK is bit 4 of FRM_NUM_H and FRM_ERR is bit 3 of FRM_NUM_L.

31.6.2 UDP Global State Register

Register Name: UDP_GLB_STAT

Access Type: Read/Write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	CONFG	FADDEN

This register is used to get and set the device state as specified in Chapter 9 of the *USB Serial Bus Specification, Rev.2.0*.

- **FADDEN: Function Address Enable**

Read:

0 = Device is not in address state.

1 = Device is in address state.

Write:

0 = No effect, only a reset can bring back a device to the default state.

1 = Sets device in address state. This occurs after a successful Set Address request. Beforehand, the UDP_ FADDR register must have been initialized with Set Address parameters. Set Address must complete the Status Stage before setting FADDEN. Refer to chapter 9 of the *Universal Serial Bus Specification, Rev. 2.0* for more details.

- **CONFG: Configured**

Read:

0 = Device is not in configured state.

1 = Device is in configured state.

Write:

0 = Sets device in a non configured state

1 = Sets device in configured state.

The device is set in configured state when it is in address state and receives a successful Set Configuration request. Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev. 2.0* for more details.

31.6.3 UDP Function Address Register

Register Name: UDP_ FADDR

Access Type: Read/Write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	FEN

7	6	5	4	3	2	1	0
–	FADD						

- **FADD[6:0]: Function Address Value**

The Function Address Value must be programmed by firmware once the device receives a set address request from the host, and has achieved the status stage of the no-data control sequence. Refer to the *Universal Serial Bus Specification, Rev. 2.0* for more information. After power up or reset, the function address value is set to 0.

- **FEN: Function Enable**

Read:

0 = Function endpoint disabled.

1 = Function endpoint enabled.

Write:

0 = Disables function endpoint.

1 = Default value.

The Function Enable bit (FEN) allows the microcontroller to enable or disable the function endpoints. The microcontroller sets this bit after receipt of a reset from the host. Once this bit is set, the USB device is able to accept and transfer data packets from and to the host.

31.6.4 UDP Interrupt Enable Register

Register Name: UDP_ IER

Access Type: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	WAKEUP	–	SOFINT	–	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
		EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: Enable Endpoint 0 Interrupt**

- **EP1INT: Enable Endpoint 1 Interrupt**

- **EP2INT: Enable Endpoint 2Interrupt**

- **EP3INT: Enable Endpoint 3 Interrupt**

- **EP4INT: Enable Endpoint 4 Interrupt**

- **EP5INT: Enable Endpoint 5 Interrupt**

0 = No effect.

1 = Enables corresponding Endpoint Interrupt.

- **RXSUSP: Enable UDP Suspend Interrupt**

0 = No effect.

1 = Enables UDP Suspend Interrupt.

- **RXRSM: Enable UDP Resume Interrupt**

0 = No effect.

1 = Enables UDP Resume Interrupt.

- **SOFINT: Enable Start Of Frame Interrupt**

0 = No effect.

1 = Enables Start Of Frame Interrupt.

- **WAKEUP: Enable UDP bus Wakeup Interrupt**

0 = No effect.

1 = Enables USB bus Interrupt.

31.6.5 UDP Interrupt Disable Register

Register Name: UDP_ IDR

Access Type: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	WAKEUP	–	SOFINT	–	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
		EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: Disable Endpoint 0 Interrupt**

- **EP1INT: Disable Endpoint 1 Interrupt**

- **EP2INT: Disable Endpoint 2 Interrupt**

- **EP3INT: Disable Endpoint 3 Interrupt**

- **EP4INT: Disable Endpoint 4 Interrupt**

- **EP5INT: Disable Endpoint 5 Interrupt**

0 = No effect.

1 = Disables corresponding Endpoint Interrupt.

- **RXSUSP: Disable UDP Suspend Interrupt**

0 = No effect.

1 = Disables UDP Suspend Interrupt.

- **RXRSM: Disable UDP Resume Interrupt**

0 = No effect.

1 = Disables UDP Resume Interrupt.

- **SOFINT: Disable Start Of Frame Interrupt**

0 = No effect.

1 = Disables Start Of Frame Interrupt

- **WAKEUP: Disable USB Bus Interrupt**

0 = No effect.

1 = Disables USB Bus Wakeup Interrupt.

31.6.6 UDP Interrupt Mask Register

Register Name: UDP_ 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(1)	11	10	9	8
–	–	WAKEUP	–	SOFINT	–	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
		EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

Note: 1. Bit 12 of UDP_IMR cannot be masked and is always read at 1.

- **EP0INT: Mask Endpoint 0 Interrupt**
- **EP1INT: Mask Endpoint 1 Interrupt**
- **EP2INT: Mask Endpoint 2 Interrupt**
- **EP3INT: Mask Endpoint 3 Interrupt**
- **EP4INT: Mask Endpoint 4 Interrupt**
- **EP5INT: Mask Endpoint 5 Interrupt**

0 = Corresponding Endpoint Interrupt is disabled.

1 = Corresponding Endpoint Interrupt is enabled.

- **RXSUSP: Mask UDP Suspend Interrupt**

0 = UDP Suspend Interrupt is disabled.

1 = UDP Suspend Interrupt is enabled.

- **RXRSM: Mask UDP Resume Interrupt.**

0 = UDP Resume Interrupt is disabled.

1 = UDP Resume Interrupt is enabled.

- **SOFINT: Mask Start Of Frame Interrupt**

0 = Start of Frame Interrupt is disabled.

1 = Start of Frame Interrupt is enabled.

- **WAKEUP: USB Bus WAKEUP Interrupt**

0 = USB Bus Wakeup Interrupt is disabled.

1 = USB Bus Wakeup Interrupt is enabled.

Note: When the USB block is in suspend mode, the application may power down the USB logic. In this case, any USB HOST resume request that is made must be taken into account and, thus, the reset value of the RXRSM bit of the register UDP_ IMR is enabled.

31.6.7 UDP Interrupt Status Register

Register Name: UDP_ ISR

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	WAKEUP	ENDBUSRES	SOFINT	–	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
		EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: Endpoint 0 Interrupt Status**

- **EP1INT: Endpoint 1 Interrupt Status**

- **EP2INT: Endpoint 2 Interrupt Status**

- **EP3INT: Endpoint 3 Interrupt Status**

- **EP4INT: Endpoint 4 Interrupt Status**

- **EP5INT: Endpoint 5 Interrupt Status**

0 = No Endpoint0 Interrupt pending.

1 = Endpoint0 Interrupt has been raised.

Several signals can generate this interrupt. The reason can be found by reading UDP_ CSR0:

RXSETUP set to 1

RX_DATA_BK0 set to 1

RX_DATA_BK1 set to 1

TXCOMP set to 1

STALLSENT set to 1

EP0INT is a sticky bit. Interrupt remains valid until EP0INT is cleared by writing in the corresponding UDP_ CSR0 bit.

- **RXSUSP: UDP Suspend Interrupt Status**

0 = No UDP Suspend Interrupt pending.

1 = UDP Suspend Interrupt has been raised.

The USB device sets this bit when it detects no activity for 3ms. The USB device enters Suspend mode.

- **RXRSM: UDP Resume Interrupt Status**

0 = No UDP Resume Interrupt pending.

1 =UDP Resume Interrupt has been raised.

The USB device sets this bit when a UDP resume signal is detected at its port.

After reset, the state of this bit is undefined, the application must clear this bit by setting the RXRSM flag in the UDP_ ICR register.

- **SOFINT: Start of Frame Interrupt Status**

0 = No Start of Frame Interrupt pending.

1 = Start of Frame Interrupt has been raised.

This interrupt is raised each time a SOF token has been detected. It can be used as a synchronization signal by using isochronous endpoints.

- **ENDBUSRES: End of BUS Reset Interrupt Status**

0 = No End of Bus Reset Interrupt pending.

1 = End of Bus Reset Interrupt has been raised.

This interrupt is raised at the end of a UDP reset sequence. The USB device must prepare to receive requests on the endpoint 0. The host starts the enumeration, then performs the configuration.

- **WAKEUP: UDP Resume Interrupt Status**

0 = No Wakeup Interrupt pending.

1 = A Wakeup Interrupt (USB Host Sent a RESUME or RESET) occurred since the last clear.

After reset the state of this bit is undefined, the application must clear this bit by setting the WAKEUP flag in the UDP_ ICR register.

31.6.8 UDP Interrupt Clear Register

Register Name: UDP_ ICR

Access Type: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	WAKEUP	ENDBUSRES	SOFINT	–	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **RXSUSP: Clear UDP Suspend Interrupt**

0 = No effect.

1 = Clears UDP Suspend Interrupt.

- **RXRSM: Clear UDP Resume Interrupt**

0 = No effect.

1 = Clears UDP Resume Interrupt.

- **SOFINT: Clear Start Of Frame Interrupt**

0 = No effect.

1 = Clears Start Of Frame Interrupt.

- **ENDBUSRES: Clear End of Bus Reset Interrupt**

0 = No effect.

1 = Clears End of Bus Reset Interrupt.

- **WAKEUP: Clear Wakeup Interrupt**

0 = No effect.

1 = Clears Wakeup Interrupt.

31.6.9 UDP Reset Endpoint Register

Register Name: UDP_ RST_EP

Access Type: Read/Write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
		EP5	EP4	EP3	EP2	EP1	EP0

- **EP0: Reset Endpoint 0**
- **EP1: Reset Endpoint 1**
- **EP2: Reset Endpoint 2**
- **EP3: Reset Endpoint 3**
- **EP4: Reset Endpoint 4**
- **EP5: Reset Endpoint 5**

This flag is used to reset the FIFO associated with the endpoint and the bit RXBYTECOUNT in the register UDP_CSRx.It also resets the data toggle to DATA0. It is useful after removing a HALT condition on a BULK endpoint. Refer to Chapter 5.8.5 in the *USB Serial Bus Specification, Rev.2.0*.

Warning: This flag must be cleared at the end of the reset. It does not clear UDP_ CSRx flags.

0 = No reset.

1 = Forces the corresponding endpoint FIF0 pointers to 0, therefore RXBYTECNT field is read at 0 in UDP_ CSRx register.

31.6.10 UDP Endpoint Control and Status Register

Register Name: UDP_ CSRx [x = 0..5]

Access Type: Read/Write

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

23	22	21	20	19	18	17	16
RXBYTECNT							

15	14	13	12	11	10	9	8
EPEDS	–	–	–	DTGLE	EPTYPE		

7	6	5	4	3	2	1	0
DIR	RX_DATA_ BK1	FORCE STALL	TXPKTRDY	STALLSENT ISOERROR	RXSETUP	RX_DATA_ BK0	TXCOMP

WARNING: Due to synchronization between MCK and UDPCK, the software application must wait for the end of the write operation before executing another write by polling the bits which must be set/cleared.

```
//! Clear flags of UDP UDP_CSR register and waits for synchronization
#define Udp_ep_clr_flag(pInterface, endpoint, flags) { \
  while (pInterface->UDP_CSR[endpoint] & (flags)) \
    pInterface->UDP_CSR[endpoint] &= ~(flags); \
  }
//! Set flags of UDP UDP_CSR register and waits for synchronization
#define Udp_ep_set_flag(pInterface, endpoint, flags) { \
  while ( (pInterface->UDP_CSR[endpoint] & (flags)) != (flags) ) \
    pInterface->UDP_CSR[endpoint] |= (flags); \
  }
```

- **TXCOMP: Generates an IN Packet with Data Previously Written in the DPR**

This flag generates an interrupt while it is set to one.

Write (Cleared by the firmware):

0 = Clear the flag, clear the interrupt.

1 = No effect.

Read (Set by the USB peripheral):

0 = Data IN transaction has not been acknowledged by the Host.

1 = Data IN transaction is achieved, acknowledged by the Host.

After having issued a Data IN transaction setting TXPKTRDY, the device firmware waits for TXCOMP to be sure that the host has acknowledged the transaction.

- **RX_DATA_BK0: Receive Data Bank 0**

This flag generates an interrupt while it is set to one.
Write (Cleared by the firmware):

0 = Notify USB peripheral device that data have been read in the FIFO's Bank 0.

1 = To leave the read value unchanged.

Read (Set by the USB peripheral):

0 = No data packet has been received in the FIFO's Bank 0.

1 = A data packet has been received, it has been stored in the FIFO's Bank 0.

When the device firmware has polled this bit or has been interrupted by this signal, it must transfer data from the FIFO to the microcontroller memory. The number of bytes received is available in RXBYTCENT field. Bank 0 FIFO values are read through the UDP_ FDRx register. Once a transfer is done, the device firmware must release Bank 0 to the USB peripheral device by clearing RX_DATA_BK0.

- **RXSETUP: Received Setup**

This flag generates an interrupt while it is set to one.

Read:

0 = No setup packet available.

1 = A setup data packet has been sent by the host and is available in the FIFO.

Write:

0 = Device firmware notifies the USB peripheral device that it has read the setup data in the FIFO.

1 = No effect.

This flag is used to notify the USB device firmware that a valid Setup data packet has been sent by the host and successfully received by the USB device. The USB device firmware may transfer Setup data from the FIFO by reading the UDP_ FDRx register to the microcontroller memory. Once a transfer has been done, RXSETUP must be cleared by the device firmware.

Ensuing Data OUT transaction is not accepted while RXSETUP is set.

- **STALLSENT: Stall Sent (Control, Bulk Interrupt Endpoints)/ISOERROR (Isochronous Endpoints)**

This flag generates an interrupt while it is set to one.

STALLSENT: This ends a STALL handshake.

Read:

0 = The host has not acknowledged a STALL.

1 = Host has acknowledged the stall.

Write:

0 = Resets the STALLSENT flag, clears the interrupt.

1 = No effect.

This is mandatory for the device firmware to clear this flag. Otherwise the interrupt remains.

Refer to chapters 8.4.5 and 9.4.5 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the STALL handshake.

ISOERROR: A CRC error has been detected in an isochronous transfer.

Read:

0 = No error in the previous isochronous transfer.

1 = CRC error has been detected, data available in the FIFO are corrupted.

Write:

0 = Resets the ISOERROR flag, clears the interrupt.

1 = No effect.

- **TXPKTRDY: Transmit Packet Ready**

This flag is cleared by the USB device.

This flag is set by the USB device firmware.

Read:

0 = Can be set to one to send the FIFO data.

1 = The data is waiting to be sent upon reception of token IN.

Write:

0 = Can be written if old value is zero.

1 = A new data payload is has been written in the FIFO by the firmware and is ready to be sent.

This flag is used to generate a Data IN transaction (device to host). Device firmware checks that it can write a data payload in the FIFO, checking that TXPKTRDY is cleared. Transfer to the FIFO is done by writing in the UDP_ FDRx register. Once the data payload has been transferred to the FIFO, the firmware notifies the USB device setting TXPKTRDY to one. USB bus transactions can start. TXCOMP is set once the data payload has been received by the host.

- **FORCESTALL: Force Stall (used by Control, Bulk and Isochronous Endpoints)**

Read:

0 = Normal state.

1 = Stall state.

Write:

0 = Return to normal state.

1 = Send STALL to the host.

Refer to chapters 8.4.5 and 9.4.5 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the STALL handshake.

Control endpoints: During the data stage and status stage, this bit indicates that the microcontroller cannot complete the request.

Bulk and interrupt endpoints: This bit notifies the host that the endpoint is halted.

The host acknowledges the STALL, device firmware is notified by the STALLSENT flag.

- **RX_DATA_BK1: Receive Data Bank 1 (only used by endpoints with ping-pong attributes)**

This flag generates an interrupt while it is set to one.

Write (Cleared by the firmware):

0 = Notifies USB device that data have been read in the FIFO’s Bank 1.

1 = To leave the read value unchanged.

Read (Set by the USB peripheral):

0 = No data packet has been received in the FIFO's Bank 1.

1 = A data packet has been received, it has been stored in FIFO's Bank 1.

When the device firmware has polled this bit or has been interrupted by this signal, it must transfer data from the FIFO to microcontroller memory. The number of bytes received is available in RXBYTECNT field. Bank 1 FIFO values are read through UDP_ FDRx register. Once a transfer is done, the device firmware must release Bank 1 to the USB device by clearing RX_DATA_BK1.

- **DIR: Transfer Direction (only available for control endpoints)**

Read/Write

0 = Allows Data OUT transactions in the control data stage.

1 = Enables Data IN transactions in the control data stage.

Refer to Chapter 8.5.3 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the control data stage.

This bit must be set before UDP_ CSRx/RXSETUP is cleared at the end of the setup stage. According to the request sent in the setup data packet, the data stage is either a device to host (DIR = 1) or host to device (DIR = 0) data transfer. It is not necessary to check this bit to reverse direction for the status stage.

- **EPTYPE[2:0]: Endpoint Type**

Read/Write

000	Control
001	Isochronous OUT
101	Isochronous IN
010	Bulk OUT
110	Bulk IN
011	Interrupt OUT
111	Interrupt IN

- **DTGLE: Data Toggle**

Read-only

0 = Identifies DATA0 packet.

1 = Identifies DATA1 packet.

Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on DATA0, DATA1 packet definitions.

- **EPEDS: Endpoint Enable Disable**

Read:

0 = Endpoint disabled.

1 = Endpoint enabled.

Write:

0 = Disables endpoint.

1 = Enables endpoint.

Control endpoints are always enabled. Reading or writing this field has no effect on control endpoints.

Note: After reset all endpoints are configured as control endpoints (zero).

- **RXBYTECNT[10:0]: Number of Bytes Available in the FIFO**

Read-only

When the host sends a data packet to the device, the USB device stores the data in the FIFO and notifies the microcontroller. The microcontroller can load the data from the FIFO by reading RXBYTECENT bytes in the UDP_ FDRx register.

31.6.11 UDP FIFO Data Register

Register Name: UDP_ FDRx [x = 0..5]

Access Type: Read/Write

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

- **FIFO_DATA[7:0]: FIFO Data Value**

The microcontroller can push or pop values in the FIFO through this register.

RXBYTECNT in the corresponding UDP_ CSRx register is the number of bytes to be read from the FIFO (sent by the host).

The maximum number of bytes to write is fixed by the Max Packet Size in the Standard Endpoint Descriptor. It can not be more than the physical memory size associated to the endpoint. Refer to the *Universal Serial Bus Specification, Rev. 2.0* for more information.

31.6.12 UDP Transceiver Control Register

Register Name: UDP_ TXVC

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
–	–	–	–	–	–	–	TXVDIS

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

WARNING: The UDP peripheral clock in the Power Management Controller (PMC) must be enabled before any read/write operations to the UDP registers including the UDP_TXVC register.

- **TXVDIS: Transceiver Disable**

When UDP is disabled, power consumption can be reduced significantly by disabling the embedded transceiver. This can be done by setting TXVDIS field.

To enable the transceiver, TXVDIS must be cleared. TXVDIS is automatically set after a reset, so it must be cleared again to reenable the transceiver.

Note: The USB transceiver pull-ups are enabled/disabled by writing to the MATRIX_USBPCR register documented in Section 19.6.7.

Note: If the USB pullup is not enabled on DP, the user should not write in any UDP register other than the UDP_ TXVC register. This is because if DP and DM are floating at 0, or pulled down, then SE0 is received by the device with the consequence of a USB Reset.

32. Analog-to-digital Converter (ADC)

32.1 Description

The ADC is based on a Successive Approximation Register (SAR) 10-bit Analog-to-Digital Converter (ADC). It also integrates an 8-to-1 analog multiplexer, making possible the analog-to-digital conversions of 8 analog lines. The conversions extend from 0V to ADVREF.

On the AT91CAP7E device, the analog inputs are AD0 - AD7.

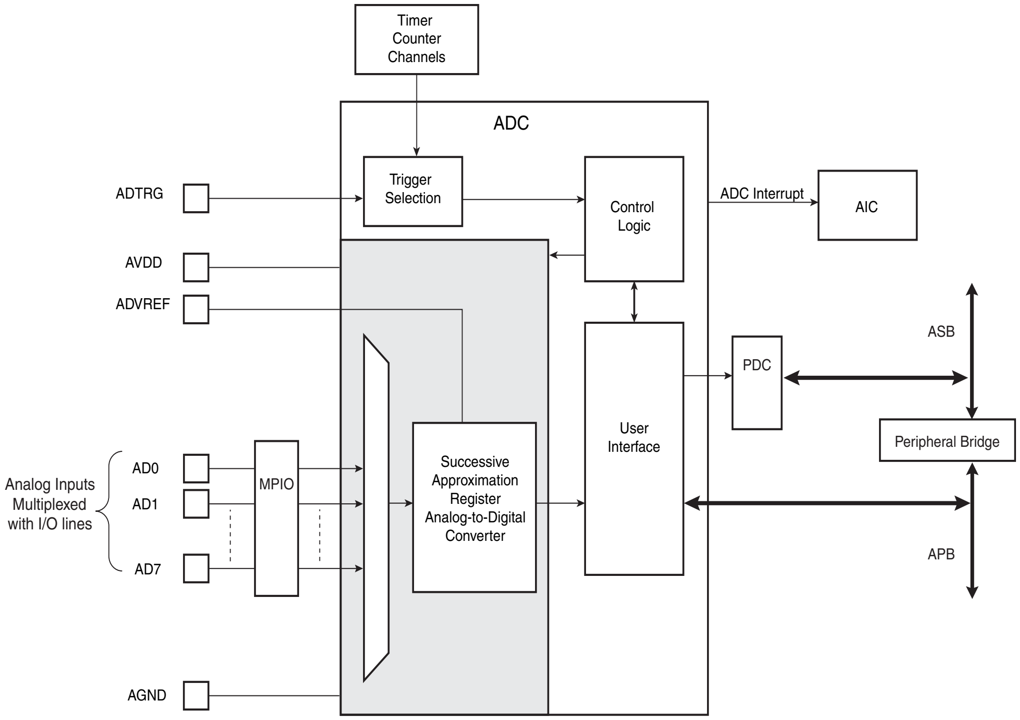
The ADC supports an 8-bit or 10-bit resolution mode, and conversion results are reported in a common register for all channels, as well as in a channel-dedicated register. Software trigger, external trigger on rising edge of the ADTRG pin or internal triggers from Timer Counter output(s) are configurable.

The ADC also integrates a Sleep Mode and a conversion sequencer and connects with a PDC channel. These features reduce both power consumption and processor intervention.

Finally, the user can configure ADC timings, such as Startup Time and Sample & Hold Time.

32.2 Block Diagram

Figure 32-1. Analog-to-Digital Converter Block Diagram


Timer Counter Channels
ADC
Trigger Selection
Control Logic
ADC Interrupt
AIC
ADTRG
AVDD
ADVREF
ASB
PDC
Peripheral Bridge
User Interface
Successive Approximation Register Analog-to-Digital Converter
MPIO
Analog Inputs Multiplexed with I/O lines
AD0
AD1
AD7
APB
AGND

32.3 Signal Description

Table 32-1. ADC Pin Description

Pin Name	Description
AVDD	Analog power supply
ADVREF	Reference voltage
AD0 - AD7	Analog input channels
ADTRG	External trigger

32.4 Product Dependencies

32.4.1 Power Management

The ADC is automatically clocked after the first conversion in Normal Mode. In Sleep Mode, the ADC clock is automatically stopped after each conversion. As the logic is small and the ADC cell can be put into Sleep Mode, the Power Management Controller has no effect on the ADC behavior.

32.4.2 Interrupt Sources

The ADC interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the ADC interrupt requires the AIC to be programmed first.

32.4.3 Analog Inputs

The analog input pins can be multiplexed with PIO lines. In this case, the assignment of the ADC input is automatically done as soon as the corresponding channel is enabled by writing the register ADC_CHER. By default, after reset, the PIO line is configured as input with its pull-up enabled and the ADC input is connected to the GND.

32.4.4 I/O Lines

The pin ADTRG may be shared with other peripheral functions through the PIO Controller. In this case, the PIO Controller should be set accordingly to assign the pin ADTRG to the ADC function.

32.4.5 Timer Triggers

Timer Counters may or may not be used as hardware triggers depending on user requirements. Thus, some or all of the timer counters may be non-connected.

32.4.6 Conversion Performances

For performance and electrical characteristics of the ADC, see the DC Characteristics section.

32.5 Functional Description

32.5.1 Analog-to-digital Conversion

The ADC uses the ADC Clock to perform conversions. Converting a single analog value to a 10-bit digital data requires Sample and Hold Clock cycles as defined in the field SHTIM of the "ADC Mode Register" on page 474 and 10 ADC Clock cycles. The ADC Clock frequency is selected in the PRESCAL field of the Mode Register (ADC_MR).

The ADC clock range is between MCK/2, if PRESCAL is 0, and MCK/128, if PRESCAL is set to 63 (0x3F). PRESCAL must be programmed in order to provide an ADC clock frequency according to the parameters given in the Product definition section.

32.5.2 Conversion Reference

The conversion is performed on a full range between 0V and the reference voltage pin ADVREF. Analog inputs between these voltages convert to values based on a linear conversion.

32.5.3 Conversion Resolution

The ADC supports 8-bit or 10-bit resolutions. The 8-bit selection is performed by setting the bit LOWRES in the ADC Mode Register (ADC_MR). By default, after a reset, the resolution is the highest and the DATA field in the data registers is fully used. By setting the bit LOWRES, the ADC switches in the lowest resolution and the conversion results can be read in the eight lowest significant bits of the data registers. The two highest bits of the DATA field in the corresponding ADC_CDR register and of the LDATA field in the ADC_LCDR register read 0.

Moreover, when a PDC channel is connected to the ADC, 10-bit resolution sets the transfer request sizes to 16-bit. Setting the bit LOWRES automatically switches to 8-bit data transfers. In this case, the destination buffers are optimized.

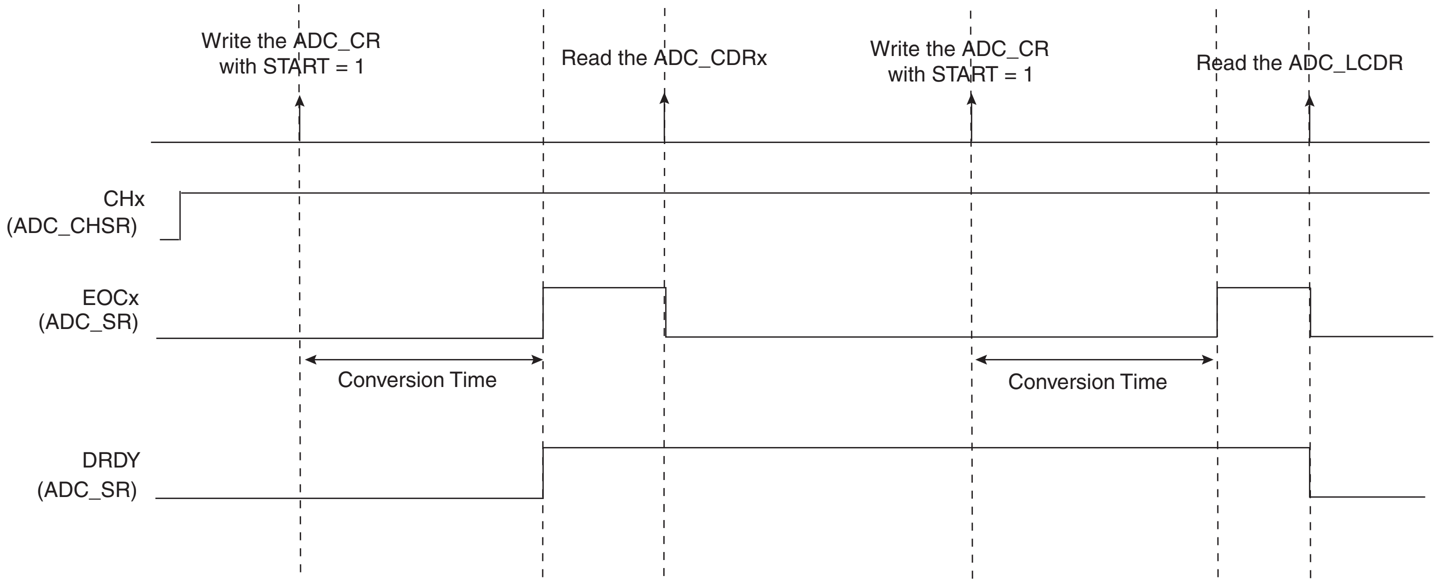
32.5.4 Conversion Results

When a conversion is completed, the resulting 10-bit digital value is stored in the Channel Data Register (ADC_CDR) of the current channel and in the ADC Last Converted Data Register (ADC_LCDR).

The channel EOC bit in the Status Register (ADC_SR) is set and the DRDY is set. In the case of a connected PDC channel, DRDY rising triggers a data transfer request. In any case, either EOC and DRDY can trigger an interrupt.

Reading one of the ADC_CDR registers clears the corresponding EOC bit. Reading ADC_LCDR clears the DRDY bit and the EOC bit corresponding to the last converted channel.

Figure 32-2. EOCx and DRDY Flag Behavior

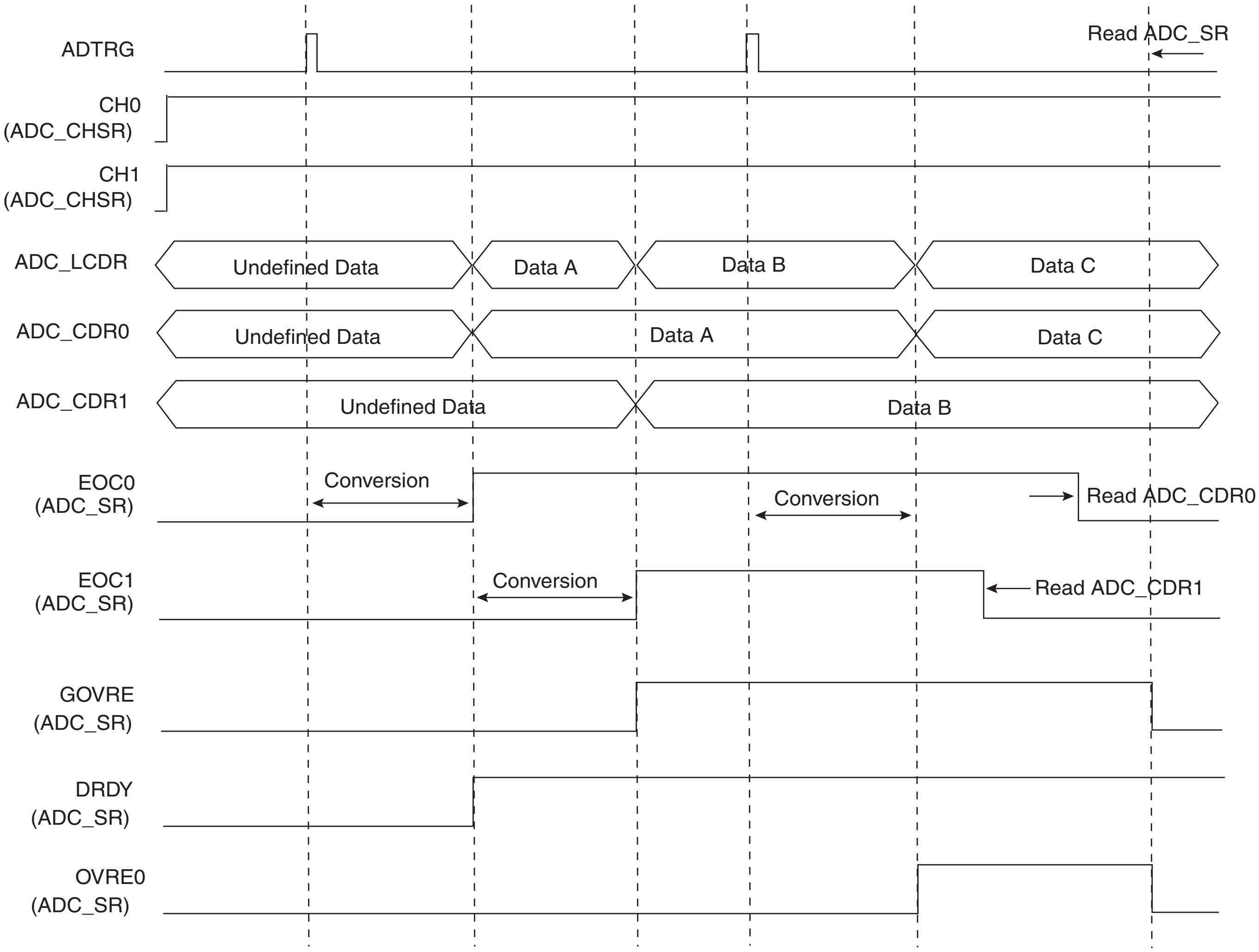

Write the ADC_CR
with START = 1
Read the ADC_CDRx
Write the ADC_CR
with START = 1
Read the ADC_LCDR
CHx
(ADC_CHSR)
EOCx
(ADC_SR)
Conversion Time
Conversion Time
DRDY
(ADC_SR)

If the ADC_CDR is not read before further incoming data is converted, the corresponding Overrun Error (OVRE) flag is set in the Status Register (ADC_SR).

In the same way, new data converted when DRDY is high sets the bit GOVRE (General Overrun Error) in ADC_SR.

The OVRE and GOVRE flags are automatically cleared when ADC_SR is read.

Figure 32-3. GOVRE and OVREx Flag Behavior



ADTRG
Read ADC_SR
CH0 (ADC_CHSR)
CH1 (ADC_CHSR)
ADC_LCDR
Undefined Data
Data A
Data B
Data C
ADC_CDR0
Undefined Data
Data A
Data C
ADC_CDR1
Undefined Data
Data B
EOC0 (ADC_SR)
Conversion
Conversion
Read ADC_CDR0
EOC1 (ADC_SR)
Conversion
Read ADC_CDR1
GOVRE (ADC_SR)
DRDY (ADC_SR)
OVRE0 (ADC_SR)

Warning: If the corresponding channel is disabled during a conversion or if it is disabled and then reenabled during a conversion, its associated data and its corresponding EOC and OVRE flags in ADC_SR are unpredictable.

32.5.5 Conversion Triggers

Conversions of the active analog channels are started with a software or a hardware trigger. The software trigger is provided by writing the Control Register (ADC_CR) with the bit START at 1.

The hardware trigger can be one of the TIOA outputs of the Timer Counter channels, or the external trigger input of the ADC (ADTRG). The hardware trigger is selected with the field TRGSEL in the Mode Register (ADC_MR). The selected hardware trigger is enabled with the bit TRGEN in the Mode Register (ADC_MR).

If a hardware trigger is selected, the start of a conversion is detected at each rising edge of the selected signal. If one of the TIOA outputs is selected, the corresponding Timer Counter channel must be programmed in Waveform Mode.

Only one start command is necessary to initiate a conversion sequence on all the channels. The ADC hardware logic automatically performs the conversions on the active channels, then waits for a new request. The Channel Enable (ADC_CHER) and Channel Disable (ADC_CHDR) Registers enable the analog channels to be enabled or disabled independently.

If the ADC is used with a PDC, only the transfers of converted data from enabled channels are performed and the resulting data buffers should be interpreted accordingly.

Warning: Enabling hardware triggers does not disable the software trigger functionality. Thus, if a hardware trigger is selected, the start of a conversion can be initiated either by the hardware or the software trigger.

32.5.6 Sleep Mode and Conversion Sequencer

The ADC Sleep Mode maximizes power saving by automatically deactivating the ADC when it is not being used for conversions. Sleep Mode is selected by setting the bit SLEEP in the Mode Register ADC_MR.

The SLEEP mode is automatically managed by a conversion sequencer, which can automatically process the conversions of all channels at lowest power consumption.

When a start conversion request occurs, the ADC is automatically activated. As the analog cell requires a start-up time, the logic waits during this time and starts the conversion on the enabled channels. When all conversions are complete, the ADC is deactivated until the next trigger. Triggers occurring during the sequence are not taken into account.

The conversion sequencer allows automatic processing with minimum processor intervention and optimized power consumption. Conversion sequences can be performed periodically using a Timer/Counter output. The periodic acquisition of several samples can be processed automatically without any intervention of the processor thanks to the PDC.

Note: The reference voltage pins always remain connected in normal mode as in sleep mode.

32.5.7 ADC Timings

Each ADC has its own minimal Startup Time that is programmed through the field STARTUP in the Mode Register ADC_MR.

In the same way, a minimal Sample and Hold Time is necessary for the ADC to guarantee the best converted final value between two channels selection. This time has to be programmed through the bitfield SHTIM in the Mode Register ADC_MR.

Warning: No input buffer amplifier to isolate the source is included in the ADC. This must be taken into consideration to program a precise value in the SHTIM field. See the section, ADC Characteristics in the product datasheet.

32.6 Analog-to-digital Converter (ADC) User Interface

Table 32-2. ADC Register Mapping

Offset	Register	Name	Access	Reset State
0x00	Control Register	ADC_CR	Write-only	–
0x04	Mode Register	ADC_MR	Read/Write	0x00000000
0x08	Reserved	–	–	–
0x0C	Reserved	–	–	–
0x10	Channel Enable Register	ADC_CHER	Write-only	–
0x14	Channel Disable Register	ADC_CHDR	Write-only	–
0x18	Channel Status Register	ADC_CHSR	Read-only	0x00000000
0x1C	Status Register	ADC_SR	Read-only	0x000C0000
0x20	Last Converted Data Register	ADC_LCDR	Read-only	0x00000000
0x24	Interrupt Enable Register	ADC_IER	Write-only	–
0x28	Interrupt Disable Register	ADC_IDR	Write-only	–
0x2C	Interrupt Mask Register	ADC_IMR	Read-only	0x00000000
0x30	Channel Data Register 0	ADC_CDR0	Read-only	0x00000000
0x34	Channel Data Register 1	ADC_CDR1	Read-only	0x00000000
...	...	...	...	...
0x4C	Channel Data Register 7	ADC_CDR7	Read-only	0x00000000
0x50 - 0xFC	Reserved	–	–	–
0x100 - 0x124	Reserved for the PDC			

32.6.1 ADC Control Register

Register Name: ADC_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
–	–	–	–	–	–	START	SWRST

- **SWRST: Software Reset**

0 = No effect.

1 = Resets the ADC simulating a hardware reset.

- **START: Start Conversion**

0 = No effect.

1 = Begins analog-to-digital conversion.

32.6.2 ADC Mode Register

Register Name: ADC_MR

Access Type: Read/Write

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

- **TRGEN: Trigger Enable**

TRGEN	Selected TRGEN
0	Hardware triggers are disabled. Starting a conversion is only possible by software.
1	Hardware trigger selected by TRGSEL field is enabled.

- **TRGSEL: Trigger Selection**

TRGSEL			Selected TRGSEL
0	0	0	Reserved
0	0	1	Reserved
0	1	0	Reserved
0	1	1	Reserved
1	0	0	Reserved
1	0	1	Reserved
1	1	0	External trigger
1	1	1	Reserved

- **LOWRES: Resolution**

LOWRES	Selected Resolution
0	10-bit resolution
1	8-bit resolution

- **SLEEP: Sleep Mode**

SLEEP	Selected Mode
0	Normal Mode
1	Sleep Mode

- **PRESCAL: Prescaler Rate Selection**
  ADCClock = MCK / ((PRESCAL+1) * 2)
- **STARTUP: Start Up Time**
  Startup Time = (STARTUP+1) * 8 / ADCClock
- **SHTIM: Sample & Hold Time**
  Sample & Hold Time = (SHTIM+1) / ADCClock

32.6.3 ADC Channel Enable Register

Register Name: ADC_CHER

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
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Enable**

0 = No effect.

1 = Enables the corresponding channel.

32.6.4 ADC Channel Disable Register

Register Name: ADC_CHDR

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
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Disable**

0 = No effect.

1 = Disables the corresponding channel.

Warning: If the corresponding channel is disabled during a conversion or if it is disabled then reenabled during a conversion, its associated data and its corresponding EOC and OVRE flags in ADC_SR are unpredictable.

32.6.5 ADC Channel Status Register

Register Name: ADC_CHSR

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**
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Status**

0 = Corresponding channel is disabled.

1 = Corresponding channel is enabled.

32.6.6 ADC Status Register

Register Name: ADC_SR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion x**

0 = Corresponding analog channel is disabled, or the conversion is not finished.

1 = Corresponding analog channel is enabled and conversion is complete.

- **OVREx: Overrun Error x**

0 = No overrun error on the corresponding channel since the last read of ADC_SR.

1 = There has been an overrun error on the corresponding channel since the last read of ADC_SR.

- **DRDY: Data Ready**

0 = No data has been converted since the last read of ADC_LCDR.

1 = At least one data has been converted and is available in ADC_LCDR.

- **GOVRE: General Overrun Error**

0 = No General Overrun Error occurred since the last read of ADC_SR.

1 = At least one General Overrun Error has occurred since the last read of ADC_SR.

- **ENDRX: End of RX Buffer**

0 = The Receive Counter Register has not reached 0 since the last write in ADC_RCR or ADC_RNCR.

1 = The Receive Counter Register has reached 0 since the last write in ADC_RCR or ADC_RNCR.

- **RXBUFF: RX Buffer Full**

0 = ADC_RCR or ADC_RNCR have a value other than 0.

1 = Both ADC_RCR and ADC_RNCR have a value of 0.

32.6.7 ADC Last Converted Data Register

Register Name: ADC_LCDR

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**
–	–	–	–	–	–	LDATA	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LDATA							

- **LDATA: Last Data Converted**

The analog-to-digital conversion data is placed into this register at the end of a conversion and remains until a new conversion is completed.

32.6.8 ADC Interrupt Enable Register

Register Name: ADC_IER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion Interrupt Enable x**

- **OVREx: Overrun Error Interrupt Enable x**

- **DRDY: Data Ready Interrupt Enable**

- **GOVRE: General Overrun Error Interrupt Enable**
- **ENDRX: End of Receive Buffer Interrupt Enable**
- **RXBUFF: Receive Buffer Full Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

32.6.9 ADC Interrupt Disable Register

Register Name: ADC_IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion Interrupt Disable x**
- **OVREx: Overrun Error Interrupt Disable x**
- **DRDY: Data Ready Interrupt Disable**
- **GOVRE: General Overrun Error Interrupt Disable**
- **ENDRX: End of Receive Buffer Interrupt Disable**
- **RXBUFF: Receive Buffer Full Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

32.6.10 ADC Interrupt Mask Register

Register Name: ADC_IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	14	13	12	11	10	9	8
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion Interrupt Mask x**
- **OVREx: Overrun Error Interrupt Mask x**
- **DRDY: Data Ready Interrupt Mask**
- **GOVRE: General Overrun Error Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask**
- **RXBUFF: Receive Buffer Full Interrupt Mask**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

32.6.11 ADC Channel Data Register

Register Name: ADC_CDRx

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
–	–	–	–	–	–	DATA	
7	6	5	4	3	2	1	0
DATA							

- **DATA: Converted Data**

The analog-to-digital conversion data is placed into this register at the end of a conversion and remains until a new conversion is completed. The Convert Data Register (CDR) is only loaded if the corresponding analog channel is enabled.

33. AT91CAP7E Electrical Characteristics

Note: This chapter contains preliminary values based on prototype silicon. These values are subject to change and will be recharacterized for the production silicon.

33.1 Absolute Maximum Ratings

Table 33-1. Absolute Maximum Ratings*

Operating Temperature (Industrial)-40· C to +85· C

Storage Temperature-60°C to +150°C

Voltage on Input Pins
with Respect to Ground-0.3V to +4.0V

Maximum Operating Voltage
(VDDCORE, VDDBU, VDDPLLB, VDDOSC, and
VDDOSC32)1.5V

Maximum Operating Voltage
(VDDIO, VDDPLLA, and AVDD)4.0V

Total DC Output Current on all I/O lines500 mA

*NOTICE: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

33.2 DC Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C, unless otherwise specified and are certified for a junction temperature up to T_J = 100°C.

Table 33-2. DC Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{VDDCORE}$	DC Supply Core		1.08		1.32	V
V_{VDDBU}	DC Supply Backup		1.08		1.32	V
V_{VDDOSC}	DC Supply Oscillator		1.08		1.32	V
$V_{VDDOSC32}$	DC Supply 32kHz Oscillator		1.08		1.32	V
$V_{VDDPLLA}$	DC Supply PLLA		3.0		3.6	V
$V_{VDDPLLB}$	DC Supply PLLB		1.08		1.32	V
V_{VDDIO}	DC Supply I/Os		3.0		3.6	V
V_{AVDD}	DC Supply ADC		3.0		3.6	V
V_{IL}	Input Low-level Voltage		-0.3		0.8	V
V_{IH}	Input High-level Voltage	V_{VDDIO}	2		V_{VDDIO}+0.3	V
V_{OL}	Output Low-level Voltage				0.4	V
V_{OH}	Output High-level Voltage	V_{VDDIO}	V_{VDDIO}-0.4			V
R_{PULLUP}	Pull-up Resistance	PA0-PA31	40	83	165	kOhm
I_O	Output Current	PA0-PA31			8	mA

Table 33-2. DC Characteristics

I_{SC}	Static Current	On $V_{VDDCORE}$ = 1.2V, MCK = 0 Hz, excluding POR All inputs driven TMS, TDI, TCK, NRST = 1	T_A =25°C		600		µA
			T_A =85°C				
		On V_{VDDBU} = 1.2V, Logic cells consumption, including POR All inputs driven WKUP = 0	T_A =25°C		30		uA
			T_A =85°C				

33.3 Power Consumption

This section contains:

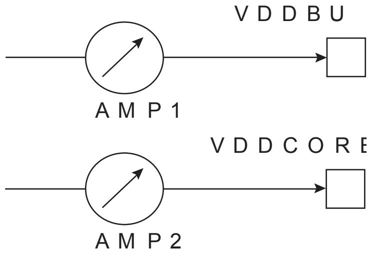
- The typical power consumption of PLLs, Slow Clock (32 kHz) and Main Oscillator.
- The power consumption of power supply in three different modes: Active, Ultra Low-power and Backup.
- The power consumption by peripheral: calculated as the difference in current measurement after having enabled then disabled the corresponding clock.

33.3.1 Power Consumption versus Modes

The values in Table 33-3 and Table 33-4 on page 483 are estimated values of the power consumption with operating conditions as follows:

- $V_{DDIO} = V_{DDPLLA} = V_{AVDD}$ =3.3 V
- $V_{DDCORE} = V_{DDBU} = V_{DDOSC}$ $V_{DDOSC32}$ = 1.2V
- T_A = 25° C
- There is no consumption on the I/Os of the device

Figure 33-1. Measures Schematics



These figures represent the power consumption estimated on the power supplies.

Table 33-3. Power Consumption for different Modes(1)

Mode	Conditions	Consumption	Unit
Active	ARM Core clock is 80MHz. MCK is 80MHz. All peripheral clocks activated. onto AMP2	tbd	mA
Idle	Idle state, waiting an interrupt. All peripheral clocks activated. onto AMP2	tbd	mA
Ultra low power	ARM Core clock is 500Hz. All peripheral clocks de-activated. onto AMP2	tbd	µA
Backup	Device only V_{DDBU} powered onto AMP1	30	µA

Table 33-4. Power Consumption by Peripheral in Active Mode

Peripheral	Consumption	Unit
PIO Controller	tbd	mA
USART	tbd	
UDP	tbd	
ADC	tbd	
SPI	tbd	
Timer Counter Channels 0 to 2	tbd	

33.4 32 kHz Crystal Oscillator Characteristics

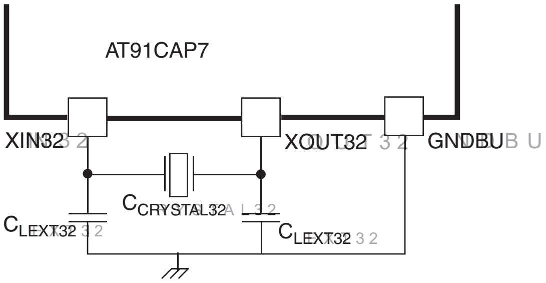
The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C and worst case of power supply, unless otherwise specified.

Table 33-5. 32 kHz Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CP32KHz})$	Crystal Oscillator Frequency			32.768		kHz
$C_{CRYSTAL32}$	Crystal Load Capacitance	Crystal @ 32.768 kHz	6		12.5	pF
C_{LEXT32} (2)	External Load Capacitance	$C_{CRYSTAL32}$ = 6 pF(3)		8		pF
		$C_{CRYSTAL32}$ = 12.5 pF(3)		21		pF
	Duty Cycle		40		60	%
t_{ST}	Startup Time	R_S = 50 kΩ, C_L = 6pF(1)			300	ms
		R_S = 50 kΩ, C_L = 12.5 pF(1)			900	ms
		R_S = 100 kΩ, C_L = 6pF(1)			600	ms
		R_S = 100 kΩ, C_L = 12.5 pF(1)			1200	ms

Notes:
1. R_S is the equivalent series resistance, C_L is the equivalent load capacitance.
2. C_{LEXT32} is determined by taking into account internal parasitic and package load capacitance.
3. Additional board load capacitance should be subtracted from C_{LEXT32}.

Figure 33-2. 32kHz Crystal Connection



33.5 12 MHz Main Oscillator Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C and worst case of power supply, unless otherwise specified.

Table 33-6. Main Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CPMAIN})$	Crystal Oscillator Frequency		8	12	16	MHz
$C_{CRYSTAL}$	Crystal Load Capacitance		15		20	pF
C_{LEXT}	External Load Capacitance	$C_{CRYSTAL}$ = 15 pF[1]		25		pF
		$C_{CRYSTAL}$ = 20 pF[1]		35		
	Duty Cycle		40	50	60	%
t_{ST}	Startup Time				2	ms
I_{DDST}	Standby Current Consumption	Standby mode			2	µA
P_{ON}	Drive Level				150	µW
$I_{DD\ ON}$	Current Dissipation	@ 12MHz		450	700	µA
I_{BYPASS}	Bypass Current Dissipation			3.6	6.2	µW/MHz

Note: 1. Additional board load capacitance should be subtracted from C_{LEXT}.

Figure 33-3. 12 MHz Crystal Connection

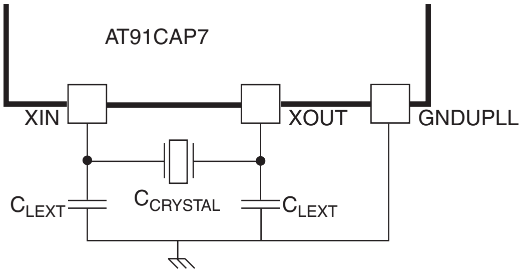


Table 33-7 gives the characteristics that the crystal must satisfy for correct operation with the oscillator.

Table 33-7. Crystal Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
ESR	Equivalent Series Resistor Rs				60	Ω
C_M	Motional Capacitance		5		9	fF
C_S	Shunt Capacitance				7	pF

Table 33-8 gives the Electrical Characteristics of the XIN pin when the oscillator is in Bypass Mode.

Table 33-8. XIN Clock Electrical Characteristics in Bypass Mode

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPXIN})$	XIN Clock Frequency			50	MHz
t_{CPXIN}	XIN Clock Period		20		ns
t_{CHXIN}	XIN Clock High Half-period		0.4 x t_{CPXIN}	0.6 x t_{CPXIN}	

Table 33-8. XIN Clock Electrical Characteristics in Bypass Mode

Symbol	Parameter	Conditions	Min	Max	Units
t_{CLXIN}	XIN Clock Low Half-period		0.4 x t_{CPXIN}	0.6 x t_{CPXIN}	
C_{IN}	XIN Input Capacitance	(1)		5	pF
R_{IN}	XIN Pulldown Resistor	(1)		500	kΩ

Note: These characteristics apply only when Main Oscillator is in Bypass Mode (i.e., when MOSCEN = 0 and OSCBYPASS = 1) in the CKGR_MOR register. See PMC Clock Generator Main Oscillator Register in Section 24. "Advanced Power Management Controller" on page 207.

33.6 PLLA Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C and worst case of power supply, unless otherwise specified.

Table 33-9. Phase Lock Loop A Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
F_{IN}	Input Frequency		1	12	32	MHz
F_{OUT}	Output Frequency	Field OUT of CKGR_PLL is 00	80	160	240	MHz
I_{PLL}	Current Consumption	active mode		2	3	mA
		standby mode			1	µA

Note: 1. Startup time depends on PLL RC filter. A calculation tool is provided by Atmel.

33.7 PLLB Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C and worst case of power supply, unless otherwise specified.

Table 33-10. Phase Lock Loop B Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
F_{IN}	Input Frequency	12 MHz recommended for best filter and USB performance	1	12	32	MHz
F_{OUT}	Output Frequency		50	100	150	MHz
I_{PLL}	Current Consumption	active mode			2.5	mA
		standby mode			TBD	µA

33.8 USB Transceiver Characteristics

33.8.1 Electrical Characteristics

Table 33-11. Electrical Parameters

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Input Levels						
V_{IL}	Low Level				0.8	V
V_{IH}	High Level		2.0			V
V_{DI}	Differential Input Sensivity	|(D+) - (D-)|	0.2			V
V_{CM}	Differential Input Common Mode Range		0.8		2.5	V
C_{IN}	Transceiver capacitance	Capacitance to ground on each line			9.18	pF
I	Hi-Z State Data Line Leakage	0V < V_{IN} < 3.3V	- 10		+ 10	µA
R_{EXT}	Recommended External USB Series Resistor	In series with each USB pin with ±5%		27		Ω
Output Levels						
V_{OL}	Low Level Output	Measured with R_L of 1.425 kΩ tied to 3.6V	0.0		0.3	V
V_{OH}	High Level Output	Measured with R_L of 14.25 kΩ tied to GND	2.8		3.6	V
V_{CRS}	Output Signal Crossover Voltage	Measure conditions described in Figure 33-4	1.3		2.0	V

33.8.2 Switching Characteristics

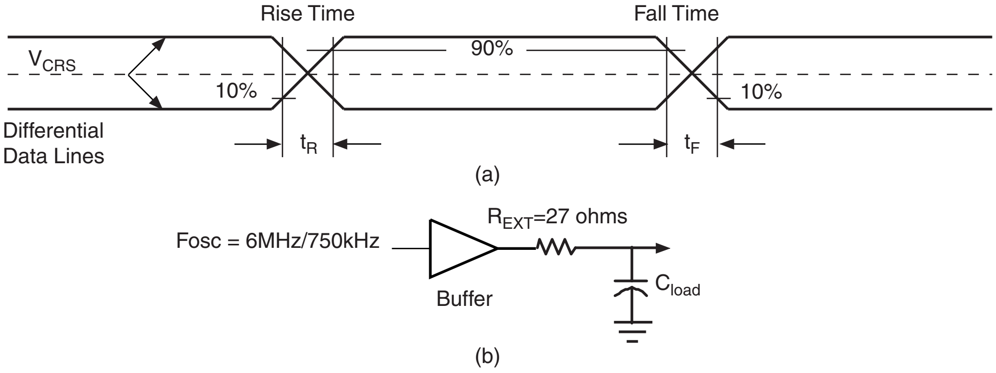
Table 33-12. In Low Speed

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{FR}	Transition Rise Time	C_{LOAD} = 400 pF	75		300	ns
t_{FE}	Transition Fall Time	C_{LOAD} = 400 pF	75		300	ns
t_{FRFM}	Rise/Fall time Matching	C_{LOAD} = 400 pF	80		125	%

Table 33-13. In Full Speed

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{FR}	Transition Rise Time	C_{LOAD} = 50 pF	4		20	ns
t_{FE}	Transition Fall Time	C_{LOAD} = 50 pF	4		20	ns
t_{FRFM}	Rise/Fall time Matching		90		111.11	%

Figure 33-4. USB Data Signal Rise and Fall Times



Rise Time
Fall Time
VCRS
90%
10%
10%
Differential Data Lines
tR
tF
(a)
Fosc = 6MHz/750kHz
REXT=27 ohms
Buffer
Cload
(b)

33.9 ADC

Table 33-14. Channel Conversion Time and ADC CLock

Parameter	Conditions	Min	Typ	Max	Units
ADC Clock Frequency	10-bit resolution mode			13.2	MHz
ADC Clock Frequency	8-bit resolution mode			TBD	MHz
Startup Time	Return from Idle Mode			40	µs
Track and Hold Acquisition Time		500			ns
Conversion Time	ADC Clock = 13.2 MHz			1.74	µs
Throughput Rate	ADC Clock = 13.2 MHz			440(1)	kSPS

Notes: 1. Corresponds to 30 clock cycles at 13.2 MHz: 500nS (7clock cycles) for track and hold acquisition time and 23 clock cycles for conversion.

Table 33-15. External Voltage Reference Input

Parameter	Conditions	Min	Typ	Max	Units
ADVREF Input Voltage Range		2.6		AVDD	V
ADVREF Average Current	Average on all DAC codes			600	µA
Operating Current on AVDD	Average on 4 conversions full speed			400	µA
Operating Current on VDDC	Average on 4 conversions full speed			80	µA
Standby Current on AVDD				300	nA
Standby Current on ADVREF				300	nA
Standby Current on VDDC				600	nA

Table 33-16. Analog Inputs

Parameter	Min	Typ	Max	Units
Input Voltage Range	0		ADVREF	
Input Leakage Current		1		µA
Input Capacitance	6	8	10	pF

The user can drive ADC input with impedance up to:

- $Z_{OUT} \leq$ (SHTIM -500) x 12.5

with SHTIM (Sample and Hold Time register) expressed in ns and Z_{OUT} expressed in ohms.

Table 33-17. Transfer Characteristics

Parameter	Min	Typ	Max	Units
Resolution		10		Bit
Integral Non-linearity			±2	LSB
Differential Non-linearity			±0.9	LSB
Offset Error	-1.5	0.5	2.5	LSB
Gain Error			±2	LSB

33.10 Timings

33.10.1 Corner Definition

Table 33-18. Corner Definition

Corner	Process	Temp (External ; Junction)	VDDCORE: 1.2V	VDDIO: 3.3V
MAX	Slow	85°C ; 100°C	1.10V	3.0V
STH	Slow	85°C; 100°C	1.2V	3.3V
MIN	Fast	-40C; -40C	1.32V	3.6V

Timings in MAX corner always result from the extraction and comparison of timings in MAX and MIN corners.

Timings in STH corner always result from the extraction and comparision of timings in STH and MIN corners.

33.10.2 Processor Clock

Table 33-19. Processor Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPPCK})$	Processor Clock Frequency	Corner MAX		80	MHz
$1/(t_{CPPCK})$	Processor Clock Frequency	Corner STH		TBD	MHz

33.10.3 Maximum Speed of the I/Os

Criteria used to define the maximum frequency of the I/Os:

- output duty cycle (40%-60%)
- minimum output swing: 100mV to VDDIO - 100mV
- Addition of rising and falling time inferior to 75% of the period

Table 33-20.

Symbol	Parameter	Conditions	Min	Max	Units
FreqMax	Pin Group x(1) frequency	3.3V domain (2)		TBD	MHz
		1.8V domain (3)		TBD	MHz
PulseminH	Pin Group(1) High Level Pulse Width	3.3V domain (2)	TBD		ns
		1.8V domain (3)	TBD		ns
PulseminL	Pin Group x(1) Low Level Pulse Width	3.3V domain (2)	TBD		ns
		1.8V domain (3)	TBD		ns

Notes: 1. Pin Group x = To Be Defined for each product

2. 3.3V domain: V_{VDDIOP} from 3.0V to 3.6V, maximum external capacitor = 40pF

3. 1.8V domain: V_{VDDIOP} from 1.65V to 1.95V, maximum external capacitor = 20pF

33.10.4 SMC Timings

33.10.4.1 Capacitance

Timings are given assuming a capacitance load on data, control and address pads.

Table 33-21. Capacitance Load

	Corner		
Supply	MAX	STH	MIN
3.3V	50pF	50pF	0 pF

In the following tables, t_{CPMCK} is MCK period.

33.10.4.2 Read Timings

Table 33-22. SMC Read Signals - NRD Controlled (READ_MODE= 1)

Symbol	**Parameter**	**Min**		**Units**
	VDDIO supply	**3.3V**		
NO HOLD SETTINGS (nrd hold = 0)				
SMC$_1$	Data Setup before NRD High	TBD		ns
SMC$_2$	Data Hold after NRD High	TBD		ns
HOLD SETTINGS (nrd hold …0)				
SMC$_3$	Data Setup before NRD High	TBD		ns
SMC$_4$	Data Hold after NRD High	TBD		ns
HOLD or NO HOLD SETTINGS (nrd hold …0, nrd hold =0)				
SMC$_5$	NBS0/A0, NBS1, NBS2/A1, NBS3, A2 - A25 Valid before NRD High	(nrd setup + nrd pulse)* t_{CPMCK} + TBD		ns
SMC$_6$	NCS low before NRD High	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} + TBD		ns
SMC$_7$	NRD Pulse Width	nrd pulse * t_{CPMCK} + TBD		ns

Table 33-23. SMC Read Signals - NCS Controlled (READ_MODE= 0)

Symbol	**Parameter**	**Min**	**Units**
	VDDIO supply	**3.3V**	
NO HOLD SETTINGS (ncs rd hold = 0)			
SMC$_8$	Data Setup before NCS High	TBD	ns
SMC$_9$	Data Hold after NCS High	TBD	ns
HOLD SETTINGS (ncs rd hold …0)			
SMC$_{10}$	Data Setup before NCS High	TBD	ns
SMC$_{11}$	Data Hold after NCS High	TBD	ns
HOLD or NO HOLD SETTINGS (ncs rd hold …0, ncs rd hold = 0)			

Table 33-23. SMC Read Signals - NCS Controlled (READ_MODE= 0)

SMC_{12}	NBS0/A0, NBS1, NBS2/A1, NBS3, A2 - A25 valid before NCS High	(ncs rd setup + ncs rd pulse)* t_{CPMCK} + TBD	ns
SMC_{13}	NRD low before NCS High	(ncs rd setup + ncs rd pulse - nrd setup)* t_{CPMCK} + TBD	ns
SMC_{14}	NCS Pulse Width	ncs rd pulse length * t_{CPMCK} + TBD	ns

33.10.4.3 Write Timings

Table 33-24. SMC Write Signals - NWE controlled (WRITE_MODE = 1)

Symbol	Parameter	Min	Max	Units
HOLD or NO HOLD SETTINGS (nwe hold ...0, nwe hold = 0)				
SMC_{15}	Data Out Valid before NWE High	nwe pulse * t_{CPMCK} + TBD		ns
SMC_{16}	NWE Pulse Width	nwe pulse * t_{CPMCK} + TBD		ns
SMC_{17}	NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25 valid before NWE low	nwe setup * t_{CPMCK} + TBD		ns
SMC_{18}	NCS low before NWE high	(nwe setup - ncs rd setup + nwe pulse) * t_{CPMCK} + TBD		ns
HOLD SETTINGS (nwe hold ...0)				
SMC_{19}	NWE High to Data OUT, NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25 change	nwe hold * t_{CPMCK} + TBD		ns
SMC_{20}	NWE High to NCS Inactive (1)	(nwe hold - ncs wr hold)* t_{CPMCK} + TBD		ns
NO HOLD SETTINGS (nwe hold = 0)				
SMC_{21}	NWE High to Data OUT, NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25, NCS change(1)	TBD		ns

Notes: 1. hold length = total cycle duration - setup duration - pulse duration. “hold length” is for “ncs wr hold length” or “NWE hold length”.

Table 33-25. SMC Write NCS Controlled (WRITE_MODE = 0)

Symbol	Parameter	Min	Units
		3.3V Supply	
SMC_{22}	Data Out Valid before NCS High	ncs wr pulse * t_{CPMCK} + TBD	ns
SMC_{23}	NCS Pulse Width	ncs wr pulse * t_{CPMCK} + TBD	ns
SMC_{24}	NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25 valid before NCS low	ncs wr setup * t_{CPMCK} + TBD	ns

Table 33-25. SMC Write NCS Controlled (WRITE_MODE = 0)

Symbol	Parameter	Min 3.3V Supply	Units
SMC_{25}	NWE low before NCS high	(ncs wr setup - nwe setup + ncs pulse)* t_{CPMCK} + TBD	ns
SMC_{26}	NCS High to Data Out, NBS0/A0, NBS1, NBS2/A1, NBS3, A2 - A25, change	ncs wr hold * t_{CPMCK} + TBD	ns
SMC_{27}	NCS High to NWE Inactive	(ncs wr hold - nwe hold)* t_{CPMCK} + TBD	ns

Figure 33-5. SMC Timings - NCS Controlled Read and Write

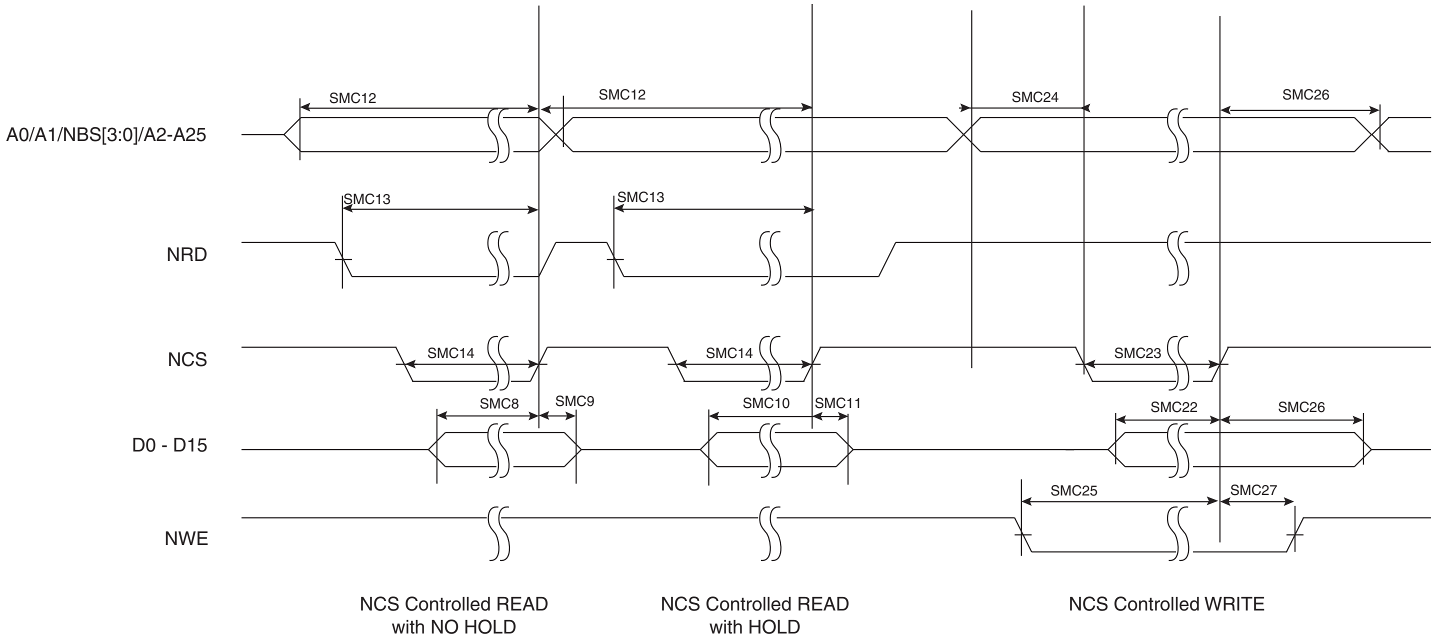
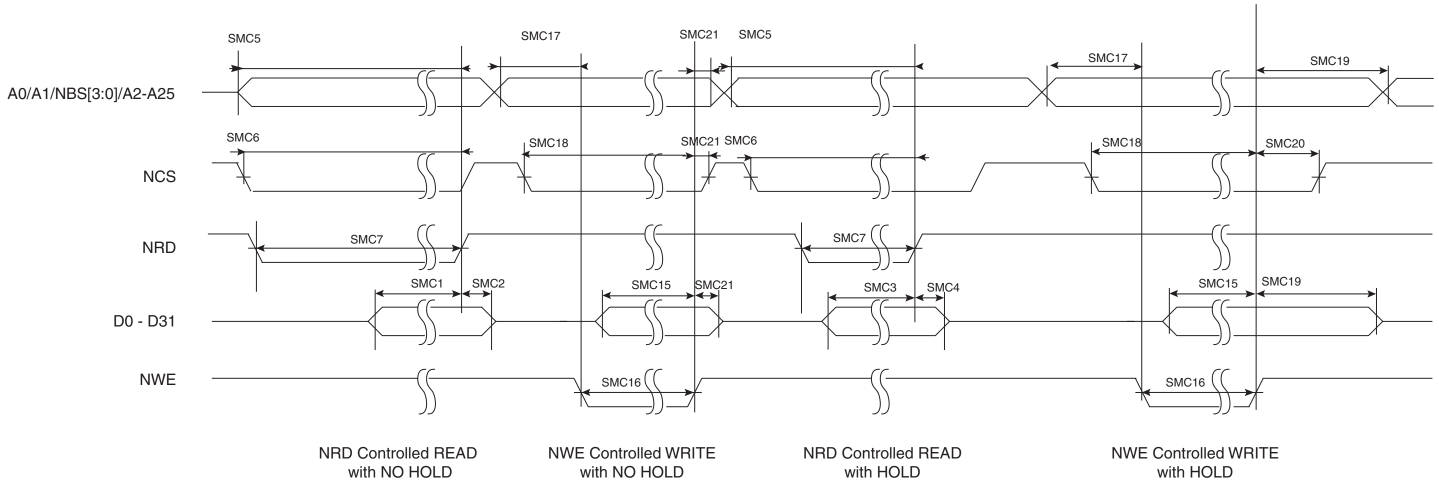

SMC12
SMC12
SMC24
SMC26
A0/A1/NBS[3:0]/A2-A25
SMC13
SMC13
NRD
NCS
SMC14
SMC14
SMC23
SMC8
SMC9
SMC10
SMC11
SMC22
SMC26
D0 - D15
SMC25
SMC27
NWE
NCS Controlled READ with NO HOLD
NCS Controlled READ with HOLD
NCS Controlled WRITE

Figure 33-6. SMC Timings - NRD Controlled Read and NWE Controlled Write


SMC5
SMC17
SMC21
SMC5
SMC17
SMC19
A0/A1/NBS[3:0]/A2-A25
SMC6
SMC18
SMC21
SMC6
SMC18
SMC20
NCS
SMC7
SMC7
NRD
SMC1
SMC2
SMC15
SMC21
SMC3
SMC4
SMC15
SMC19
D0 - D31
NWE
SMC16
SMC16
NRD Controlled READ with NO HOLD
NWE Controlled WRITE with NO HOLD
NRD Controlled READ with HOLD
NWE Controlled WRITE with HOLD

33.10.5 SDRAMC Timings

The SDRAM Controller satisfies the timing of standard SDRAM modules given in Table 33-28 and in **MAX and STH corners.**

Timings are given assuming a capacitance load on data, control and address pads :

Table 33-26. Capacitance Load on Data, Control and Address Pads

	Corner		
Supply	MAX	STH	MIN
3.3V	50pF	50pF	0 pF
1.8V	30 pF	30 pF	0 pF

Table 33-27. Capacitance Load on SDCK Pad

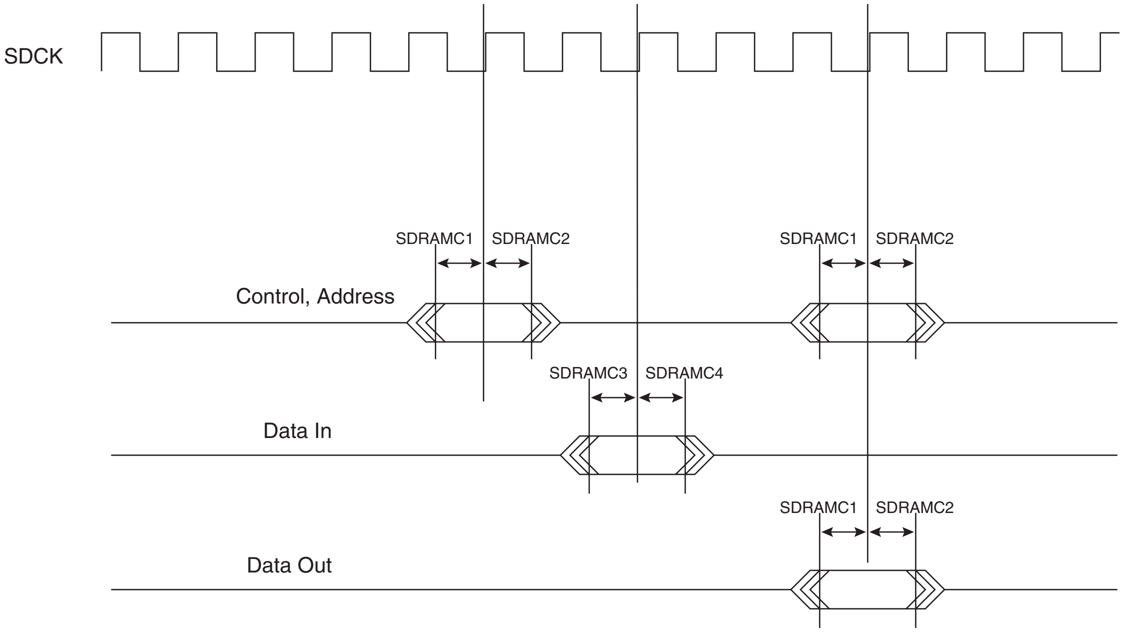
	Corner		
Supply	MAX	STH	MIN
3.3V	10pF	10pF	10pF
1.8V	10pF	10pF	10pF

Table 33-28. SDRAMC Timings

		Min	
Symbol	**Parameter**	**3.3V Supply**	**Units**
$SDRAMC_1$	Control/Address/Data out valid before SDCK Rising Edge(1)	$0.5*t_{CPMCK+}$TBD	ns
$SDRAMC_2$	Control/Address/Data out change after SDCK Rising Edge(1)	$0.5*t_{CPMCK+}$TBD	ns
$SDRAMC_3$	Data Input Setup before SDCK Rising Edge	TBD	ns
$SDRAMC_4$	Data Input Hold after SDCK Rising Edge	TBD	ns

Control/Address is the set of following timings : A0-A9, A11-A13, SDA10, SDCKE, SDCS, RAS, CAS, BAx, DQMx, and SDWE

Figure 33-7. SDRAMC Timings


SDCK
SDRAMC1
SDRAMC2
Control, Address
SDRAMC1
SDRAMC2
SDRAMC3
SDRAMC4
Data In
SDRAMC1
SDRAMC2
Data Out

33.10.6 SPI

Figure 33-8. SPI Master Mode with (CPOL = NCPHA = 0) or (CPOL= NCPHA= 1)

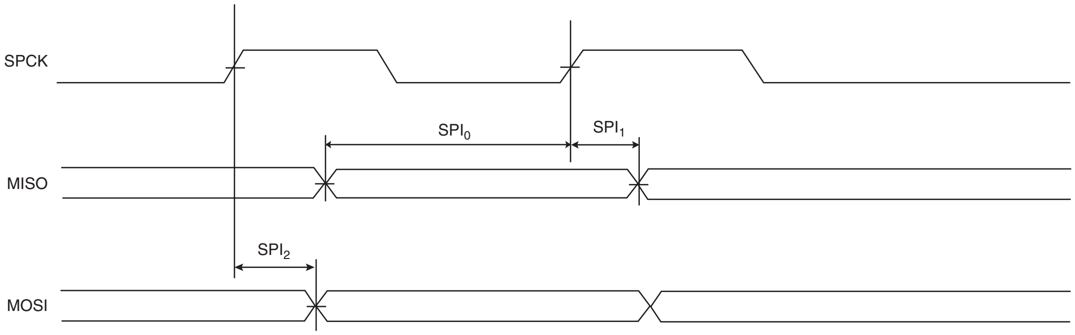

SPCK
SPI0
SPI1
MISO
SPI2
MOSI

Figure 33-9. SPI Master Mode with (CPOL=0 and NCPHA=1) or (CPOL=1 and NCPHA=0)

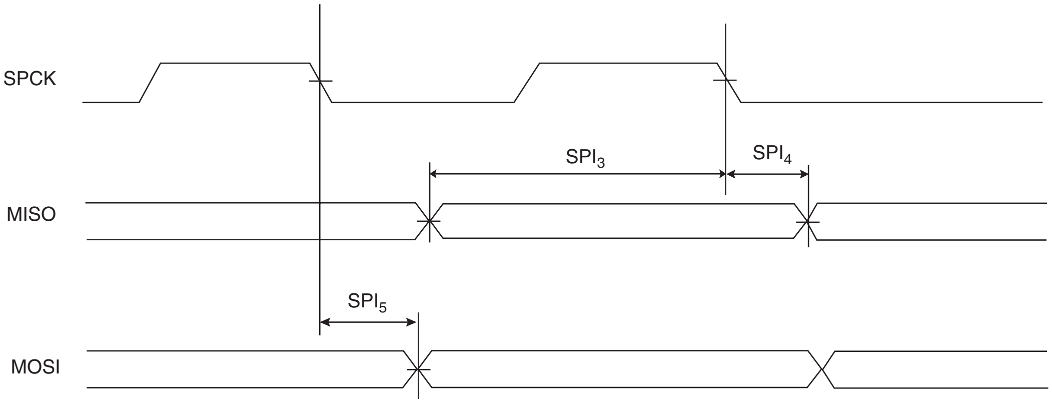

SPCK
SPI3
SPI4
MISO
SPI5
MOSI

Figure 33-10. SPI Slave Mode with (CPOL=0 and NCPHA=1) or (CPOL=1 and NCPHA=0)

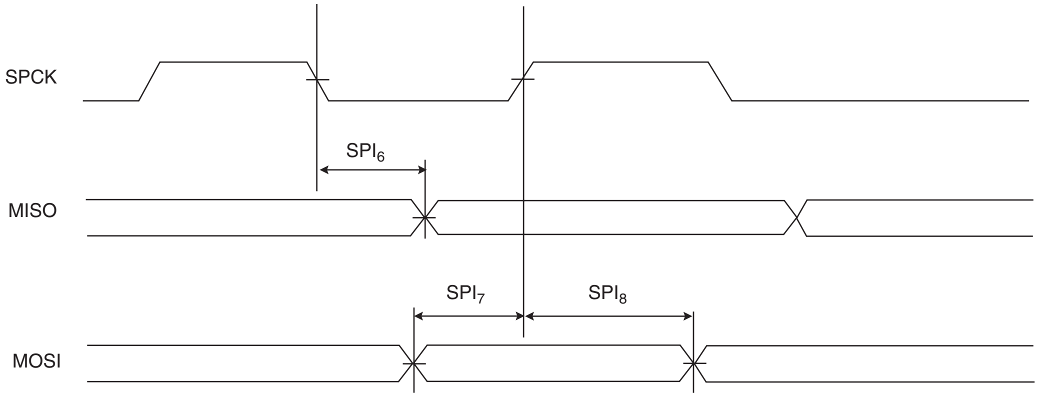


Figure 33-11. SPI Slave Mode with (CPOL = NCPHA = 0) or (CPOL= NCPHA= 1)

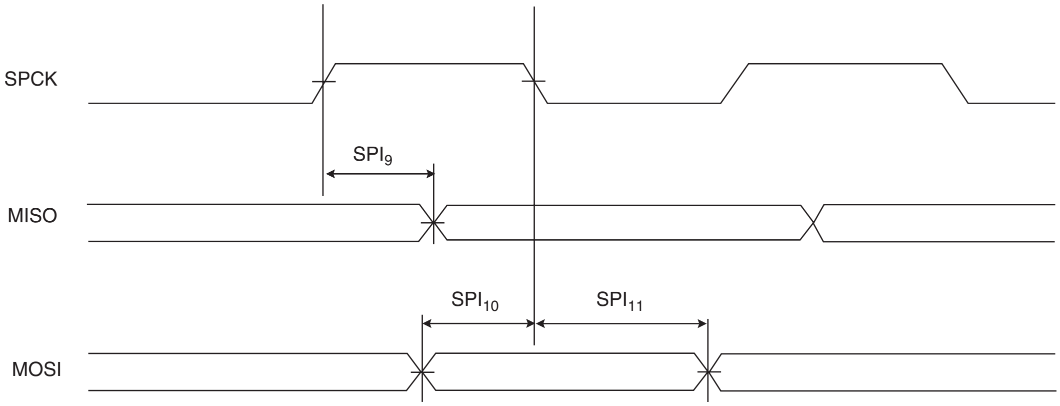


Table 33-29. SPI Timings

Symbol	Parameter	Cond	Min	Max	Units
Master Mode					
SPI_0	MISO Setup time before SPCK rises	(1)	TBD + 0.5*t_{CPMCK}		ns
SPI_1	MISO Hold time after SPCK rises	(1)	TBD - 0.5* t_{CPMCK}		ns
SPI_2	SPCK rising to MOSI valid	(1)		TBD	ns
SPI_2	SPCK rising to MOSI change	(1)	TBD		ns
SPI_3	MISO Setup time before SPCK falls	(1)	TBD + 0.5*t_{CPMCK}		ns
SPI_4	MISO Hold time after SPCK falls	(1)	TBD - 0.5* t_{CPMCK}		ns
SPI_5	SPCK falling to MOSI valid	(1)		TBD	ns
SPI_2	SPCK falling to MOSI change	(1)	TBD		ns
Slave Mode					
SPI_6	SPCK falling to MISO valid	(1)		TBD	ns
SPI_6	SPCK falling to MISO change	(1)	TBD		ns

Table 33-29. SPI Timings

Symbol	Parameter	Cond	Min	Max	Units
SPI_7	MOSI Setup time before SPCK rises	(1)	TBD		ns
SPI_8	MOSI Hold time after SPCK rises	(1)	TBD		ns
SPI_9	SPCK rising to MISO valid	(1)		TBD	ns
SPI_9	SPCK rising to MISO change	(1)	TBD		ns
SPI_{10}	MOSI Setup time before SPCK falls	(1)	TBD		ns
SPI_{11}	MOSI Hold time after SPCK falls	(1)	TBD		ns
SPI_{12}	NPCS0,1,2,3 to MOSI	(1)	TBD		ns
SPI_{13}	NPCS0,1,2,3 to MISO	(1)	TBD		ns

Notes: 1. Cload is 8pF for MISO and 6pF for SPCK and MOSI.

34. AT91CAP7E Mechanical Characteristics

34.1 Thermal Considerations

34.1.1 Thermal Data

Table 34-1 summarizes the thermal resistance data depending on the package.

Table 34-1. Thermal Resistance Data

Symbol	Parameter	Condition	Package	Typ	Unit
θ_{JA}	Junction-to-ambient thermal resistance	Still Air	LFBGA 225 13x13mm 0.8mm pitch	35.3	°C/W
θ_{JC}	Junction-to-case thermal resistance	Still Air	LFBGA 225 13x13mm 0.8mm pitch	28	°C/W

34.1.2 Junction Temperature

The average chip-junction temperature, T_J, in °C can be obtained from the following:

4. $T_J = T_A + (P_D \times \theta_{JA})$
5. $T_J = T_A + (P_D \times (\theta_{HEATSINK} + \theta_{JC}))$

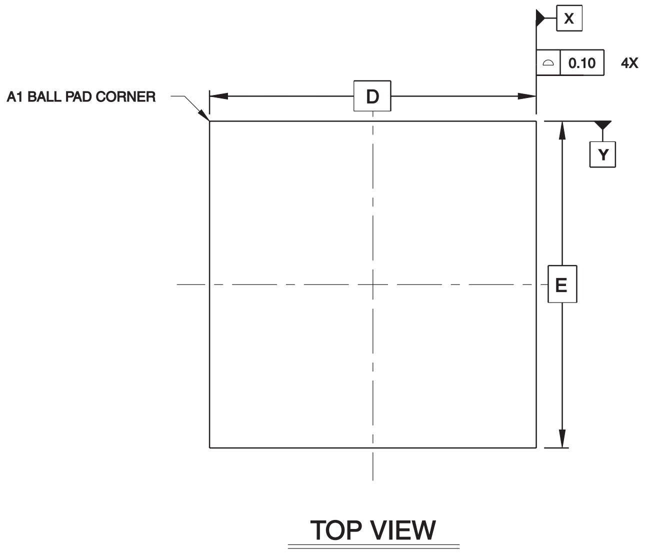
where:

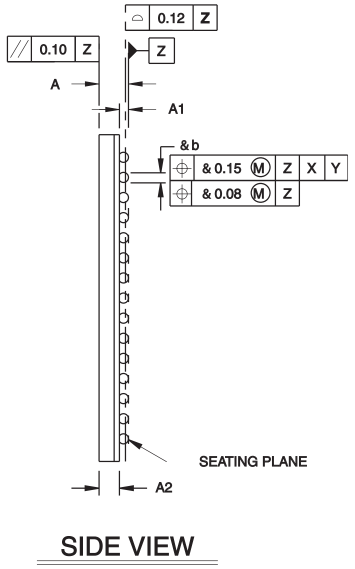
- θ_{JA} = package thermal resistance, Junction-to-ambient (°C/W), provided in Table 34-1 on page 499.
- θ_{JC} = package thermal resistance, Junction-to-case thermal resistance (°C/W), provided in Table 34-1 on page 499.
- $\theta_{HEAT\ SINK}$ = cooling device thermal resistance (°C/W), provided in the device datasheet.
- P_D = device power consumption (W) estimated from data provided in the section Section 33.3 ”Power Consumption” on page 482.
- T_A = ambient temperature (°C).

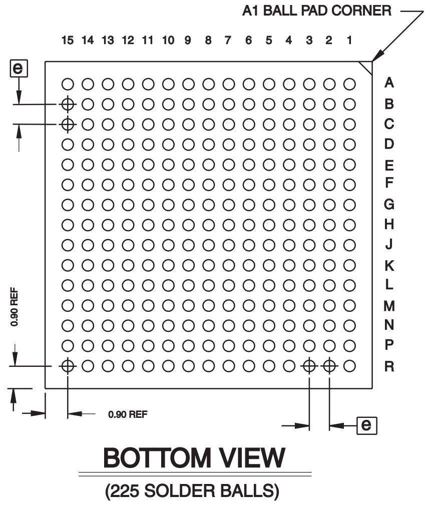
From the first equation, the user can derive the estimated lifetime of the chip and decide if a cooling device is necessary or not. If a cooling device is to be fitted on the chip, the second equation should be used to compute the resulting average chip-junction temperature T_J in °C.

34.2 Package Drawings

225-ball LFBGA Package DrawingSoldering Profile


X
0.10
4X
A1 BALL PAD CORNER
D
Y
E
TOP VIEW


0.12 Z
0.10 Z
Z
A
A1
& b
& 0.15 M Z X Y
& 0.08 M Z
SEATING PLANE
A2
SIDE VIEW


A1 BALL PAD CORNER
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
A B C D E F G H J K L M N P R
e
0.90 REF
0.90 REF
e
BOTTOM VIEW
(225 SOLDER BALLS)

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
D		13.00 BSC		
E		13.00 BSC		
A	–	–	1.70	3
A1	0.25	–	–	3
A2	0.85	–	–	
e		0.80 BSC		
b	0.45	0.50	0.55	4

Table 34-2. Soldering Information

Ball Land	0.530 mm +/- 0.03
Soldering Mask Opening	0.370mm to 0.03 mm

Table 34-3. Device and 225-ball LFBGA Package Maximum Weight

365.2	mg

Table 34-4. 225-ball LFBGA Package Characteristics

Moisture Sensitivity Level	3

Table 34-5. Package Reference

JEDEC Drawing Reference	MO-205
JESD97 Classification	e1

35. AT91CAP7E Ordering Information

Table 35-1. AT91CAP7E Ordering Information

Ordering Code	Package	Package Type	Temperature Operating Range
AT91CAP7E	BGA225	RoHS Compliant	Industrial -40°C to 85°C

36. Revision History

Doc. Rev.	Date	Comments
8549A	10/2008	Initial document release.

Table of Contents

1 Description 2

2 Block Diagram 3

3 Signal Description 4

4 Package and Pinout 11

4.1Mechanical Overview of the 225-ball LFBGA Package 11

4.2225-ball LFBGA Package Pinout 11

5 Power Considerations 14

5.1Power Supplies 14

5.2Power Consumption 14

6 I/O Line Considerations 15

6.1JTAG Port Pins 15

6.2Test Pin 15

6.3Reset Pins 15

6.4PIO Controllers 15

6.5Shut Down Logic pins 15

7 Processor and Architecture 16

7.1ARM7TDMI Processor 16

7.2Debug and Test Features 16

7.3Bus Matrix 16

7.4.1Matrix Masters 17

7.5.2Matrix Slaves 17

7.6Peripheral DMA Controller 17

8 Memories 18

8.1Embedded Memories 18

8.2Memory Mapping 18

8.3Internal Memory Mapping 19

8.4.1Internal 160-kBytes Fast SRAM 19

8.5.2Boot Memory 19

8.6Boot Program 19

8.7External Memories Mapping 19

8.8External Bus Interface 19

8.9.1Static Memory Controller 20

8.10.2SDRAM Controller 20

9 System Controller 22
9.1System Controller Block Diagram 23
9.2System Controller Mapping 24
9.3Reset Controller 25
9.4Shut Down Controller 25
9.5Clock Generator 25
9.6Power Management Controller 26
9.7Periodic Interval Timer 27
9.8Watchdog Timer 27
9.9Real-Time Timer 27
9.10General-Purpose Backed-up Registers 28
9.11Backup Power Switch 28
9.12Advanced Interrupt Controller 28
9.13Debug Unit 28
9.14Chip Identification 29
9.15PIO Controllers 29
9.16User Interface 30
9.17.1Special System Controller Register Mapping 30
9.18.2Oscillator Mode Register 30
9.19.3General Purpose Backup Register 31
10 Peripherals 32
10.1Peripheral Mapping 32
10.2Peripheral Identifiers 34
10.3Peripheral Interrupts and Clock Control 35
10.4.1System Interrupt 35
10.5.2External Interrupts 35
10.6.3Timer Counter Interrupts 35
10.7Peripherals Signals Multiplexing on I/O Lines 35
10.8.1PIO Controller A Multiplexing 36
10.9.2PIO Controller B Multiplexing 37
10.10.3Resource Multiplexing 37
10.11Embedded Peripherals Overview 38
10.12.1Serial Peripheral Interface 38
10.13.2USART 38
10.14.3Timer Counter 39
10.15.4USB Device Port 39
10.16.5Analog to Digital Converter 39
11 FPGA Interface (FPIF) 41
11.1Description 41

11.2System Requirements and Integration 41

11.3Functional Description 42

- 11.4.1Interface Modules 43
- 11.5.2Serializer Modules 43
- 11.6.3Serializer Programmability 44
- 11.7.4Transfer Timing 45

11.8Programmability Options 46

- 11.9.1Mode-Bits 46
- 11.10.2PIO Controller B Multiplexing 47
- 11.11.3Other MPIO Signal Assignments/Multiplexing 48

11.12Interfacing using PIO 49

- 11.13.1PIO-FPGA Connections 50
- 11.14.2PIO-FPGA Access Routines 50
- 11.15.3PIO-FPGA Waveforms 51

11.16Interfacing using EBI 52

- 11.17.1EBI-FPGA Connections 52
- 11.18.2EBI TIming 52

12 ARM7TDMI Processor Overview 55

12.1Overview 55

12.2ARM7TDMI Processor 55

- 12.3.1Instruction Type 55
- 12.4.2Data Type 55
- 12.5.3ARM7TDMI Operating Mode 55
- 12.6.4ARM7TDMI Registers 56
- 12.7.5ARM Instruction Set Overview 58
- 12.8.6Thumb Instruction Set Overview 59

13 CAP7E Debug and Test 61

13.1Overview 61

13.2Block Diagram 61

13.3Application Examples 62

- 13.4.1Debug Environment 62
- 13.5.2Test Environment 63

13.6Debug and Test Pin Description 63

13.7Functional Description 64

- 13.8.1Test Pin 64
- 13.9.2Embedded In-circuit Emulator 64
- 13.10.3Debug Unit 64
- 13.11.4IEEE 1149.1 JTAG Boundary Scan 64
- 13.12.5ID Code Register 65

14 Reset Controller (RSTC) 67

14.1Description 67

14.2Block Diagram 67

14.3Functional Description 67

14.4.1Reset Controller Overview 67

14.5.2NRST Manager 68

14.6.3Reset States 69

14.7.4Reset State Priorities 73

14.8.5Reset Controller Status Register 73

14.9Reset Controller (RSTC) User Interface 74

14.10.1Reset Controller Control Register 75

14.11.2Reset Controller Status Register 75

14.12.3Reset Controller Mode Register 76

15 Real-time Timer (RTT) 79

15.1Description 79

15.2Block Diagram 79

15.3Functional Description 79

15.4Real-time Timer User Interface 81

15.5.1Register Mapping 81

15.6.2Real-time Timer Mode Register 82

15.7.3Real-time Timer Alarm Register 83

15.8.4Real-time Timer Value Register 83

15.9.5Real-time Timer Status Register 84

16 Periodic Interval Timer (PIT) 85

16.1Description 85

16.2Block Diagram 85

16.3Functional Description 85

16.4Periodic Interval Timer (PIT) User Interface 87

16.5.1Periodic Interval Timer Mode Register 87

16.6.2Periodic Interval Timer Status Register 88

16.7.3Periodic Interval Timer Value Register 88

16.8.4Periodic Interval Timer Image Register 89

17 Watchdog Timer (WDT) 91

17.1Description 91

17.2Block Diagram 91

17.3Functional Description 91

17.4User Interface 93

17.5.1Register Mapping 93

17.6.2Watchdog Timer Control Register 93

17.7.3Watchdog Timer Mode Register 94

17.8.4Watchdog Timer Status Register 95

18 Shutdown Controller (SHDWC) 97

18.1Description 97

18.2Block Diagram 97
18.3I/O Lines Description 97
18.4Product Dependencies 97
18.5.1Power Management 97
18.6Functional Description 97
18.7Shutdown Controller (SHDWC) User Interface 98
18.8.1Register Mapping 98
18.9.2Shutdown Control Register 99
18.10.3Shutdown Mode Register 100
18.11.4Shutdown Status Register 101

19 Bus Matrix 103

19.1Description 103
19.2Memory Mapping 103
19.3Special Bus Granting Mechanism 103
19.4.1No Default Master 103
19.5.2Last Access Master 103
19.6.3Fixed Default Master 103
19.7Arbitration 104
19.8Arbitration Rules 104
19.9.1Undefined Length Burst Arbitration 104
19.10.2Slot Cycle Limit Arbitration 105
19.11.3Round-Robin Arbitration 105
19.12.4Fixed Priority Arbitration 105
19.13AHB Generic Bus Matrix User Interface 106
19.14.1Bus Matrix Master Configuration Registers 108
19.15.2Bus Matrix Slave Configuration Registers 109
19.16.3Bus Matrix Priority Registers A For Slaves 110
19.17.4Bus Matrix Priority Registers B For Slaves 110
19.18.5Bus Matrix Master Remap Control Register 111
19.19.6EBI Chip Select Assignment Register 112
19.20.7Matrix USB Pad Pull-up Control Register 113

20 External Bus Interface (EBI) 115

20.1Overview 115
20.2Block Diagram 116
20.3I/O Lines Description 117
20.4Application Example 118
20.5.1Hardware Interface 118
20.6.2Connection Examples 121
20.7Product Dependencies 121
20.8.1I/O Lines 121
20.9Functional Description 122
20.10.1Bus Multiplexing 122

20.11.2Pull-up Control 122
20.12.3Static Memory Controller 122
20.13.4SDRAM Controller 122
20.14.5CompactFlash Support 122
20.15.6NAND Flash Support 127

21 Static Memory Controller (SMC) 131

21.1Description 131

21.2I/O Lines Description 131

21.3Multiplexed Signals 131

21.4Application Example 132

21.5.1Hardware Interface 132

21.6Product Dependencies 132

21.7.1I/O Lines 132

21.8External Memory Mapping 133

21.9Connection to External Devices 133

21.10.1Data Bus Width 133
21.11.2Byte Write or Byte Select Access 133

21.12Standard Read and Write Protocols 137

21.13.1Read Waveforms 138
21.14.2Read Mode 140
21.15.3Write Waveforms 142
21.16.4Write Mode 144
21.17.5Coding Timing Parameters 145
21.18.6Reset Values of Timing Parameters 146
21.19.7Usage Restriction 146

21.20Automatic Wait States 146

21.21.1Chip Select Wait States 146
21.22.2Early Read Wait State 147
21.23.3Reload User Configuration Wait State 149
21.24.4Read to Write Wait State 150

21.25Data Float Wait States 150

21.26.1READ_MODE 150
21.27.2TDF Optimization Enabled (TDF_MODE = 1) 152
21.28.3TDF Optimization Disabled (TDF_MODE = 0) 152

21.29External Wait 154

21.30.1Restriction 154
21.31.2Frozen Mode 155
21.32.3Ready Mode 157
21.33.4NWAIT Latency and Read/write Timings 159

21.34Slow Clock Mode 160

21.35.1Slow Clock Mode Waveforms 160
21.36.2Switching from (to) Slow Clock Mode to (from) Normal Mode 161

21.37Asynchronous Page Mode 163

21.38.1Protocol and Timings in Page Mode 163

21.39.2Byte Access Type in Page Mode 164
21.40.3Page Mode Restriction 164
21.41.4Sequential and Non-sequential Accesses 164
21.42Static Memory Controller (SMC) User Interface 166
21.43.1SMC Setup Register 167
21.44.2SMC Pulse Register 168
21.45.3SMC Cycle Register 169
21.46.4SMC MODE Register 170

22 SDRAM Controller (HSDRAMC) 173

22.1Description 173
22.2I/O Lines Description 173
22.3Application Example 173
22.4Software Interface 173
22.5.132-bit Memory Data Bus Width 174
22.6.216-bit Memory Data Bus Width 175
22.7Product Dependencies 176
22.8.1SDRAM Device Initialization 176
22.9.2I/O Lines 177
22.10.3Interrupt 177
22.11Functional Description 177
22.12.1SDRAM Controller Write Cycle 177
22.13.2SDRAM Controller Read Cycle 178
22.14.3Border Management 179
22.15.4SDRAM Controller Refresh Cycles 180
22.16.5Power Management 181
22.17SDRAM Controller User Interface 185
22.18.1SDRAMC Mode Register 186
22.19.2SDRAMC Refresh Timer Register 187
22.20.3SDRAMC Configuration Register 187
22.21.4SDRAMC High Speed Register 189
22.22.5SDRAMC Low Power Register 190
22.23.6SDRAMC Interrupt Enable Register 191
22.24.7SDRAMC Interrupt Disable Register 191
22.25.8SDRAMC Interrupt Mask Register 192
22.26.9SDRAMC Interrupt Status Register 192
22.27.10SDRAMC Memory Device Register 193

23 Peripheral DMA Controller (PDC) 195

23.1Description 195
23.2Block Diagram 196
23.3Functional Description 196
23.4.1Configuration 196
23.5.2Memory Pointers 197
23.6.3Transfer Counters 197
23.7.4Data Transfers 198

23.8.5PDC Flags and Peripheral Status Register 198
23.9Peripheral DMA Controller (PDC) User Interface 199
23.10.1Receive Pointer Register 200
23.11.2Receive Counter Register 200
23.12.3Transmit Pointer Register 201
23.13.4Transmit Counter Register 201
23.14.5Receive Next Pointer Register 202
23.15.6Receive Next Counter Register 202
23.16.7Transmit Next Pointer Register 203
23.17.8Transmit Next Counter Register 203
23.18.9Transfer Control Register 204
23.19.10Transfer Status Register 205

24 Advanced Power Management Controller 207

24.1Clock Generator 207
24.2.1Description 207
24.3.2Slow Clock Crystal Oscillator 207
24.4.3Slow Clock RC Oscillator 207
24.5.4Main Oscillator 207
24.6.5Divider and PLL Block 209
24.7Power Management Controller (PMC) 212
24.8.1Description 212
24.9.2Master Clock Controller 212
24.10.3Processor Clock Controller 213
24.11.4USB Clock Controller 213
24.12.5Peripheral Clock Controller 214
24.13.6HClock Controller 214
24.14.7Programmable Clock Output Controller 214
24.15.8Programming Sequence 214
24.16.9Clock Switching Details 220
24.17.10Power Management Controller (PMC) User Interface 224

25 Advanced Interrupt Controller (AIC) 241

25.1Description 241
25.2Block Diagram 242
25.3.1Application Block Diagram 242
25.4.2AIC Detailed Block Diagram 242
25.5I/O Line Description 243
25.6Product Dependencies 243
25.7.1I/O Lines 243
25.8.2Power Management 243
25.9.3Interrupt Sources 243
25.10Functional Description 244
25.11.1Interrupt Source Control 244
25.12.2Interrupt Latencies 246
25.13.3Normal Interrupt 247
25.14.4Interrupt Handlers 248

25.15.5Fast Interrupt 250
25.16.6Protect Mode 252
25.17.7Spurious Interrupt 253
25.18.8General Interrupt Mask 253
25.19Advanced Interrupt Controller (AIC) User Interface 254
25.20.1Base Address 254
25.21.2Register Mapping 254
25.22.3AIC Source Mode Register 255
25.23.4AIC Source Vector Register 256
25.24.5AIC Interrupt Vector Register 256
25.25.6AIC FIQ Vector Register 257
25.26.7AIC Interrupt Status Register 257
25.27.8AIC Interrupt Pending Register 258
25.28.9AIC Interrupt Mask Register 258
25.29.10AIC Core Interrupt Status Register 259
25.30.11AIC Interrupt Enable Command Register 259
25.31.12AIC Interrupt Disable Command Register 260
25.32.13AIC Interrupt Clear Command Register 260
25.33.14AIC Interrupt Set Command Register 261
25.34.15AIC End of Interrupt Command Register 261
25.35.16AIC Spurious Interrupt Vector Register 262
25.36.17AIC Debug Control Register 262
25.37.18AIC Fast Forcing Enable Register 263
25.38.19AIC Fast Forcing Disable Register 263
25.39.20AIC Fast Forcing Status Register 264

26 Debug Unit (DBGU) 265

26.1Description 265
26.2Block Diagram 266
26.3Product Dependencies 267
26.4.1I/O Lines 267
26.5.2Power Management 267
26.6.3Interrupt Source 267
26.7UART Operations 267
26.8.1Baud Rate Generator 267
26.9.2Receiver 268
26.10.3Start Detection and Data Sampling 268
26.11.4Transmitter 270
26.12.5Peripheral Data Controller 271
26.13.6Test Modes 272
26.14.7Debug Communication Channel Support 272
26.15.8Chip Identifier 273
26.16ICE Access Prevention 273
26.17Debug Unit User Interface 274
26.18.1Debug Unit Control Register 275
26.19.2Debug Unit Mode Register 276
26.20.3Debug Unit Interrupt Enable Register 277

26.21.4Debug Unit Interrupt Disable Register 278
26.22.5Debug Unit Interrupt Mask Register 279
26.23.6Debug Unit Status Register 280
26.24.7Debug Unit Receiver Holding Register 282
26.25.8Debug Unit Transmit Holding Register 282
26.26.9Debug Unit Baud Rate Generator Register 283
26.27.10Debug Unit Chip ID Register 284
26.28.11Debug Unit Chip ID Extension Register 287

26.29Debug Unit Force NTRST Register 287

27 Parallel Input/Output Controller (PIO) 289

27.1Description 289

27.2Block Diagram 290

27.3Product Dependencies 291

27.4.1Pin Multiplexing 291
27.5.2External Interrupt Lines 291
27.6.3Power Management 291
27.7.4Interrupt Generation 291

27.8Functional Description 292

27.9.1Pull-up Resistor Control 293
27.10.2I/O Line or Peripheral Function Selection 293
27.11.3Peripheral A or B Selection 293
27.12.4Output Control 293
27.13.5Synchronous Data Output 294
27.14.6Multi Drive Control (Open Drain) 294
27.15.7Output Line Timings 294
27.16.8Inputs 295
27.17.9Input Glitch Filtering 295
27.18.10Input Change Interrupt 296

27.19I/O Lines Programming Example 296

27.20User Interface 297

27.21.1PIO Controller PIO Enable Register 300
27.22.2PIO Controller PIO Disable Register 300
27.23.3PIO Controller PIO Status Register 301
27.24.4PIO Controller Output Enable Register 301
27.25.5PIO Controller Output Disable Register 302
27.26.6PIO Controller Output Status Register 302
27.27.7PIO Controller Input Filter Enable Register 303
27.28.8PIO Controller Input Filter Disable Register 303
27.29.9PIO Controller Input Filter Status Register 304
27.30.10PIO Controller Set Output Data Register 304
27.31.11PIO Controller Clear Output Data Register 305
27.32.12PIO Controller Output Data Status Register 305
27.33.13PIO Controller Pin Data Status Register 306
27.34.14PIO Controller Interrupt Enable Register 306
27.35.15PIO Controller Interrupt Disable Register 307
27.36.16PIO Controller Interrupt Mask Register 307

27.37.17PIO Controller Interrupt Status Register 308
27.38.18PIO Multi-driver Enable Register 308
27.39.19PIO Multi-driver Disable Register 309
27.40.20PIO Multi-driver Status Register 309
27.41.21PIO Pull Up Disable Register 310
27.42.22PIO Pull Up Enable Register 310
27.43.23PIO Pull Up Status Register 311
27.44.24PIO Peripheral A Select Register 311
27.45.25PIO Peripheral B Select Register 312
27.46.26PIO Peripheral A B Status Register 312
27.47.27PIO Output Write Enable Register 313
27.48.28PIO Output Write Disable Register 313
27.49.29PIO Output Write Status Register 314

28 Serial Peripheral Interface (SPI) 315

28.1Description 315

28.2Block Diagram 316

28.3Application Block Diagram 317

28.4Signal Description 317

28.5Product Dependencies 317

28.6.1I/O Lines 317
28.7.2Power Management 317
28.8.3Interrupt 318

28.9Functional Description 318

28.10.1Modes of Operation 318
28.11.2Data Transfer 319
28.12.3Master Mode Operations 320
28.13.4SPI Slave Mode 328

28.14Serial Peripheral Interface (SPI) User Interface 329

28.15.1SPI Control Register 330
28.16.2SPI Mode Register 331
28.17.3SPI Receive Data Register 332
28.18.4SPI Transmit Data Register 334
28.19.5SPI Status Register 335
28.20.6SPI Interrupt Enable Register 337
28.21.7SPI Interrupt Disable Register 338
28.22.8SPI Interrupt Mask Register 339
28.23.9SPI Chip Select Register 340

29 Universal Synchronous Asynchronous Receiver Transmitter (USART) 343

29.1Description 343

29.2Block Diagram 344

29.3Application Block Diagram 345

29.4I/O Lines Description 345

29.5Product Dependencies 346

29.6.1I/O Lines 346
29.7.2Power Management 346
29.8.3Interrupt 346

29.9Functional Description 347

29.10.1Baud Rate Generator 347
29.11.2Receiver and Transmitter Control 352
29.12.3Synchronous and Asynchronous Modes 352
29.13.4ISO7816 Mode 369
29.14.5IrDA Mode 371
29.15.6RS485 Mode 374
29.16.7Modem Mode 375
29.17.8Test Modes 375

29.18USART User Interface 378

29.19.1USART Control Register 379
29.20.2USART Mode Register 381
29.21.3USART Interrupt Enable Register 384
29.22.4USART Interrupt Disable Register 385
29.23.5USART Interrupt Mask Register 386
29.24.6USART Channel Status Register 387
29.25.7USART Receive Holding Register 390
29.26.8USART Transmit Holding Register 390
29.27.9USART Baud Rate Generator Register 391
29.28.10USART Receiver Time-out Register 392
29.29.11USART Transmitter Timeguard Register 392
29.30.12USART FI DI RATIO Register 393
29.31.13USART Number of Errors Register 393
29.32.14USART Manchester Configuration Register 394
29.33.15USART IrDA FILTER Register 395

30 Timer/Counter (TC) 397

30.1Description 397

30.2Block Diagram 398

30.3Pin Name List 399

30.4Product Dependencies 399

30.5.1I/O Lines 399
30.6.2Power Management 399
30.7.3Interrupt 399

30.8Functional Description 400

30.9.1TC Description 400
30.10.216-bit Counter 400
30.11.3Clock Selection 400
30.12.4Clock Control 402
30.13.5TC Operating Modes 402
30.14.6Trigger 402
30.15.7Capture Operating Mode 403
30.16.8Capture Registers A and B 403

30.17.9Trigger Conditions 403
30.18.10Waveform Operating Mode 405
30.19.11Waveform Selection 405
30.20.12External Event/Trigger Conditions 412
30.21.13Output Controller 412
30.22Timer Counter (TC) User Interface 413
30.23.1TC Block Control Register 414
30.24.2TC Block Mode Register 414
30.25.3TC Channel Control Register 415
30.26.4TC Channel Mode Register: Capture Mode 416
30.27.5TC Channel Mode Register: Waveform Mode 418
30.28.6TC Counter Value Register 421
30.29.7TC Register A 422
30.30.8TC Register B 422
30.31.9TC Register C 423
30.32.10TC Status Register 423
30.33.11TC Interrupt Enable Register 425
30.34.12TC Interrupt Disable Register 426
30.35.13TC Interrupt Mask Register 427

31 USB Device Port (UDP) 429

31.1Description 429
31.2Block Diagram 430
31.3Product Dependencies 431
31.4.1I/O Lines 431
31.5.2Power Management 431
31.6.3Interrupt 431
31.7Typical Connection 432
31.8.1USB Device Transceiver 432
31.9.2VBUS Monitoring 432
31.10Functional Description 433
31.11.1USB V2.0 Full-speed Introduction 433
31.12.2Handling Transactions with USB V2.0 Device Peripheral 435
31.13.3Controlling Device States 443
31.14USB Device Port (UDP) User Interface 447
31.15.1UDP Frame Number Register 448
31.16.2UDP Global State Register 449
31.17.3UDP Function Address Register 450
31.18.4UDP Interrupt Enable Register 451
31.19.5UDP Interrupt Disable Register 452
31.20.6UDP Interrupt Mask Register 453
31.21.7UDP Interrupt Status Register 455
31.22.8UDP Interrupt Clear Register 457
31.23.9UDP Reset Endpoint Register 458
31.24.10UDP Endpoint Control and Status Register 459
31.25.11UDP FIFO Data Register 464
31.26.12UDP Transceiver Control Register 465

32 Analog-to-digital Converter (ADC) 467

32.1Description 467

32.2Block Diagram 467

32.3Signal Description 468

32.4Product Dependencies 468

32.5.1Power Management 468

32.6.2Interrupt Sources 468

32.7.3Analog Inputs 468

32.8.4I/O Lines 468

32.9.5Timer Triggers 468

32.10.6Conversion Performances 468

32.11Functional Description 468

32.12.1Analog-to-digital Conversion 468

32.13.2Conversion Reference 469

32.14.3Conversion Resolution 469

32.15.4Conversion Results 470

32.16.5Conversion Triggers 471

32.17.6Sleep Mode and Conversion Sequencer 472

32.18.7ADC Timings 472

32.19Analog-to-digital Converter (ADC) User Interface 473

32.20.1ADC Control Register 474

32.21.2ADC Mode Register 474

32.22.3ADC Channel Enable Register 476

32.23.4ADC Channel Disable Register 476

32.24.5ADC Channel Status Register 477

32.25.6ADC Status Register 477

32.26.7ADC Last Converted Data Register 478

32.27.8ADC Interrupt Enable Register 478

32.28.9ADC Interrupt Disable Register 479

32.29.10ADC Interrupt Mask Register 480

32.30.11ADC Channel Data Register 480

33 AT91CAP7E Electrical Characteristics 481

33.1Absolute Maximum Ratings 481

33.2DC Characteristics 481

33.3Power Consumption 482

33.4.1Power Consumption versus Modes 482

33.532 kHz Crystal Oscillator Characteristics 484

33.612 MHz Main Oscillator Characteristics 485

33.7PLLA Characteristics 486

33.8PLLB Characteristics 486

33.9USB Transceiver Characteristics 487

33.10.1Electrical Characteristics 487

33.11.2Switching Characteristics 487

33.12ADC 489

33.13Timings 490

33.14.1Corner Definition 490
33.15.2Processor Clock 490
33.16.3Maximum Speed of the I/Os 490
33.17.4SMC Timings 491
33.18.5SDRAMC Timings 495
33.19.6SPI 496

34 AT91CAP7E Mechanical Characteristics 499

34.1Thermal Considerations 499

34.2.1Thermal Data 499
34.3.2Junction Temperature 499

34.4Package Drawings 500

35 AT91CAP7E Ordering Information 501

36 Revision History 503

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

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

Product Contact

Web Site
www.atmel.com

Technical Support
CAP@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2007 Atmel Corporation. **All rights reserved.** Atmel®, logo and combinations thereof, and others, are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. ARM®, ARM7TDMI® and Thumb® and others are registered trademarks or trademarks of ARM Ltd. Other terms and product names may be trademarks of others.