

## Features

- Incorporates the ARM926EJ-S™ ARM® Thumb® Processor
  - DSP Instruction Extensions
  - ARM Jazelle® Technology for Java® Acceleration
  - 4 Kbyte Data Cache, 4 Kbyte Instruction Cache, Write Buffer
  - 210 MIPS at 190 MHz
  - Memory Management Unit
  - EmbeddedICE™ In-circuit Emulation, Debug Communication Channel Support
  - Mid-level implementation Embedded Trace Macrocell™
- Multi-layer AHB Bus Matrix for Large Bandwidth Transfers
  - Six 32-bit-layer Matrix
  - Boot Mode Select Option, Remap Command
- One 32-KByte internal ROM, Single-cycle Access at Maximum Speed
- One 64-KByte internal SRAM, Single-cycle Access at Maximum Speed
  - 4 Blocks of 16 Kbytes Configurable in TCM or General-purpose SRAM on the AHB Bus Matrix
  - Single-cycle Accessible on AHB Bus at Bus Speed
  - Single-cycle Accessible on TCM Interface at Processor Speed
- 2-channel DMA
  - Memory to Memory Transfer
  - 16 Bytes FIFO
  - Linked List
- External Bus Interface (EBI)
  - EBI Supports SDRAM, Static Memory, ECC-enabled NAND Flash and CompactFlash®
- LCD Controller (for AT91SAM9RL64 only)
  - Supports Passive or Active Displays
  - Up to 24 Bits per Pixel in TFT Mode, Up to 16 bits per Pixel in STN Color Mode
  - Up to 16M Colors in TFT Mode, Resolution Up to 2048x2048, Virtual Screen Support
- High Speed (480 Mbit/s) USB 2.0 Device Controller
  - On-Chip High Speed Transceiver, UTMI+ Physical Interface
  - Integrated FIFOs and Dedicated DMA
  - 4 Kbyte Configurable Integrated DPRAM
- Fully-featured System Controller, including
  - Reset Controller, Shutdown Controller
  - Four 32-bit Battery Backup Registers for a Total of 16 Bytes
  - Clock Generator and Power Management Controller
  - Advanced Interrupt Controller and Debug Unit
  - Periodic Interval Timer, Watchdog Timer and Real-time Timer and Real-time Clock
- Reset Controller (RSTC)
  - Based on Two Power-on Reset Cells
  - Reset Source Identification and Reset Output Control
- Shutdown Controller (SHDC)
  - Programmable Shutdown Pin Control and Wake-up Circuitry
- Clock Generator (CKGR)
  - Selectable 32768 Hz Low-power Oscillator or Internal Low-power RC Oscillator on Battery Backup Power Supply, Providing a Permanent Slow Clock
  - 12 MHz On-chip Oscillator for Main System Clock and USB Clock



## AT91 ARM Thumb Microcontrollers

AT91SAM9R64  
AT91SAM9RL64

Preliminary

6289A-ATARM-15-Jan-08





- One PLL up to 240 MHz
- One PLL 480 MHz Optimized for USB HS
- Power Management Controller (PMC)
  - Very Slow Clock Operating Mode, Software Programmable Power Optimization Capabilities
  - Two Programmable External Clock Signals
- Advanced Interrupt Controller (AIC)
  - Individually Maskable, Eight-level Priority, Vectored Interrupt Sources
  - One External Interrupt Sources and One Fast Interrupt Source, Spurious Interrupt Protected
- Debug Unit (DBGU)
  - 2-wire UART and Support for Debug Communication Channel
- 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
- Real-time Clock (RTC)
  - Time, Date and Alarm 32-bit Parallel Load
  - Low Power Consumption
  - Programmable Periodic Interrupt
- One 6-channel 10-Bit Analog-to-Digital Converter
  - Touch Screen Interface Compatible with Industry Standard 4-wire Sensitive Touch Panels
- Four 32-bit Parallel Input/Output Controllers (PIOA, PIOB, PIOC and PIOD)
  - 118 Programmable I/O Lines Multiplexed with up to Two Peripheral I/Os for 217-ball BGA Package
  - Input Change Interrupt Capability on Each I/O Line
  - Individually Programmable Open-drain, Pull-up Resistor and Synchronous Output
- 22-channel Peripheral DMA Controller (PDC)
- One MultiMedia Card Interface (MCI)
  - SDCard/SDIO 1.0 and MultiMediaCard™ 3.1 Compliant
  - Automatic Protocol Control and Fast Automatic Data Transfers with PDC
- Two Synchronous Serial Controllers (SSC)
  - Independent Clock and Frame Sync Signals for Each Receiver and Transmitter
  - I<sup>2</sup>S Analog Interface Support, Time Division Multiplex Support
  - High-speed Continuous Data Stream Capabilities with 32-bit Data Transfer
- One AC97 Controller (AC97C)
  - 6-channel Single AC97 Analog Front End Interface, Slot Assigner
- Four Universal Synchronous/Asynchronous Receiver Transmitters (USART)
  - Individual Baud Rate Generator, IrDA® Infrared Modulation/Demodulation, Manchester Encoding/Decoding
  - Support for ISO7816 T0/T1 Smart Card, Hardware Handshaking, RS485 Support
- One Master/Slave Serial Peripheral Interface (SPI)
  - 8- to 16-bit Programmable Data Length, Four External Peripheral Chip Selects
  - High-speed Synchronous Communications
- One Three-channel 16-bit Timer/Counter (TC)
  - Three External Clock Inputs, Two Multi-purpose I/O Pins per Channel
  - Double PWM Generation, Capture/Waveform Mode, Up/Down Capability
- One Four-channel 16-bit PWM Controller (PWMC)
- Two Two-wire Interfaces (TWI)
  - Compatible with Standard Two-wire Serial Memories
  - One, Two or Three Bytes for Slave Address
  - Sequential Read/Write Operations

- Master, Multi-master and Slave Mode Operation
- Bit Rate: Up to 400 Kbits
- General Call Supported in Slave Mode
- Connection to Peripheral DMA Controller (PDC) Channel Capabilities Optimizes Data Transfers in Master Mode Only (TWI0 only)
- SAM-BA™ Boot Assistant
  - Default Boot Program
  - Interface with SAM-BA Graphic User Interface
- IEEE® 1149.1 JTAG Boundary Scan on All Digital Pins
- Required Power Supplies:
  - 1.08 to 1.32V for VDDCORE, VDDUTMIC, VDDPLL B and VDDBU
  - 3.0V to 3.6V for VDDPLLA, VDDANA, VDDUTMII and VDDIOP
  - Programmable 1.65V to 1.95V or 3.0V to 3.6V for VDDIOM
- Available in a 144-ball BGA (AT91SAM9R64) and a 217-ball LFBGA (AT91SAM9RL64) Package

## 1. Description

The AT91SAM9R64/RL64 device is based on the integration of an ARM926EJ-S processor with a large fast SRAM and a wide range of peripherals.

The AT91SAM9R64/RL64 embeds one USB Device High Speed Controller, one LCD Controller (for AT91SAM9RL64 only), one AC97 controller, a 2-channel DMA Controller, four USARTs, two SSCs, one SPI, two TWIs, three Timer Counter channels, a 4-channel PWM generator, one Multimedia Card interface and a 6-channel Analog-to-digital converter that also provides touch screen management.

The AT91SAM9R64/RL64 is architected on a 6-layer bus matrix. It also features an External Bus Interface capable of interfacing with a wide range of memory and peripheral devices.

Some features are not available for AT91SAM9R64 in the 144-ball BGA package.

Separate block diagrams and PIO multiplexing are provided in this document. [Table 1-1](#) lists the features and signals of AT91SAM9RL64 that are not available or partially available for AT91SAM9R64. When the signal is multiplexed on a PIO, the PIO line is specified.

**Table 1-1.** Unavailable or Partially Available Features and Signals in AT91SAM9R64

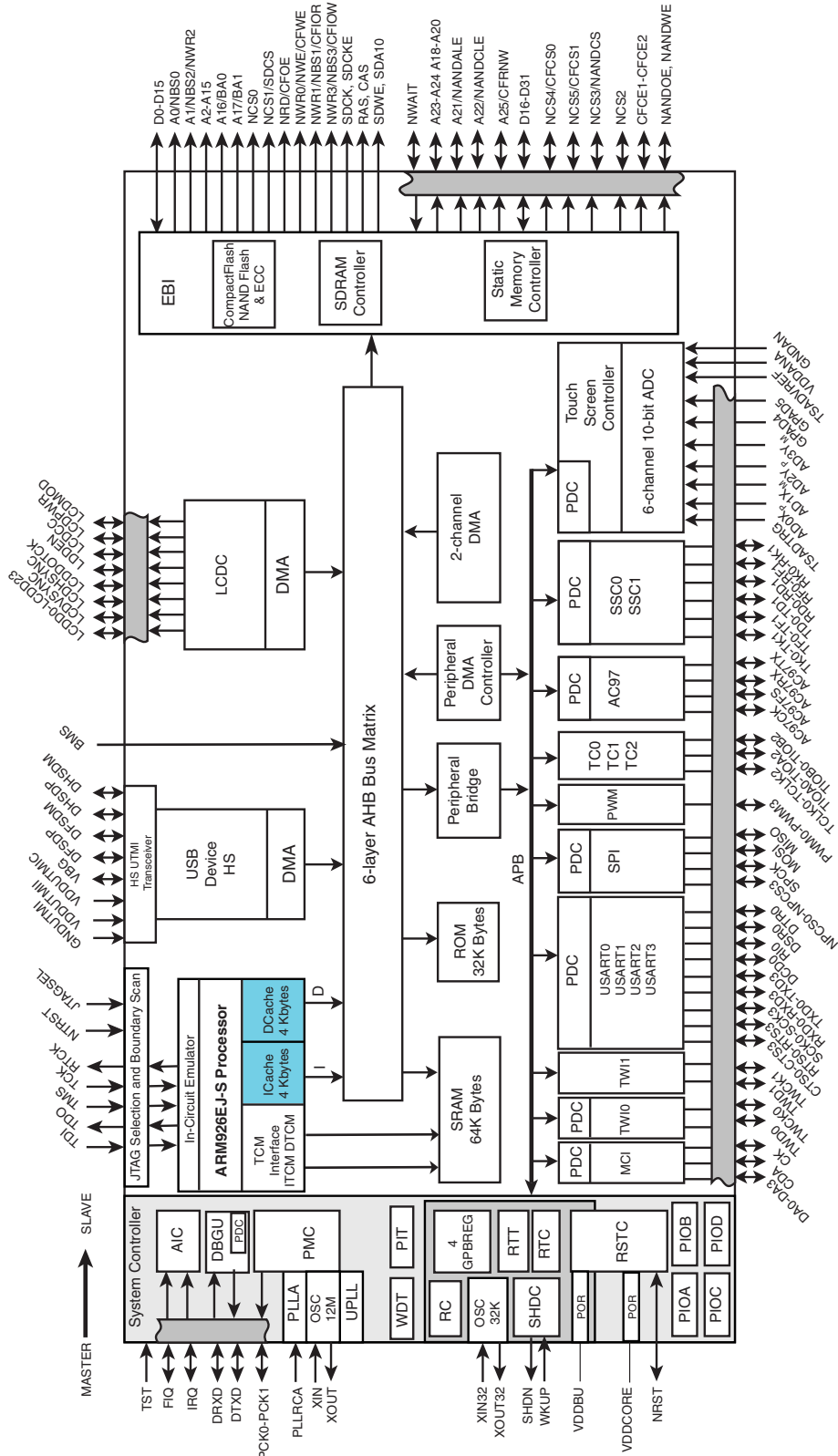
Feature	Full/Partial	Signal	Peripheral A	Peripheral B
AC97	Full	AC97FS AC97CK AC97TX AC97RX	PD1 PD2 PD3 PD4	-
EBI	Partial	D16-D31 NCS2 NCS5/CFCS1	PB16-PB31 PD0 PD13	-
LCDC	Full	LCDMOD LCDCC LCDVSYNC LCDHSYNC LCDDOTCK LCDDEN LCDD0-LCDD23	PC2 PC3 PC4 PC5 PC6 PC7 PC8-PC31	-

**Table 1-1.** Unavailable or Partially Available Features and Signals in AT91SAM9R64

Feature	Full/Partial	Signal	Peripheral A	Peripheral B
PWM	Partial	PWM2	PD5 and PD12	-
SPI	Partial	NPCS2 NPCS3	PD8	PD9 and PD13
SSC1	Full	RF1 RK1 TD1 RD1 TK1 TF1	-	PA8 PA9 PA13 PA14 PA29 PA30
Touchscreen ADC	Partial	AD3YM GPAD4 GPAD5	PA20 PD6 PD7	-
TC	Partial	TIOA1 TIOB1 TCLK1 TIOA2 TIOB2	-	PC29 PC30 PC31 PD10 PD11
TWI	Full	TWD1 TWCK1	PD10 PD11	-
USART0	Partial	SCK0 RTS0 CTS0 DSR0 DTR0 DCD0 RI0	PA8 PA9 PA10 PD14 PD15 PD16 PD17	-
USART1	Partial	SCK1	-	PD2
USART2	Partial	SCK2 RTS2 CTS2	PD9 PA29 PA30	-
USART3	Partial	SCK3 RTS3 CTS3	-	PA20 PD3 PD4



Figure 2-2. AT91SAM9RL64 Block Diagram



## 3. Signal Description

Table 3-1 gives details on the signal name classified by peripheral.

Table 3-1. Signal Description List

Signal Name	Function	Type	Active Level	Comments
<b>Power Supplies</b>				
VDDIOM	EBI I/O Lines Power Supply	Power		1.65V to 3.6V
VDDIOP	Peripherals I/O Lines Power Supply	Power		3.0V to 3.6V
VDDUTMII	USB UTMI+ Interface Power Supply	Power		3.0V to 3.6V
VDDUTMIC	USB UTMI+ Core Power Supply	Power		1.08V to 1.32V
GNDUTMI	USB UTMI Ground	Ground		
VDDDBU	Backup I/O Lines Power Supply	Power		1.08V to 1.32V
GNDDBU	Backup Ground	Ground		
VDDPLLA	PLL Power Supply	Power		3.0V to 3.6V
GNDPLLA	PLL Ground	Ground		
VDDPLLB	UTMI PLL and OSC 12M Power Supply	Power		1.08 V to 1.32V
GNDPLLB	UTMI PLL and OSC 12M Ground	Ground		
VDDANA	ADC Analog Power Supply	Power		3.0V to 3.6V
GNDANA	ADC Analog Ground	Ground		
VDDCORE	Core Chip Power Supply	Power		1.08V to 1.32V
GNDCORE	Ground	Ground		
GND	Ground	Ground		
<b>Clocks, Oscillators and PLLs</b>				
XIN	Main Oscillator Input	Input		
XOUT	Main Oscillator Output	Output		
XIN32	Slow Clock Oscillator Input	Input		
XOUT32	Slow Clock Oscillator Output	Output		
VBG	Bias Voltage Reference	Analog		
PLLCA	PLL A Filter	Input		
PCK0 - PCK1	Programmable Clock Output	Output		
<b>Shutdown, Wakeup Logic</b>				
SHDN	Shutdown Control	Output		Driven at 0V only. Do not tie over VDDDBU
WKUP	Wake-Up Input	Input		Accept between 0V and VDDDBU
<b>ICE and JTAG</b>				
TCK	Test Clock	Input		No pull-up resistor
TDI	Test Data In	Input		No pull-up resistor
TDO	Test Data Out	Output		
TMS	Test Mode Select	Input		No pull-up resistor

**Table 3-1. Signal Description List (Continued)**

Signal Name	Function	Type	Active Level	Comments
JTAGSEL	JTAG Selection	Input		Pull-down resistor
NTRST	Test Reset Signal	Input	Low	Pull-up resistor.
<b>Reset/Test</b>				
NRST	Microcontroller Reset	I/O	Low	Pull-up resistor
TST	Test Mode Select	Input		Pull-down resistor
BMS	Boot Mode Select	Input		Must be connected to GND or VDDIOP.
<b>Debug Unit - DBGU</b>				
DRXD	Debug Receive Data	Input		
DTXD	Debug Transmit Data	Output		
<b>Advanced Interrupt Controller - AIC</b>				
IRQ	External Interrupt Input	Input		
FIQ	Fast Interrupt Input	Input		
<b>PIO Controller - PIOA - PIOB - PIOC-PIOD</b>				
PA0 - PA31	Parallel IO Controller A	I/O		Pulled-up input at reset
PB0 - PB31	Parallel IO Controller B	I/O		Pulled-up input at reset
PC0 - PC31	Parallel IO Controller C	I/O		Pulled-up input at reset
PD0 - PD21	Parallel IO Controller D	I/O		Pulled-up input at reset
<b>External Bus Interface - EBI</b>				
D0 - D31	Data Bus	I/O		Pulled-up input at reset. D16-D31 not present on AT91SAM9R64.
A0 - A25	Address Bus	Output		0 at reset
NWAIT	External Wait Signal	Input	Low	
<b>Static Memory Controller - SMC</b>				
NCS0 - NCS5	Chip Select Lines	Output	Low	NCS2, NCS5 not present on AT91SAM9R64.
NWR0 - NWR3	Write Signal	Output	Low	
NRD	Read Signal	Output	Low	
NWE	Write Enable	Output	Low	
NBS0 - NBS3	Byte Mask Signal	Output	Low	
<b>CompactFlash Support</b>				
CFCE1 - CFCE2	CompactFlash Chip Enable	Output	Low	
CFOE	CompactFlash Output Enable	Output	Low	
CFWE	CompactFlash Write Enable	Output	Low	
CFIOR	CompactFlash IO Read	Output	Low	
CFIOW	CompactFlash IO Write	Output	Low	
CFRNW	CompactFlash Read Not Write	Output		



**Table 3-1.** Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
CFCS0 - CFCS1	CompactFlash Chip Select Lines	Output	Low	CFCS1 not present on AT91SAM9R64.
<b>NAND Flash Support</b>				
NANDCS	NAND Flash Chip Select	Output	Low	
NANDOE	NAND Flash Output Enable	Output	Low	
NANDWE	NAND Flash Write Enable	Output	Low	
<b>SDRAM Controller</b>				
SDCK	SDRAM Clock	Output		
SDCKE	SDRAM Clock Enable	Output	High	
SDCS	SDRAM Controller Chip Select	Output	Low	
BA0 - BA1	Bank Select	Output		
SDWE	SDRAM Write Enable	Output	Low	
RAS - CAS	Row and Column Signal	Output	Low	
SDA10	SDRAM Address 10 Line	Output		
<b>Multimedia Card Interface MCI</b>				
CK	Multimedia Card Clock	I/O		
CDA	Multimedia Card Slot A Command	I/O		
DA0 - DA3	Multimedia Card Slot A Data	I/O		
<b>Universal Synchronous Asynchronous Receiver Transmitter USARTx</b>				
SCKx	USARTx Serial Clock	I/O		SCKx not present on AT91SAM9R64.
TXDx	USARTx Transmit Data	I/O		
RXDx	USARTx Receive Data	Input		
RTSx	USARTx Request To Send	Output		RTS0, RTS2, RTS3 not present on AT91SAM9R64.
CTSx	USARTx Clear To Send	Input		CTS0, CTS2, CTS3 not present on AT91SAM9R64.
DTR0	USART0 Data Terminal Ready	I/O		Not present on AT91SAM9R64.
DSR0	USART0 Data Set Ready	Input		Not present on AT91SAM9R64.
DCD0	USART0 Data Carrier Detect	Output		Not present on AT91SAM9R64.
RI0	USART0 Ring Indicator	Input		Not present on AT91SAM9R64.
<b>Synchronous Serial Controller - SSCx</b>				
TD0 - TD1	SSC Transmit Data	Output		TD1 not present on AT91SAM9R64.
RD0 - RD1	SSC Receive Data	Input		RD1 not present on AT91SAM9R64.
TK0 - TK1	SSC Transmit Clock	I/O		TK1 not present on AT91SAM9R64.
RK0 - RK1	SSC Receive Clock	I/O		RK1 not present on AT91SAM9R64.
TF0 - TF1	SSC Transmit Frame Sync	I/O		TF1 not present on AT91SAM9R64.
RF0 - RF1	SSC Receive Frame Sync	I/O		RF1 not present on AT91SAM9R64.



**Table 3-1.** Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
<b>AC97 Controller - AC97C</b>				
AC97RX	AC97 Receive Signal	Input		Not present on AT91SAM9R64.
AC97TX	AC97 Transmit Signal	Output		Not present on AT91SAM9R64.
AC97FS	AC97 Frame Synchronization Signal	Output		Not present on AT91SAM9R64.
AC97CK	AC97 Clock signal	Input		Not present on AT91SAM9R64.
<b>Timer/Counter - TC</b>				
TCLKx	TC Channel x External Clock Input	Input		TCLK1 not present on AT91SAM9R64.
TIOAx	TC Channel x I/O Line A	I/O		TIOA1, TIOA2 not present on AT91SAM9R64.
TIOBx	TC Channel x I/O Line B	I/O		TIOB1, TIOB2 not present on AT91SAM9R64.
<b>Pulse Width Modulation Controller- PWMC</b>				
PMWx	Pulse Width Modulation Output	Output		PWM2 not present on AT91SAM9R64.
<b>Serial Peripheral Interface - SPI</b>				
MISO	Master In Slave Out	I/O		
MOSI	Master Out Slave In	I/O		
SPCK	SPI Serial Clock	I/O		
NPCS0	SPI Peripheral Chip Select 0	I/O	Low	
NPCS1 - NPCS3	SPI Peripheral Chip Select	Output	Low	NPCS2, NPCS3 not present on AT91SAM9R64.
<b>Two-Wire Interface - TWIx</b>				
TWDx	TWix Two-wire Serial Data	I/O		TWD1 not present on AT91SAM9R64.
TWCKx	TWix Two-wire Serial Clock	I/O		TWCK1 not present on AT91SAM9R64.
<b>Touch Screen Analog-to-Digital Converter</b>				
GPAD0-GPAD5	Analog Inputs	Analog		GPAD4, GPAD5 not present on AT91SAM9R64.
AD0X <sub>P</sub>	Touch Panel Right side	Analog		Multiplexed with AD0
AD1X <sub>M</sub>	Touch Panel Left side	Analog		Multiplexed with AD1
AD2Y <sub>P</sub>	Touch Panel Top side	Analog		Multiplexed with AD2
AD3Y <sub>M</sub>	Touch Panel Bottom side	Analog		Multiplexed with AD3. Not present on AT91SAM9R64.
TSADTRG	ADC Trigger	Input		
TSADVREF	ADC Reference	Analog		
<b>LCD Controller - LCDC</b>				
LCDD0 - LCDD23	LCD Data Bus	Output		Not present on AT91SAM9R64.
LCDVSYNC	LCD Vertical Synchronization	Output		Not present on AT91SAM9R64.
LCDHSYNC	LCD Horizontal Synchronization	Output		Not present on AT91SAM9R64.

**Table 3-1.** Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
LCDDOTCK	LCD Dot Clock	Output		Not present on AT91SAM9R64.
LCDDEN	LCD Data Enable	Output		Not present on AT91SAM9R64.
LCDC	LCD Contrast Control	Output		Not present on AT91SAM9R64.
LCDPWR	LCD panel Power enable control	Output		Not present on AT91SAM9R64.
LCDMOD	LCD Modulation signal	Output		Not present on AT91SAM9R64.
<b>USB High Speed Device</b>				
DFSDM	USB Device Full Speed Data -	Analog		
DFSDP	USB Device Full Speed Data +	Analog		
DHSDM	USB Device High Speed Data -	Analog		
DHSDP	USB Device High Speed Data +	Analog		

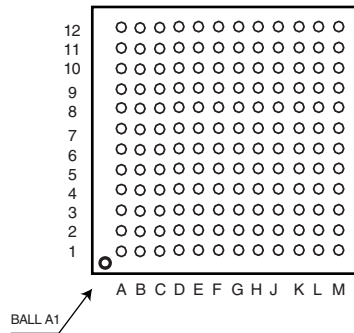
## 4. Package and Pinout

The AT91SAM9R64 is available in a 144-ball BGA package. The AT91SAM9RL64 is available in a 217-ball LFBGA package.

### 4.1 144-ball BGA Package Outline

Figure 4-1 shows the orientation of the 144-ball BGA package.

Figure 4-1. 144-ball BGA Pinout (Top View)



## 4.2 Pinout

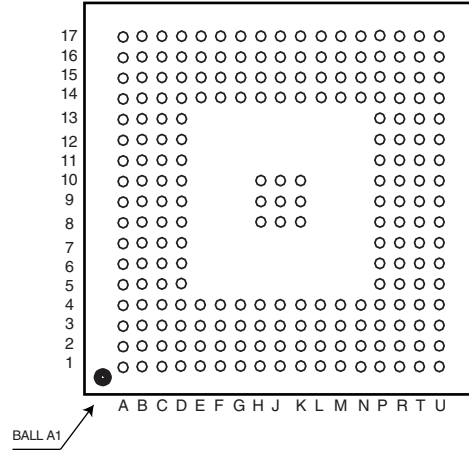
**Table 4-1.** AT91SAM9R64 Pinout for 144-ball BGA Package

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
A1	DFSDM	D1	PLLRC	G1	PB[10]	K1	A[5]
A2	DHSDM	D2	VDDUTMII	G2	PB[11]	K2	A[6]
A3	XIN	D3	NWR3/NBS3/CFIOW	G3	PB[12]	K3	A[13]
A4	XOUT	D4	NWR1/NBS1/CFIOR	G4	PB[9]	K4	A[15]
A5	XIN32	D5	JTAGSEL	G5	PB[13]	K5	RAS
A6	XOUT32	D6	GNDBU	G6	GND	K6	D[3]
A7	TDO	D7	TCK	G7	GND	K7	D[6]
A8	PA[31]	D8	PA[26]	G8	GND	K8	D[13]
A9	PA[22]	D9	PA[24]	G9	GNDUTMI	K9	VDDIOM
A10	PA[16]	D10	PA[13]	G10	VDDCORE	K10	VDDIOM
A11	PA[14]	D11	PA[6]	G11	VDDIOP	K11	D[11]
A12	PA[11]	D12	PD[20]	G12	VDDIOP	K12	PB[1]
B1	DFSDP	E1	GNDPLLA	H1	PB[14]	L1	A[7]
B2	DHSDP	E2	NWR0/NWE/CFWE	H2	PB[15]	L2	A[8]
B3	NC	E3	NRD/CFOE	H3	A[0]	L3	A[11]
B4	VDDPLLB	E4	NCS0	H4	A[2]	L4	A[16]
B5	GNDPLLB	E5	NCS1/SDCS	H5	SDA10	L5	SDWE
B6	TMS	E6	PB[2]	H6	D[1]	L6	D[4]
B7	RTCK	E7	NRST	H7	GND	L7	D[7]
B8	PA[27]	E8	BMS	H8	GND	L8	D[15]
B9	PA[21]	E9	PA[25]	H9	VDDIOM	L9	PC[1]
B10	PA[12]	E10	PA[15]	H10	SDCKE	L10	PC[0]
B11	PD[21]	E11	PA[5]	H11	VDDCORE	L11	PB[0]
B12	PA[10]	E12	PA[4]	H12	VDDIOP	L12	GNDANA
C1	VDDPLLA	F1	PB[5]	J1	A[4]	M1	A[9]
C2	VBG	F2	PB[6]	J2	A[1]	M2	A[10]
C3	VDDBU	F3	PB[7]	J3	A[3]	M3	A[12]
C4	SHDN	F4	PB[8]	J4	A[14]	M4	A[17]
C5	WKUP	F5	PB[3]	J5	CAS	M5	D[0]
C6	NTRST	F6	PB[4]	J6	D[2]	M6	SDCK
C7	TDI	F7	TST	J7	D[5]	M7	D[8]
C8	PA[28]	F8	VDDUTMIC	J8	D[12]	M8	ADVREF
C9	PA[23]	F9	PA[3]	J9	D[14]	M9	VDDANA
C10	PA[7]	F10	PA[2]	J10	VDDIOM	M10	PA[17]
C11	PD[19]	F11	PA[0]	J11	D[10]	M11	PA[18]
C12	PD[18]	F12	PA[1]	J12	D[9]	M12	PA[19]

### 4.3 217-ball LFBGA Package Outline

Figure 4-2 shows the orientation of the 217-ball LFBGA package.

Figure 4-2. 217-ball LFBGA Pinout (Top View)



## 4.4 Pinout

**Table 4-2.** AT91SAM9RL64 Pinout for 217-ball LFBGA Package <sup>(1)</sup>

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
A1	DFSDM	D5	SHDN	J14	PD[1]	P17	PC[11]
A2	DHSDP	D6	JTAGSEL	J15	PD[0]	R1	A[0]
A3	VDDPLLB	D7	NTRST	J16	PC[30]	R2	A[2]
A4	XIN	D8	BMS	J17	PC[31]	R3	A[7]
A5	XOUT	D9	TDO	K1	PB[14]	R4	A[10]
A6	GNDPLLB	D10	PA[30]	K2	PB[15]	R5	A[14]
A7	XOUT32	D11	GND	K3	PB[17]	R6	SDA10
A8	GND	D12	PA[23]	K4	PB[16]	R7	D[0]
A9	NRST	D13	PA[15]	K8	VDDUTMIC	R8	VDDIOM
A10	RTCK	D14	PA[12]	K9	VDDIOP	R9	D[6]
A11	PA[29]	D15	PA[8]	K10	PC[28]	R10	D[9]
A12	PA[26]	D16	PD[13]	K14	PC[25]	R11	NC
A13	PA[22]	D17	PD[16]	K15	PC[24]	R12	VDDIOM
A14	PA[14]	E1	GNDPLLA	K16	PC[26]	R13	PC[1]
A15	PA[10]	E2	NCS1/SDCS	K17	PC[27]	R14	PB[1]
A16	PD[20]	E3	NCS0	L1	PB[18]	R15	PC[5]
A17	PD[17]	E4	NWR3/NBS3/CFIOW	L2	PB[19]	R16	PC[6]
B1	DFSDP	E14	PD[15]	L3	PB[21]	R17	PC[7]
B2	DHSDM	E15	PD[14]	L4	PB[20]	T1	A[3]
B3	VBG	E16	PA[5]	L14	PC[21]	T2	A[5]
B4	NC	E17	PA[4]	L15	PC[20]	T3	A[8]
B5	NC	F1	NRD/CFOE	L16	PC[22]	T4	A[12]
B6	XIN32	F2	PB[2]	L17	PC[23]	T5	A[16]
B7	TST	F3	NWRO/NWE/CFWE	M1	PB[22]	T6	RAS
B8	GND	F4	PB[3]	M2	PB[23]	T7	D[2]
B9	TMS	F14	PA[1]	M3	PB[25]	T8	D[4]
B10	VDDCORE	F15	PA[0]	M4	PB[24]	T9	D[7]
B11	PA[28]	F16	PA[2]	M14	PC[17]	T10	D[10]
B12	PA[25]	F17	PA[3]	M15	PC[16]	T11	D[14]
B13	PA[21]	G1	GND	M16	PC[18]	T12	VDDANA
B14	PA[13]	G2	VDDIOM	M17	PC[19]	T13	PA[17]
B15	PD[21]	G3	PB[5]	N1	PB[26]	T14	PA[19]
B16	PD[19]	G4	PB[4]	N2	PB[27]	T15	PC[2]
B17	PA[9]	G14	PD[12]	N3	PB[29]	T16	PC[3]
C1	VDDPLLA	G15	PD[11]	N4	PB[28]	T17	PC[4]
C2	VDDUTMII	G16	PD[10]	N14	PC[13]	U1	A[4]
C3	GND	G17	PD[9]	N15	PC[12]	U2	A[6]
C4	GNDUTMI	H1	PB[8]	N16	PC[14]	U3	A[9]
C5	VDDBU	H2	PB[9]	N17	PC[15]	U4	A[13]
C6	WKUP	H3	PB[7]	P1	PB[30]	U5	A[17]
C7	GNDDBU	H4	PB[6]	P2	PB[31]	U6	SDWE
C8	TCK	H8	VDDCORE	P3	A[1]	U7	D[3]
C9	TDI	H9	VDDIOP	P4	A[11]	U8	SDCK
C10	PA[31]	H10	PD[4]	P5	A[15]	U9	D[11]
C11	PA[27]	H14	PD[8]	P6	CAS	U10	D[12]
C12	PA[24]	H15	PD[5]	P7	D[1]	U11	D[13]
C13	PA[16]	H16	PD[2]	P8	SDCKE	U12	TSADVREF
C14	PA[11]	H17	PD[3]	P9	D[5]	U13	PA[18]
C15	PD[18]	J1	PB[12]	P10	D[8]	U14	PA[20]
C16	PA[7]	J2	PB[13]	P11	D[15]	U15	PD[6]
C17	PA[6]	J3	PB[11]	P12	PC[0]	U16	PD[7]
D1	PLLRCAL	J4	PB[10]	P13	PB[0]	U17	GNDANA
D2	NWR1/NBS1/CFIOR	J8	VDDCORE	P14	PC[8]		
D3	GND	J9	VDDIOP	P15	PC[9]		
D4	GND	J10	PC[29]	P16	PC[10]		

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



## 5. Power Considerations

### 5.1 Power Supplies

The AT91SAM9R64/RL64 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.
- VDDIOM pins: Power the External Bus Interface; voltage ranges between 1.65V and 1.95V (1.8V nominal) or between 3.0V and 3.6V (3.3V nominal).
- VDDIOP pins: Power the Peripherals I/O lines; voltage ranges from 3.0V and 3.6V, 3.3V nominal.
- VDDDBU pin: Powers the Slow Clock oscillator and a part of the System Controller; voltage ranges from 1.08V and 1.32V, 1.2V nominal.
- VDDPLLA pin: Powers the PLL cell; voltage ranges from 3.0V and 3.6V, 3.3V nominal.
- VDDPLLB pin: Powers the UTMI PLL (480MHz) and OSC 12M cells; voltage ranges from 1.08V and 1.32V, 1.2V nominal.
- VDDUTMII pin: Powers the UTMI+ interface; voltage ranges from 3.0V and 3.6V, 3.3V nominal.
- VDDUTMIC pin: Powers the UTMI+ core; voltage ranges from 1.08V and 1.32V, 1.2V nominal.
- VDDANA pin: Powers the ADC cell; voltage ranges from 3.0V and 3.6V, 3.3V nominal.

The power supplies VDDIOM and VDDIOP are identified in the pinout table and the PIO multiplexing tables. These supplies enable the user to power the device differently for interfacing with memories and for interfacing with peripherals.

Ground pins GND are common to VDDCORE, VDDIOM and VDDIOP pins power supplies.

Separated ground pins are provided for VDDDBU, VDDPLLA, VDDPLLB and VDDANA. These ground pins are respectively GNDDBU, GNDPLLA, GNDPLLB and GNDANA. A common ground pin is provided for VDDUTMII and VDDUTMIC. This ground pin is GNDUTMI.

### 5.2 Power Consumption

The AT91SAM9R64/RL64 consumes about 500  $\mu$ A of static current on VDDCORE at 25°C and up to 5 mA at 85°C.

On VDDDBU, the current does not exceed 5  $\mu$ A @25°C and 20  $\mu$ A @85°C.

For dynamic power consumption, the AT91SAM9R64/RL64 consumes a maximum of 70 mA on VDDCORE in worst case conditions (1.2V, 85°C, processor running full-performance algorithm).

### 5.3 Programmable I/O Lines Power Supplies

The power supplies pins VDDIOM support two voltage ranges. This allows the device to reach its maximum speed either out of 1.8V or 3.3V external memories.

The maximum speed is MCK on the pin SDCK (SDRAM Clock) loaded with 30pF for power supply at 1.8V and 50 pF for power supply at 3.3V.

The maximum speed on the other signals of the External Bus Interface (control, address and data signals) is 50 MHz.



The voltage ranges are determined by programming registers in the Chip Configuration registers located in the Matrix User Interface.

At reset, the selected voltage defaults to 3.3V nominal and power supply pins can accept either 1.8V or 3.3V. The user must make sure to program the EBI voltage range before getting the device out of its Slow Clock Mode.

The PIO lines are supplied through VDDIOP and the speed of the signal that can be driven on them can reach 50 MHz with 50 pF load.

## 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 is an output, driven at up to VDDIOP, 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 k $\Omega$  to GNDBU, so that it can be left unconnected for normal operations.

All the JTAG signals are supplied with VDDIOP except JTAGSEL supplied by VDDBU.

### 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 k $\Omega$  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 VDDIOP.

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

The NRST and NTRST pins integrates a permanent pull-up resistor of 100 k $\Omega$  typical to VDDIOP.

The NRST signal is inserted in the Boundary Scan.

### 6.4 PIO Controllers

All the I/O lines which are managed by the PIO Controllers integrate a programmable pull-up resistor. Refer to the section “AT91SAM9R64/RL64 Electrical Characteristics” in the product datasheet for more details.

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 require to be enabled as Peripheral at reset. This is explicitly indicated in the column “Reset State” of the PIO Controller multiplexing tables.

## 6.5 Shutdown Logic Pins

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

The pin WKUP is an input-only. It can accept voltages only between 0V and VDDBU.

## 7. Processor and Architecture

### 7.1 ARM926EJ-S Processor

- RISC Processor Based on ARM v5TEJ Architecture with Jazelle technology for Java acceleration
- Two Instruction Sets
  - ARM High-performance 32-bit Instruction Set
  - Thumb High Code Density 16-bit Instruction Set
- DSP Instruction Extensions
- 5-Stage Pipeline Architecture:
  - Instruction Fetch (F)
  - Instruction Decode (D)
  - Execute (E)
  - Data Memory (M)
  - Register Write (W)
- 4-Kbyte Data Cache, 4-Kbyte Instruction Cache
  - Virtually-addressed 4-way Associative Cache
  - Eight words per line
  - Write-through and Write-back Operation
  - Pseudo-random or Round-robin Replacement
- Write Buffer
  - Main Write Buffer with 16-word Data Buffer and 4-address Buffer
  - DCache Write-back Buffer with 8-word Entries and a Single Address Entry
  - Software Control Drain
- Standard ARM v4 and v5 Memory Management Unit (MMU)
  - Access Permission for Sections
  - Access Permission for large pages and small pages can be specified separately for each quarter of the page
  - 16 embedded domains
- Bus Interface Unit (BIU)
  - Arbitrates and Schedules AHB Requests
  - Separate Masters for both instruction and data access providing complete Matrix system flexibility
  - Separate Address and Data Buses for both the 32-bit instruction interface and the 32-bit data interface
  - On Address and Data Buses, data can be 8-bit (Bytes), 16-bit (Half-words) or 32-bit (Words)

## 7.2 Matrix Masters

The Bus Matrix of the AT91SAM9R64/RL64 product manages 6 masters, which means that each master can perform an access concurrently with others, to an available slave.

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 decodings.

**Table 7-1.** List of Bus Matrix Masters

Master 0	DMA Controller
Master 1	USB Device High Speed DMA
Master 2	LCD Controller DMA
Master 3	Peripheral DMA Controller
Master 4	ARM926™ Instruction
Master 5	ARM926 Data

## 7.3 Matrix Slaves

The Bus Matrix of the AT91SAM9R64/RL64 product manages 6 slaves. Each slave has its own arbiter, allowing a different arbitration per slave.

**Table 7-2.** List of Bus Matrix Slaves

Slave 0	Internal ROM
Slave 1	Internal SRAM
Slave 2	LCD Controller User Interface
Slave 3	USB HS Device User Interface
Slave 4	External Bus Interface (EBI)
Slave 5	Peripheral Bridge

## 7.4 Master to Slave Access

All the Masters can normally access all the Slaves. However, some paths do not make sense, for example allowing access from the USB Device High speed DMA to the Internal Peripherals. Thus, these paths are forbidden or simply not wired, and shown as “-” in the following table.

**Table 7-3.** AT91SAM9R64/RL64 Master to Slave Access

Masters		0	1	2	3	4	5
Slaves		DMA Controller	USB HS Device DMA	LCD Controller DMA	Peripheral DMA	ARM926 Instruction	ARM926 Data
0	Internal ROM	X	X		X	X	X
1	Internal SRAM	X	X	X	X	X	X
2	LCD Controller User Interface	-	-	-	-	X	X
3	USB HS Device User Interface	-	-	-	-	X	X
4	External Bus Interface	X	X	X	X	X	X
5	Peripheral Bridge	X	X	X	-	-	-

## 7.5 Peripheral DMA Controller (PDC)

- Acting as one AHB Bus Matrix Master
- Allows data transfers from/to peripheral to/from any memory space without any intervention of the processor.
- Next Pointer support, prevents strong real-time constraints on buffer management.

The Peripheral DMA Controller handles transfer requests from the channel according to the following priorities (Low to High priorities):

- a. TWI0 Transmit Channel
- b. DBGU Transmit Channel
- c. USART3 Transmit Channel
- d. USART2 Transmit Channel
- e. USART1 Transmit Channel
- f. USART0 Transmit Channel
- g. AC97 Transmit Channel
- h. SPI Transmit Channel
- i. SSC1 Transmit Channel
- j. SSC0 Transmit Channel
- k. TWI0 Receive Channel
- l. DBGU Receive Channel
- m. ADC Receive Channel
- n. USART3 Receive Channel
- o. USART2 Receive Channel
- p. USART1 Receive Channel
- q. USART0 Receive Channel
- r. AC97 Receive Channel
- s. SPI Receive Channel
- t. SSC1 Receive Channel
- u. SSC0 Transmit Channel
- v. MCI Receive/Transmit Channel

## 7.6 DMA Controller

- Acting as one Matrix Master
- Embeds 2 channels
- 16 bytes/FIFO for Channel Buffering
- Linked List support with Status Write Back operation at End of Transfer
- Word, Half-word, Byte transfer support

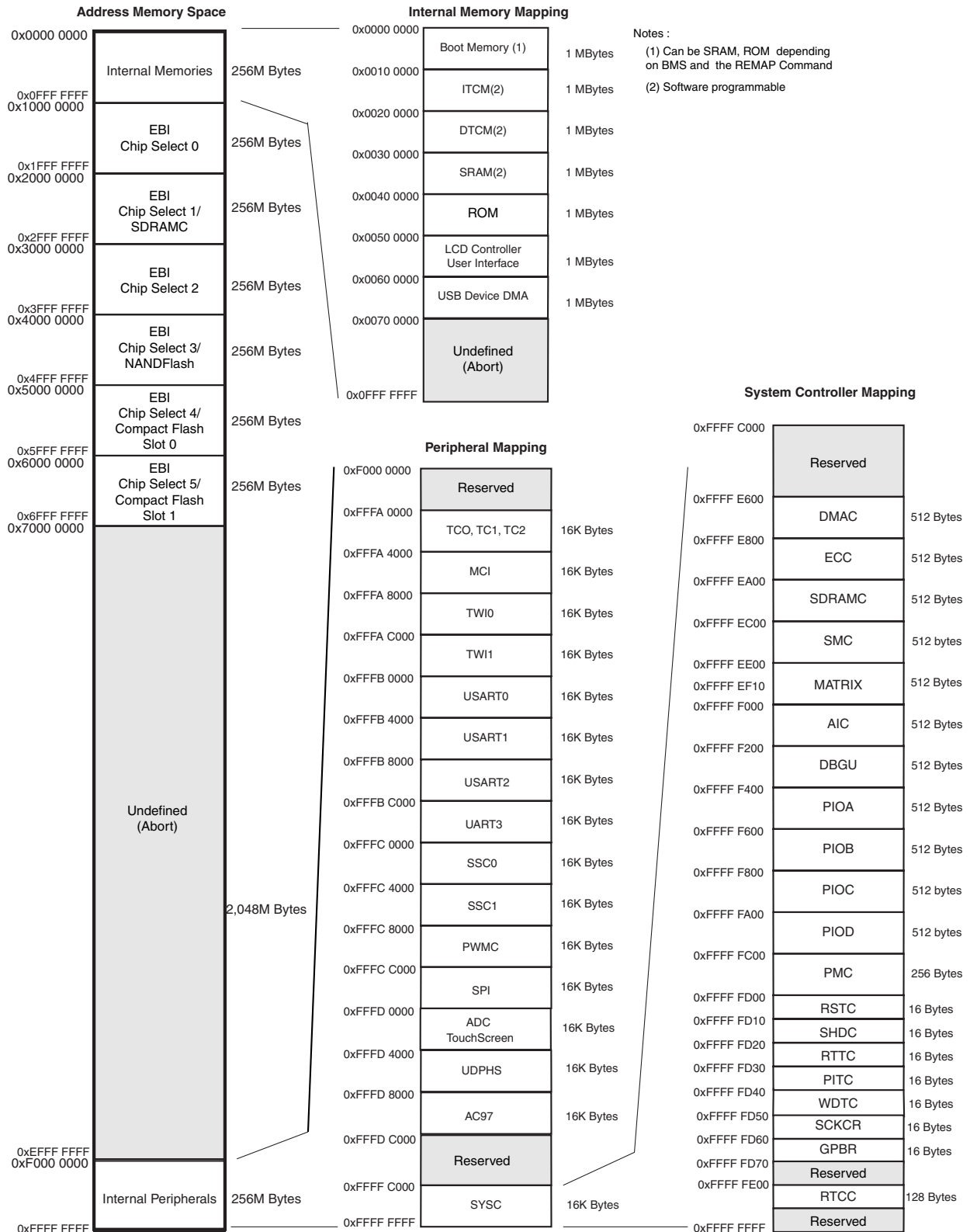
## 7.7 Debug and Test Features

- ARM926 Real-time In-circuit Emulator
  - Two real-time Watchpoint Units
  - Two Independent Registers: Debug Control Register and Debug Status Register
  - Test Access Port Accessible through JTAG Protocol

- Debug Communications Channel
- Debug Unit
  - Two-pin UART
  - Debug Communication Channel Interrupt Handling
  - Chip ID Register
- IEEE1149.1 JTAG Boundary-scan on All Digital Pins

# 8. Memories

Figure 8-1. AT91SAM9R64/RL64 Memory Mapping



A first level of address decoding is performed by the AHB 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 8 are directed to the EBI that associates these banks to the external chip selects EBI\_NCS0 to EBI\_NCS5. 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.

## 8.1 Embedded Memories

- 32 KB ROM
  - Single Cycle Access at full bus speed
- 64 KB Fast SRAM
  - Single Cycle Access at full bus speed
  - Supports ARM926EJ-S TCM interface at full processor speed

### 8.1.1 Internal Memory Mapping

[Table 8-1](#) summarizes the Internal Memory Mapping for each Master, depending on the Remap status (RCBx bit) and the BMS state at reset.

**Table 8-1.** Internal Memory Mapping

Address	RCBx <sup>(1)</sup> = 0		RCBx <sup>(1)</sup> = 1
	BMS = 1	BMS = 0	
0x0000 0000	ROM	EBI_NCS0 <sup>(2)</sup>	SRAM

- Notes:
1. x = 0 to maximum Master number.
  2. EBI NCS0 is to be connected to a 16-bit non-volatile memory. The access configuration is defined by the reset state of SMC Setup, SMC Pulse, SMC Cycle and SMC Mode CS0 registers.

#### 8.1.1.1 Internal SRAM

The AT91SAM9R64/RL64 product embeds a total of 64Kbyte high-speed SRAM split in 4 blocks of 16KBytes.

After reset and until the Remap Command is performed, the SRAM is only accessible at address 0x0030 0000.

After Remap, the SRAM also becomes available at address 0x0.

This Internal SRAM can be allocated to three areas. Its Memory Mapping is detailed in [Table 8-2](#).

- Internal SRAM A is the ARM926EJ-S Instruction TCM. The user can map this SRAM block anywhere in the ARM926 instruction memory space using CP15 instructions and the TCR configuration register located in the Chip Configuration User Interface. This SRAM block is also accessible by the ARM926 Data Master and by the AHB Masters through the AHB bus at address 0x0010 0000.

- Internal SRAM B is the ARM926EJ-S Data TCM. The user can map this SRAM block anywhere in the ARM926 data memory space using CP15 instructions. This SRAM block is also accessible by the ARM926 Data Master and by the AHB Masters through the AHB bus at address 0x0020 0000.
- Internal SRAM C is only accessible by all the AHB Masters. After reset and until the Remap Command is performed, this SRAM block is accessible through the AHB bus at address 0x0030 0000 by all the AHB Masters. After Remap, this SRAM block also becomes accessible through the AHB bus at address 0x0 by the ARM926 Instruction and the ARM926 Data Masters.

Within the 64Kbyte SRAM size available, the amount of memory assigned to each block is software programmable as a multiple of 16K Bytes according to [Table 8-2](#). This Table provides the size of the Internal SRAM C according to the size of the Internal SRAM A and the Internal SRAM B.

**Table 8-2.** Internal SRAM Block Size

Remaining Internal SRAM C		Internal SRAM A (ITCM) Size		
		0	16K Bytes	32K Bytes
Internal SRAM B (DTCM) size	0	64K Bytes	48K Bytes	32K Bytes
	16K Bytes	48K Bytes	32K Bytes	16K Bytes
	32K Bytes	32K Bytes	16K Bytes	0K Bytes

At reset, the whole memory is assigned to Internal SRAM C.

The memory blocks assigned to SRAM A, SRAM B and SRAM C areas are not contiguous and when the user dynamically changes the Internal SRAM configuration, the new 16-Kbyte block organization may affect the previous configuration from a software point of view.

[Table 8-3](#) illustrates different configurations and the related 16-Kbyte blocks (RB0 to RB3) assignments.

**Table 8-3.** 16-Kbyte Block Allocation example

Decoded Area	Address	Configuration examples and related 16-Kbyte block assignments								
		I = 0K D = 0K A = 64K <sup>(1)</sup>	I = 16K D = 0K A = 48K	I = 32K D = 0K A = 32K	I = 0K D = 16K A = 48K	I = 16K D = 16K A = 32K	I = 32K D = 16K A = 16K	I = 0K D = 32K A = 32K	I = 16K D = 32K A = 16K	I = 32K D = 32K A = 0K
Internal SRAM A (ITCM)	0x0010 0000		RB1	RB1		RB1	RB1		RB1	RB1
	0x0010 4000			RB0			RB0			RB0
Internal SRAM B (DTCM)	0x0020 0000				RB3	RB3	RB3	RB3	RB3	RB3
	0x0020 4000							RB2	RB2	RB2
Internal SRAM C (AHB)	0x0030 0000	RB3	RB3	RB3	RB2	RB2	RB2	RB1	RB0	
	0x0030 4000	RB2	RB2	RB2	RB1	RB0		RB0		
	0x0030 8000	RB1	RB0		RB0					
	0x0030 C000	RB0								

Note: 1. Configuration after reset.



When accessed from the AHB, the internal Fast SRAM is single cycle accessible at full matrix speed (MCK). When accessed from the processor's TCM Interface, they are also single cycle accessible at full processor speed.

## 8.1.1.2 Internal ROM

The AT91SAM9R64/RL64 embeds an Internal ROM, which contains the SAM-BA program.

At any time, the ROM is mapped at address 0x0040 0000. It is also accessible at address 0x0 (BMS =1) after the reset and before the Remap Command.

## 8.1.2 Boot Strategies

The system always boots at address 0x0. To ensure maximum boot possibilities, the memory layout can be changed with two parameters.

REMAP allows the user to layout the internal SRAM bank to 0x0 to ease the development. This is done by software once the system has boot. Refer to the Bus Matrix Section for more details.

When REMAP = 0 BMS allows the user to lay out to 0x0, at his convenience, the ROM or an external memory. This is done by a hardware way at reset.

Note: All the memory blocks can always be seen at their specified base addresses that are not concerned by these parameters.

The AT91SAM9R64/RL64 Bus Matrix manages a boot memory that depends on the level on the pin BMS at reset. The internal memory area mapped between address 0x0 and 0x000F FFFF is reserved to this effect.

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

If BMS is detected at 0, the boot memory is the memory connected on the Chip Select 0 of the External Bus Interface.

### 8.1.2.1 BMS = 1, boot on embedded ROM

The system boots on Boot Program.

- Boot on on-chip RC
- Enable the 32768 Hz oscillator
- Auto baudrate detection
- Downloads and runs an application from external storage media into internal SRAM
- Downloaded code size depends on embedded SRAM size
- Automatic detection of valid application
- Bootloader on a non-volatile memory
  - SDCard
  - NANDFlash
  - SPI DataFlash<sup>®</sup> connected on NPCS0 of the SPI0
- SAM-BA Boot in case no valid program is detected in external NVM, supporting
  - Serial communication on a DBGU
  - USB Device HS Port

### 8.1.2.2 BMS = 0, boot on external memory

- Boot on on-chip RC

- Boot with the default configuration for the Static Memory Controller, byte select mode, 16-bit data bus, Read/Write controlled by Chip Select, allows boot on 16-bit non-volatile memory.

For optimization purposes, nothing else is done. To speed up the boot sequence user programmed software should perform a complete configuration:

- Enable the 32768 Hz oscillator if best accuracy needed
- Program the PMC (main oscillator enable or bypass mode)
- Program and Start the PLL
- Reprogram the SMC setup, cycle, hold, mode timings registers for CS0 to adapt them to the new clock
- Switch the main clock to the new value

## 8.2 External Memories

The AT91SAM9R64/RL64 features one External Bus Interface to offer interface to a wide range of external memories and to any parallel peripheral.

### 8.2.1 External Bus Interface

- Integrates three External Memory Controllers:
  - Static Memory Controller
  - SDRAM Controller
  - SLC Nand Flash ECC 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 (SDCS) 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<sup>M</sup> support

### 8.2.2 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
  - 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

## 8.2.3 SDRAM Controller

- Supported devices:
  - Standard and Low Power SDRAM (Mobile SDRAM)
  - **2K, 4K, 8K Row Address Memory Parts**
  - **SDRAM with two or four Internal Banks**
  - **SDRAM with 16- or 32-bit Data Path**
- Programming facilities
  - **Word, half-word, byte access**
  - **Automatic page break when Memory Boundary has been reached**
  - **Multibank Ping-pong Access**
  - **Timing parameters specified by software**
  - **Automatic refresh operation, refresh rate is programmable**
- **Energy-saving capabilities**
  - **Self-refresh, power down and deep power down modes supported**
- Error detection
  - **Refresh Error Interrupt**
- **SDRAM Power-up Initialization by software**
- **SDRAM CAS Latency of 1, 2 and 3 supported**
- **Auto Precharge Command not used**

## 8.2.4 NAND Flash Error Corrected Code Controller

- Tracking the accesses to a NAND Flash device by triggering on the corresponding chip select
- Single bit error correction and 2-bit Random detection.
- Automatic Hamming Code Calculation while writing
  - ECC value available in a register
- Automatic Hamming Code Calculation while reading
  - Error Report, including error flag, correctable error flag and word address being detected erroneous
  - Support 8- or 16-bit NAND Flash devices with 512-, 1024-, 2048- or 4096-bytes pages

## 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 embeds also the registers allowing to configure the Matrix and a set of registers configuring the EBI chip select assignment and the voltage range for external memories.

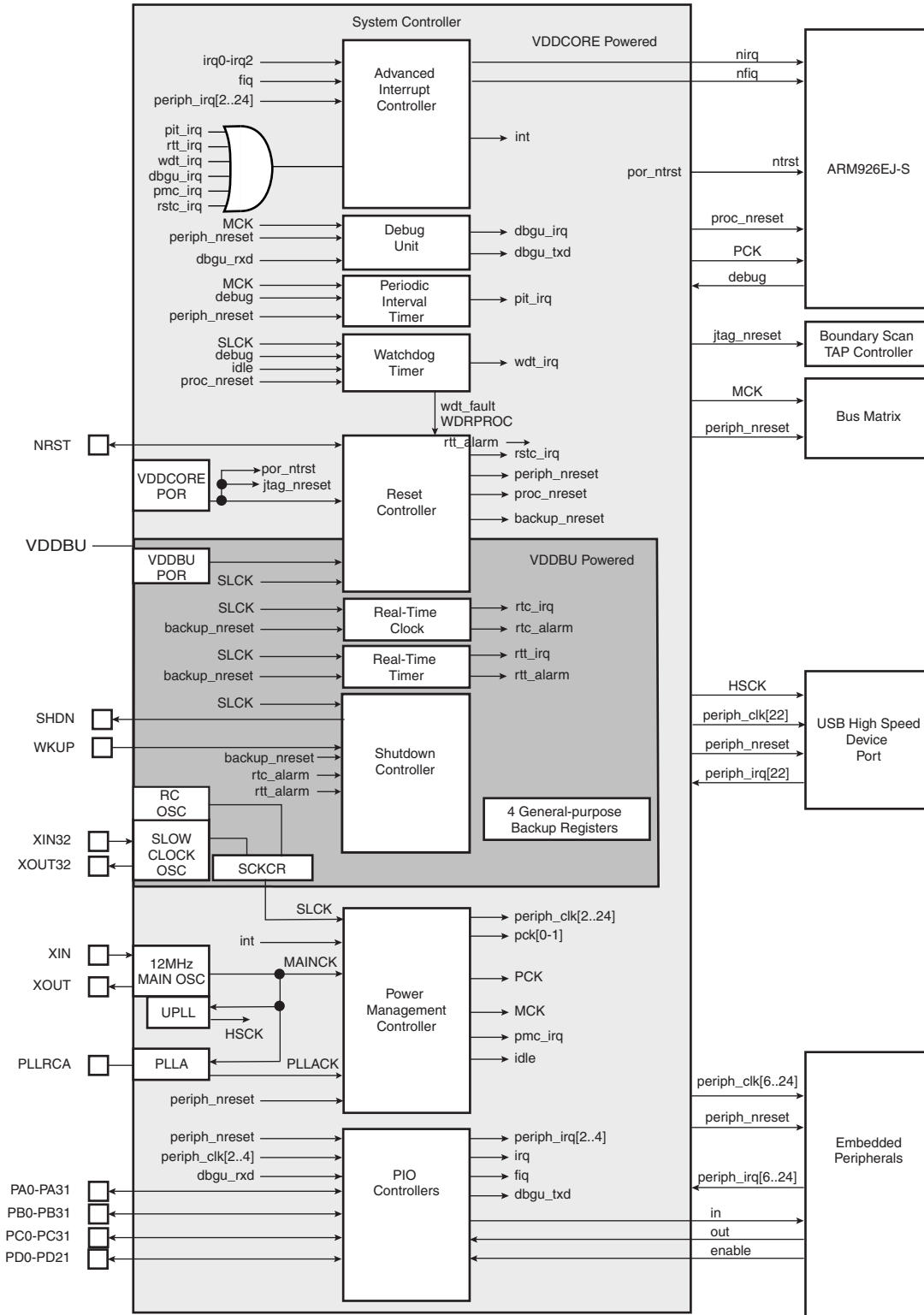
### 9.1 System Controller Mapping

As shown in [Figure 8-1](#), the System Controller's peripherals are all mapped within the highest 16K bytes of the 4 Gbyte 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, as the Load/Store instruction have an indexing mode of +/- 4kbytes.

## 9.2 Block Diagram

Figure 9-1. System Controller Block Diagram



### 9.3 Reset Controller

The Reset Controller is based on two Power-on-Reset cells, one on VDDDBU and one on VDDCORE.

The Reset Controller is capable to return to the software the source of the last reset, either a general reset (VDDDBU rising), a wake-up reset (VDDCORE rising), a software reset, a user reset or a watchdog reset.

The Reset Controller controls the internal resets of the system and the NRST pin output. It is capable to shape a reset signal for the external devices, simplifying to a minimum connection of a push-button on the NRST pin to implement a manual reset.

The configuration of the Reset Controller is saved as supplied on VDDDBU.

### 9.4 Shutdown Controller

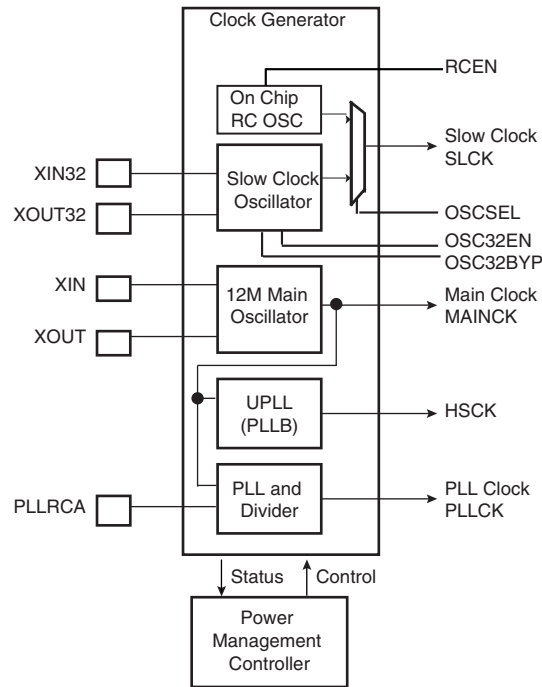
The Shutdown Controller is supplied on VDDDBU and allows a software-controllable shut down of the system through the pin SHDN. An input change of the WKUP pin or an alarm releases the SHDN pin, and thus wakes up the system power supply.

### 9.5 Clock Generator

The Clock Generator is made up of:

- One low-power 32768 Hz Slow Clock Oscillator with bypass mode
- One low-power RC oscillator
- One 12 MHz Main Oscillator, which can be bypassed
- One 480 MHz PLL (UPLL or PLLB) providing a clock for the USB High Speed Device Controller
- One 80 to 240 MHz programmable PLL, providing the PLL Clock (PLLCK). This PLL has an input divider to offer a wider range of output frequencies from the 12 MHz input, the only limitation being the lowest input frequency shall be higher or equal to 1 MHz.

**Figure 9-2.** Clock Generator Block Diagram



## 9.6 Slow Clock Selection

### 9.6.1 Description

The AT91SAM9R64/RL64 slow clock can be generated either by an external 32768Hz crystal or the on-chip RC oscillator. The 32768Hz crystal oscillator can be bypassed to accept an external slow clock on XIN32.

Configuration is located in the slow clock control register (SCKCR) located at address 0xFFFFD50 in the backed up part of the system controller and so is preserved while VDDBU is present.

Refer to the “Clock Generator” section for more details.

## 9.7 Power Management Controller

The Power Management Controller provides all the clock signals to the system. It provides:

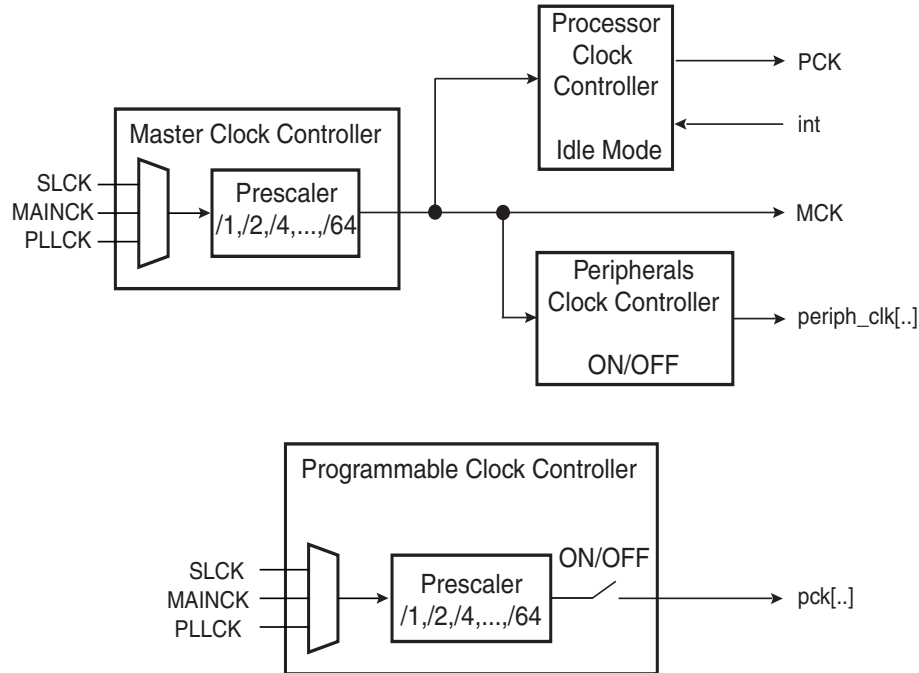
- the Processor Clock PCK
- the Master Clock MCK, in particular to the Matrix and the memory interfaces
- the USB Device HS Clock HSK
- independent peripheral clocks, typically at the frequency of MCK
- two programmable clock outputs: PCK0 and PCK1

This allows the software control of 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 9-3.** AT91SAM9R64/RL64 Power Management Controller Block Diagram



## 9.8 Periodic Interval Timer

- Includes a 20-bit Periodic Counter, with less than 1  $\mu$ s accuracy
- Includes a 12-bit Interval Overlay Counter
- Real Time OS or Linux<sup>®</sup>/WindowsCE<sup>®</sup> compliant tick generator

## 9.9 Watchdog Timer

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

## 9.10 Real-Time Timer

- Real-Time Timer, allowing backup of time with different accuracies
  - 32-bit Free-running back-up Counter
  - Integrates a 16-bit programmable prescaler running on slow clock
  - Alarm Register capable to generate a wake-up of the system through the Shut Down Controller

## 9.11 Real-Time Clock

- Low power consumption
- Full asynchronous design



- Two hundred year calendar
- Programmable Periodic Interrupt
- Alarm and update parallel load
- Control of alarm and update Time/Calendar Data In

## 9.12 General-Purpose Backed-up Registers

- Four 32-bit backup general-purpose registers

## 9.13 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
- One 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

## 9.14 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

## 9.15 Chip Identification

- Chip ID: 0x019B03A0
- JTAG ID: 0x05B2003F
- ARM926 TAP ID: 0x0792603F

## 9.16 PIO Controllers

- 4 PIO Controllers, PIOA, PIOB, PIOC and PIOD, controlling a maximum of 118 I/O Lines
- Each PIO Controller controls up to 32 programmable I/O Lines
  - PIOA has 32 I/O Lines
  - PIOB has 32 I/O Lines
  - PIOC has 32 I/O Lines
  - PIOD has 22 I/O Lines
- Fully programmable through Set/Clear Registers
- Multiplexing of two peripheral functions per I/O Line
- For each I/O Line (whether assigned to a peripheral or used as general purpose I/O)
  - Input change interrupt
  - Glitch filter
  - Multi-drive option enables driving in open drain
  - Programmable pull up on each I/O line
  - Pin data status register, supplies visibility of the level on the pin at any time
- Synchronous output, provides Set and Clear of several I/O lines in a single write

# 10. Peripherals

## 10.1 Peripheral Mapping

As shown in [Figure 8-1](#), the Peripherals are mapped in the upper 256M bytes of the address space between the addresses 0xFFFA 0000 and 0xFFFC FFFF.

Each User Peripheral is allocated 16K bytes of address space.

## 10.2 Peripheral Identifiers

The [Table 10-1](#) defines the Peripheral Identifiers of the AT91SAM9R64/RL64. 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.** AT91SAM9R64/RL64 Peripheral Identifiers

Peripheral ID	Peripheral Mnemonic	Peripheral Name	External Interrupt
0	AIC	Advanced Interrupt Controller	FIQ
1	SYSC	System Controller Interrupt	
2	PIOA	Parallel I/O Controller A,	
3	PIOB	Parallel I/O Controller B	
4	PIOC	Parallel I/O Controller C	
5	PIOD	Parallel I/O Controller D	
6	US0	USART 0	
7	US1	USART 1	
8	US2	USART 2	
9	US3	USART 3	
10	MCI	Multimedia Card Interface	
11	TWI0	Two-Wire Interface 0	
12	TWI1	Two-Wire Interface 1	
13	SPI	Serial Peripheral Interface	
14	SSC0	Synchronous Serial Controller 0	
15	SSC1	Synchronous Serial Controller 1	
16	TC0	Timer Counter 0	
17	TC1	Timer Counter 1	
18	TC2	Timer Counter 2	
19	PWMC	Pulse Width Modulation Controller	
20	TSADCC	Touch Screen ADC Controller	
21	DMAC	DMA Controller	
22	UDPHS	USB Device High Speed	
23	LCDC	LCD Controller (AT91SAM9RL64 only)	
24	AC97	AC97 Controller	
25-30	-	Reserved	
31	AIC	Advanced Interrupt Controller	IRQ

Note: Setting AIC, SYSIRQ, LCDC and IRQ bits in the clock set/clear registers of the PMC has no effect.

## 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 Real-time Clock
- 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 signal IRQ, use a dedicated Peripheral ID. However, there is no clock control associated with these peripheral IDs.

## 10.4 Peripherals Signals Multiplexing on I/O Lines

The AT91SAM9R64/RL64 features 4 PIO controllers, PIOA, PIOB, PIOC and PIOD, which multiplexes the I/O lines of the peripheral set.

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

Note that some peripheral functions which are output only, might be duplicated within the both tables.

The column “Reset State” indicates whether the PIO Line resets in I/O mode or in peripheral mode. If I/O is mentioned, the PIO Line resets in input with the pull-up enabled, so that the device is maintained in a static state as soon as the 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 mentioned in the “Reset State” column, the PIO Line is assigned to this function and the corresponding bit in PIO\_PSR resets high. This is the case for pins controlling memories, 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.

The AT91SAM9RL64 and AT91SAM9R64 do not have the same peripheral signal multiplexing, each one follows.

## 10.4.1 AT91SAM9RL64 PIO Multiplexing

### 10.4.1.1 AT91SAM9RL64 PIO Controller A Multiplexing

**Table 10-2.** AT91SAM9RL64 Multiplexing on PIO Controller A

PIO Controller A					Application Usage	
I/O Line	Peripheral A	Peripheral B	Reset State	Power Supply	Function	Comments
PA0	MC_DA0		I/O	VDDIOP		
PA1	MC_CDA		I/O	VDDIOP		
PA2	MC_CK		I/O	VDDIOP		
PA3	MC_DA1	TCLK0	I/O	VDDIOP		
PA4	MC_DA2	TIOA0	I/O	VDDIOP		
PA5	MC_DA3	TIOB0	I/O	VDDIOP		
PA6	TXD0		I/O	VDDIOP		
PA7	RXD0		I/O	VDDIOP		
PA8	SCK0	RF1	I/O	VDDIOP		
PA9	RTS0	RK1	I/O	VDDIOP		
PA10	CTS0	RK0	I/O	VDDIOP		
PA11	TXD1		I/O	VDDIOP		
PA12	RXD1		I/O	VDDIOP		
PA13	TXD2	TD1	I/O	VDDIOP		
PA14	RXD2	RD1	I/O	VDDIOP		
PA15	TD0		I/O	VDDIOP		
PA16	RD0		I/O	VDDIOP		
PA17	AD0		I/O	VDDANA		
PA18	AD1	RTS1	I/O	VDDANA		
PA19	AD2	CTS1	I/O	VDDANA		
PA20	AD3	SCK3	I/O	VDDANA		
PA21	DRXD		I/O	VDDIOP		
PA22	DTXD	RF0	I/O	VDDIOP		
PA23	TWD0		I/O	VDDIOP		
PA24	TWCK0		I/O	VDDIOP		
PA25	MISO		I/O	VDDIOP		
PA26	MOSI		I/O	VDDIOP		
PA27	SPCK		I/O	VDDIOP		
PA28	NPCS0		I/O	VDDIOP		
PA29	RTS2	TF1	I/O	VDDIOP		
PA30	CTS2	TK1	I/O	VDDIOP		
PA31	NWAIT	IRQ	I/O	VDDIOP		

10.4.1.2 AT91SAM9RL64 PIO Controller B Multiplexing

**Table 10-3.** AT91SAM9RL64 Multiplexing on PIO Controller B

PIO Controller B					Application Usage	
I/O Line	Peripheral A	Peripheral B	Reset State	Power Supply	Function	Comments
PB0	TXD3		I/O	VDDIOP		
PB1	RXD3		I/O	VDDIOP		
PB2	A21/NANDALE		A21	VDDIOM		
PB3	A22/NANDCLE		A22	VDDIOM		
PB4	NANDOE		I/O	VDDIOM		
PB5	NANDWE		I/O	VDDIOM		
PB6	NCS3/NANDCS		I/O	VDDIOM		
PB7	NCS4/CFCS0	NPCS1	I/O	VDDIOM		
PB8	CFE1	PWM0	I/O	VDDIOM		
PB9	CFE2	PWM1	I/O	VDDIOM		
PB10	A25/CFRNW	FIQ	A25	VDDIOM		
PB11	A18		A18	VDDIOM		
PB12	A19		A19	VDDIOM		
PB13	A20		A20	VDDIOM		
PB14	A23	PCK0	A23	VDDIOM		
PB15	A24	ADTRG	A24	VDDIOM		
PB16	D16		I/O	VDDIOM		
PB17	D17		I/O	VDDIOM		
PB18	D18		I/O	VDDIOM		
PB19	D19		I/O	VDDIOM		
PB20	D20		I/O	VDDIOM		
PB21	D21		I/O	VDDIOM		
PB22	D22		I/O	VDDIOM		
PB23	D23		I/O	VDDIOM		
PB24	D24		I/O	VDDIOM		
PB25	D25		I/O	VDDIOM		
PB26	D26		I/O	VDDIOM		
PB27	D27		I/O	VDDIOM		
PB28	D28		I/O	VDDIOM		
PB29	D29		I/O	VDDIOM		
PB30	D30		I/O	VDDIOM		
PB31	D31		I/O	VDDIOM		

## 10.4.1.3 AT91SAM9RL64 PIO Controller C Multiplexing

**Table 10-4.** AT91SAM9RL64 Multiplexing on PIO Controller C

PIO Controller C				Application Usage		
I/O Line	Peripheral A	Peripheral B	Reset State	Power Supply	Function	Comments
PC0	TF0		I/O	VDDIOP		
PC1	TK0	LCDPWR	I/O	VDDIOP		
PC2	LCDMOD	PWM0	I/O	VDDIOP		
PC3	LCDC	PWM1	I/O	VDDIOP		
PC4	LCDVSYNC		I/O	VDDIOP		
PC5	LCDHSYNC		I/O	VDDIOP		
PC6	LCDDOTCK		I/O	VDDIOP		
PC7	LCDDEN		I/O	VDDIOP		
PC8	LCDD0	LCDD2	I/O	VDDIOP		
PC9	LCDD1	LCDD3	I/O	VDDIOP		
PC10	LCDD2	LCDD4	I/O	VDDIOP		
PC11	LCDD3	LCDD5	I/O	VDDIOP		
PC12	LCDD4	LCDD6	I/O	VDDIOP		
PC13	LCDD5	LCDD7	I/O	VDDIOP		
PC14	LCDD6	LCDD10	I/O	VDDIOP		
PC15	LCDD7	LCDD11	I/O	VDDIOP		
PC16	LCDD8	LCDD12	I/O	VDDIOP		
PC17	LCDD9	LCDD13	I/O	VDDIOP		
PC18	LCDD10	LCDD14	I/O	VDDIOP		
PC19	LCDD11	LCDD15	I/O	VDDIOP		
PC20	LCDD12	LCDD18	I/O	VDDIOP		
PC21	LCDD13	LCDD19	I/O	VDDIOP		
PC22	LCDD14	LCDD20	I/O	VDDIOP		
PC23	LCDD15	LCDD21	I/O	VDDIOP		
PC24	LCDD16	LCDD22	I/O	VDDIOP		
PC25	LCDD17	LCDD23	I/O	VDDIOP		
PC26	LCDD18		I/O	VDDIOP		
PC27	LCDD19		I/O	VDDIOP		
PC28	LCDD20		I/O	VDDIOP		
PC29	LCDD21	TIOA1	I/O	VDDIOP		
PC30	LCDD22	TIOB1	I/O	VDDIOP		
PC31	LCDD23	TCLK1	I/O	VDDIOP		



10.4.1.4 AT91SAM9RL64 PIO Controller D Multiplexing

**Table 10-5.** AT91SAM9RL64 Multiplexing on PIO Controller D

PIO Controller D					Application Usage		
I/O Line	Peripheral A	Peripheral B	Comments	Reset State	Power Supply	Function	Comments
PD0	NCS2			I/O	VDDIOP		
PD1	AC97_FS			I/O	VDDIOP		
PD2	AC97_CK	SCK1		I/O	VDDIOP		
PD3	AC97_TX	CTS3		I/O	VDDIOP		
PD4	AC97_RX	RTS3		I/O	VDDIOP		
PD5	DTXD	PWM2		I/O	VDDIOP		
PD6	AD4			I/O	VDDANA		
PD7	AD5			I/O	VDDANA		
PD8	NPCS2	PWM3		I/O	VDDIOP		
PD9	SCK2	NPCS3		I/O	VDDIOP		
PD10	TWD1	TIOA2		I/O	VDDIOP		
PD11	TWCK1	TIOB2		I/O	VDDIOP		
PD12	PWM2	PCK1		I/O	VDDIOP		
PD13	NCS5/CFCS1	NPCS3		I/O	VDDIOP		
PD14	DSR0	PWM0		I/O	VDDIOP		
PD15	DTR0	PWM1		I/O	VDDIOP		
PD16	DCD0	PWM2		I/O	VDDIOP		
PD17	RI0			I/O	VDDIOP		
PD18	PWM3			I/O	VDDIOP		
PD19	PCK0			I/O	VDDIOP		
PD20	PCK1			I/O	VDDIOP		
PD21	TCLK2			I/O	VDDIOP		



## 10.4.2 AT91SAM9R64 PIO Multiplexing

Note: In Table 10-6, Table 10-7, Table 10-8 and Table 10-9, shaded cells indicate I/O lines that are NOT available on the AT91SAM9R64.

### 10.4.2.1 AT91SAM9R64 PIO Controller A Multiplexing

**Table 10-6.** AT91SAM9R64 Multiplexing on PIO Controller A

PIO Controller A					Application Usage	
I/O Line	Peripheral A	Peripheral B	Reset State	Power Supply	Function	Comments
PA0	MC_DA0		I/O	VDDIOP		
PA1	MC_CDA		I/O	VDDIOP		
PA2	MC_CK		I/O	VDDIOP		
PA3	MC_DA1	TCLK0	I/O	VDDIOP		
PA4	MC_DA2	TIOA0	I/O	VDDIOP		
PA5	MC_DA3	TIOB0	I/O	VDDIOP		
PA6	TXD0		I/O	VDDIOP		
PA7	RXD0		I/O	VDDIOP		
PA8	NA	NA				Reserved
PA9	NA	NA				Reserved
PA10	CTS0	RK0	I/O	VDDIOP		
PA11	TXD1		I/O	VDDIOP		
PA12	RXD1		I/O	VDDIOP		
PA13	TXD2		I/O	VDDIOP		
PA14	RXD2		I/O	VDDIOP		
PA15	TD0		I/O	VDDIOP		
PA16	RD0		I/O	VDDIOP		
PA17	AD0		I/O	VDDIOP		
PA18	AD1	RTS1	I/O	VDDIOP		
PA19	AD2	CTS1	I/O	VDDIOP		
PA20	NA	NA				Reserved
PA21	DRXD		I/O	VDDIOP		
PA22	DTXD	RF0	I/O	VDDIOP		
PA23	TWD0		I/O	VDDIOP		
PA24	TWCK0		I/O	VDDIOP		
PA25	MISO		I/O	VDDIOP		
PA26	MOSI		I/O	VDDIOP		
PA27	SPCK		I/O	VDDIOP		
PA28	NPCS0		I/O	VDDIOP		
PA29	NA	NA				Reserved
PA30	NA	NA				Reserved
PA31	NWAIT	IRQ	I/O	VDDIOP		

10.4.2.2 AT91SAM9R64 PIO Controller B Multiplexing

**Table 10-7.** AT91SAM9R64 Multiplexing on PIO Controller B

PIO Controller B				Application Usage		
I/O Line	Peripheral A	Peripheral B	Reset State	Power Supply	Function	Comments
PB0	TXD3		I/O	VDDIOP		
PB1	RXD3		I/O	VDDIOP		
PB2	A21/NANDALE		A21	VDDIOM		
PB3	A22/NANDCLE		A22	VDDIOM		
PB4	NANDOE		I/O	VDDIOM		
PB5	NANDWE		I/O	VDDIOM		
PB6	NCS3/NANDCS		I/O	VDDIOM		
PB7	NCS4/CFCS0	NPCS1	I/O	VDDIOM		
PB8	CFE1	PWM0	I/O	VDDIOM		
PB9	CFE2	PWM1	I/O	VDDIOM		
PB10	A25/CFRNW	FIQ	A25	VDDIOM		
PB11	A18		A18	VDDIOM		
PB12	A19		A19	VDDIOM		
PB13	A20		A20	VDDIOM		
PB14	A23	PCK0	A23	VDDIOM		
PB15	A24	ADTRG	A24	VDDIOM		
PB16- PB31	NA	NA				Reserved

## 10.4.2.3 AT91SAM9R64 PIO Controller C Multiplexing

**Table 10-8.** AT91SAM9R64 Multiplexing on PIO Controller C

PIO Controller C				Application Usage		
I/O Line	Peripheral A	Peripheral B	Reset State	Power Supply	Function	Comments
PC0	TF0		I/O	VDDIOP		
PC1	TK0		I/O	VDDIOP		
PC2-PC31	NA	NA				Reserved

## 10.4.2.4 AT91SAM9R64 PIO Controller D Multiplexing

**Table 10-9.** AT91SAM9R64 Multiplexing on PIO Controller D

PIO Controller D				Application Usage			
I/O Line	Peripheral A	Peripheral B	Comments	Reset State	Power Supply	Function	Comments
PD0-PD17	NA	NA					Reserved
PD18	PWM3			I/O	VDDIOP		
PD19	PCK0			I/O	VDDIOP		
PD20	PCK1			I/O	VDDIOP		
PD21	TCLK2			I/O	VDDIOP		

## 11. Embedded Peripherals Overview

### 11.1 Serial Peripheral Interface (SPI)

- 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

### 11.2 Two-wire Interface (TWI)

- Compatibility with standard two-wire serial memory
- One, two or three bytes for slave address
- Sequential read/write operations
- Supports either master or slave modes
- Compatible with Standard Two-wire Serial Memories
- Master, Multi-master and Slave Mode Operation
- Bit Rate: Up to 400 Kbits
- General Call Supported in Slave mode
- Connection to Peripheral DMA Controller (PDC) Channel Capabilities Optimizes Data Transfers in Master Mode Only
  - One Channel for the Receiver, One Channel for the Transmitter
  - Next Buffer Support

### 11.3 USART

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

- Optional break generation and detection
- By 8 or by-16 over-sampling receiver frequency
- Hardware handshaking RTS-CTS
- Receiver time-out and transmitter timeguard
- 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

## 11.4 Serial Synchronous Controller (SSC)

- Provides serial synchronous communication links used in audio and telecom applications (with CODECs in Master or Slave Modes, I<sup>2</sup>S, TDM Buses, Magnetic Card Reader, etc.)
- Contains an independent receiver and transmitter and a common clock divider
- Offers a configurable frame sync and data length
- Receiver and transmitter can be programmed to start automatically or on detection of different event on the frame sync signal
- Receiver and transmitter include a data signal, a clock signal and a frame synchronization signal

## 11.5 AC97 Controller

- Compatible with AC97 Component Specification V2.2
- Capable to Interface with a Single Analog Front end
- Three independent RX Channels and three independent TX Channels
  - One RX and one TX channel dedicated to the AC97 Analog Front end control
  - One RX and one TX channel for data transfers, associated with a PDC
  - One RX and one TX channel for data transfers with no PDC
- Time Slot Assigner allowing to assign up to 12 time slots to a channel
- Channels support mono or stereo up to 20 bit sample length
  - Variable sampling rate AC97 Codec Interface (48KHz and below)

## 11.6 Timer Counter (TC)

- 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

## 11.7 Pulse Width Modulation Controller (PWM)

- 4 channels, one 16-bit counter per channel
- Common clock generator, providing Thirteen Different Clocks
  - A Modulo n counter providing eleven clocks
  - Two independent Linear Dividers working on modulo n counter outputs
- Independent channel programming
  - Independent Enable Disable Commands
  - Independent Clock Selection
  - Independent Period and Duty Cycle, with Double Bufferization
  - Programmable selection of the output waveform polarity
  - Programmable center or left aligned output waveform

## 11.8 Multimedia Card Interface (MCI)

- Compatibility with MultiMedia Card Specification Version 3.31
- Compatibility with SD Memory Card Specification Version 1.0
- Compatibility with SDIO Specification Version V1.0.
- Cards clock rate up to Master Clock divided by 2
- Embedded power management to slow down clock rate when not used
- MCI has one slot supporting
  - One MultiMediaCard bus (up to 30 cards) or
  - One SD Memory Card
  - One SDIO Card
- Support for stream, block and multi-block data read and write

## 11.9 USB High Speed Device Port (UDPHS)

- USB V2.0 high-speed compliant, 480 Mbits per second
- Embedded USB V2.0 UTMI+ high-speed transceiver
- Embedded 4K-byte dual-port RAM for endpoints
- Embedded 6 channels DMA controller
- Suspend/Resume logic
- Up to 3 banks for isochronous and bulk endpoints
- Seven endpoints:

- Endpoint 0: 64 bytes, 1 bank mode
- Endpoint 1 & 2: 1024 bytes, 2 banks mode, HS isochronous capable, DMA
- Endpoint 3 & 4: 1024bytes, 3 banks mode, DMA
- Endpoint 5 & 6: 1024 bytes, 3 banks mode, HS isochronous capable, DMA

## 11.10 LCD Controller (LDC)

- Single and Dual scan color and monochrome passive STN LCD panels supported
- Single scan active TFT LCD panels supported.
- 4-bit single scan, 8-bit single or dual scan, 16-bit dual scan STN interfaces supported
- Up to 24-bit single scan TFT interfaces supported
- Up to 16 gray levels for mono STN and up to 4096 colors for color STN displays
- 1, 2 bits per pixel (palletized), 4 bits per pixel (non-palletized) for mono STN
- 1, 2, 4, 8 bits per pixel (palletized), 16 bits per pixel (non-palletized) for color STN
- 1, 2, 4, 8 bits per pixel (palletized), 16, 24 bits per pixel (non-palletized) for TFT
- Single clock domain architecture
- Resolution supported up to 2048 x 2048

## 11.11 Touch Screen Analog-to-digital Converter (TSADC)

- 6-channel ADC
- Support 4-wire resistive Touch Screen
- 10-bit 384 Ksamples/sec. Successive Approximation Register ADC
- -3/+3 LSB Integral Non Linearity, -2/+2 LSB Differential Non Linearity
- Integrated 6-to-1 multiplexer, offering eight independent 3.3V analog inputs
- External voltage reference for better accuracy on low voltage inputs
- Individual enable and disable of each channel
- Multiple trigger sources
  - Hardware or software trigger
  - External trigger pin
  - Timer Counter 0 to 2 outputs TIOA0 to TIOA2 trigger
- Sleep Mode and conversion sequencer
  - Automatic wakeup on trigger and back to sleep mode after conversions of all enabled channels





## 12. ARM926EJ-S Processor Overview

### 12.1 Overview

The ARM926EJ-S processor is a member of the ARM9™ family of general-purpose microprocessors. The ARM926EJ-S implements ARM architecture version 5TEJ and is targeted at multi-tasking applications where full memory management, high performance, low die size and low power are all important features.

The ARM926EJ-S processor supports the 32-bit ARM and 16-bit THUMB instruction sets, enabling the user to trade off between high performance and high code density. It also supports 8-bit Java instruction set and includes features for efficient execution of Java bytecode, providing a Java performance similar to a JIT (Just-In-Time compilers), for the next generation of Java-powered wireless and embedded devices. It includes an enhanced multiplier design for improved DSP performance.

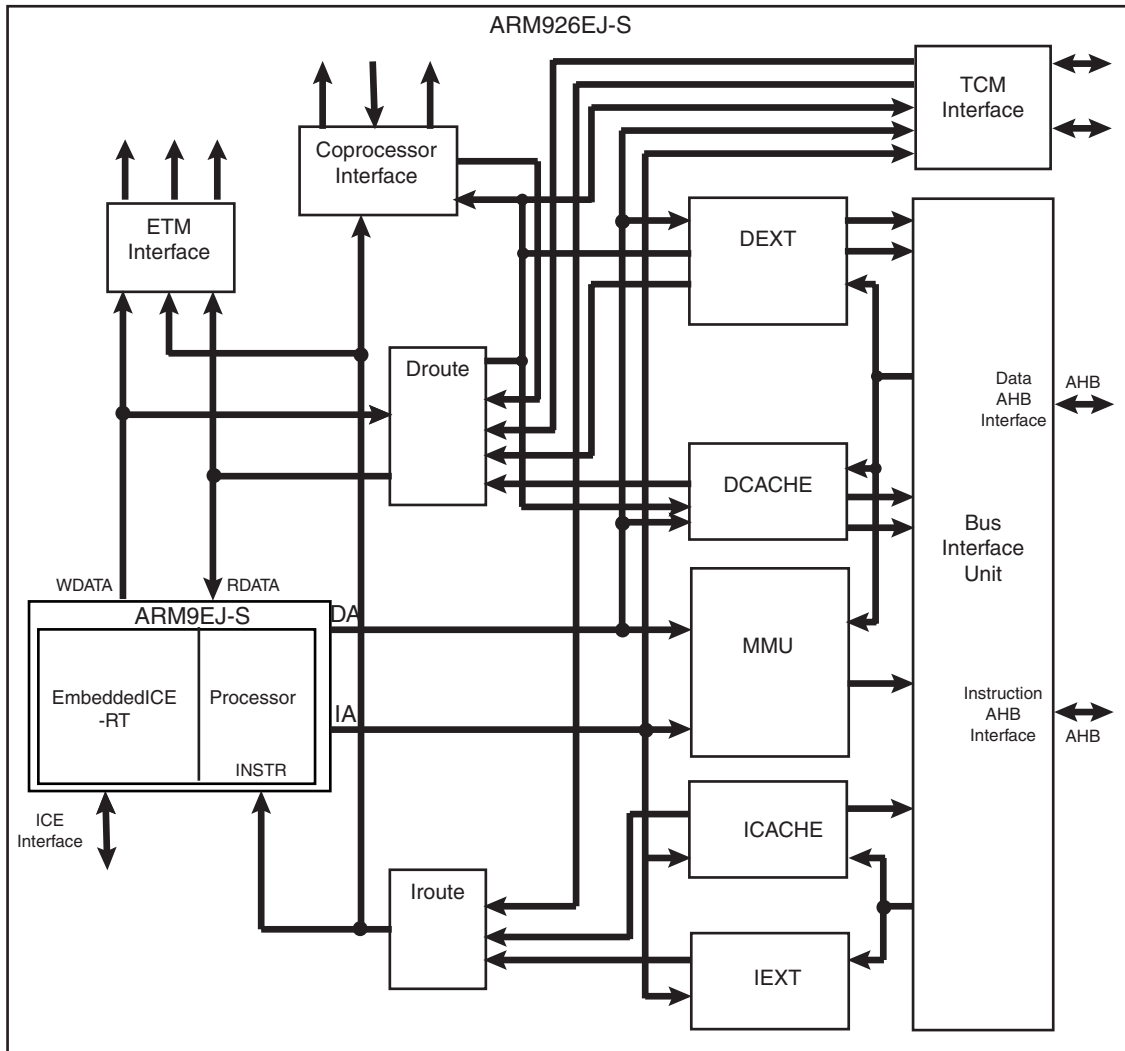
The ARM926EJ-S processor supports the ARM debug architecture and includes logic to assist in both hardware and software debug.

The ARM926EJ-S provides a complete high performance processor subsystem, including:

- an ARM9EJ-S™ integer core
- a Memory Management Unit (MMU)
- separate instruction and data AMBA™ AHB bus interfaces
- separate instruction and data TCM interfaces

## 12.2 Block Diagram

Figure 12-1. ARM926EJ-S Internal Functional Block Diagram



## 12.3 ARM9EJ-S Processor

### 12.3.1 ARM9EJ-S Operating States

The ARM9EJ-S processor can operate in three different states, each with a specific instruction set:

- ARM state: 32-bit, word-aligned ARM instructions.
- THUMB state: 16-bit, halfword-aligned Thumb instructions.
- Jazelle state: variable length, byte-aligned Jazelle instructions.

In Jazelle state, all instruction Fetches are in words.

### 12.3.2 Switching State

The operating state of the ARM9EJ-S core can be switched between:

- ARM state and THUMB state using the BX and BLX instructions, and loads to the PC

- ARM state and Jazelle state using the BXJ instruction

All exceptions are entered, handled and exited in ARM state. If an exception occurs in Thumb or Jazelle states, the processor reverts to ARM state. The transition back to Thumb or Jazelle states occurs automatically on return from the exception handler.

### 12.3.3 Instruction Pipelines

The ARM9EJ-S core uses two kinds of pipelines to increase the speed of the flow of instructions to the processor.

A five-stage (five clock cycles) pipeline is used for ARM and Thumb states. It consists of Fetch, Decode, Execute, Memory and Writeback stages.

A six-stage (six clock cycles) pipeline is used for Jazelle state. It consists of Fetch, Jazelle/Decode (two clock cycles), Execute, Memory and Writeback stages.

### 12.3.4 Memory Access

The ARM9EJ-S core supports byte (8-bit), half-word (16-bit) and word (32-bit) access. Words must be aligned to four-byte boundaries, half-words must be aligned to two-byte boundaries and bytes can be placed on any byte boundary.

Because of the nature of the pipelines, it is possible for a value to be required for use before it has been placed in the register bank by the actions of an earlier instruction. The ARM9EJ-S control logic automatically detects these cases and stalls the core or forward data.

### 12.3.5 Jazelle Technology

The Jazelle technology enables direct and efficient execution of Java byte codes on ARM processors, providing high performance for the next generation of Java-powered wireless and embedded devices.

The new Java feature of ARM9EJ-S can be described as a hardware emulation of a JVM (Java Virtual Machine). Java mode will appear as another state: instead of executing ARM or Thumb instructions, it executes Java byte codes. The Java byte code decoder logic implemented in ARM9EJ-S decodes 95% of executed byte codes and turns them into ARM instructions without any overhead, while less frequently used byte codes are broken down into optimized sequences of ARM instructions. The hardware/software split is invisible to the programmer, invisible to the application and invisible to the operating system. All existing ARM registers are re-used in Jazelle state and all registers then have particular functions in this mode.

Minimum interrupt latency is maintained across both ARM state and Java state. Since byte codes execution can be restarted, an interrupt automatically triggers the core to switch from Java state to ARM state for the execution of the interrupt handler. This means that no special provision has to be made for handling interrupts while executing byte codes, whether in hardware or in software.

### 12.3.6 ARM9EJ-S Operating Modes

In all states, there are seven operation modes:

- User mode is the usual ARM program execution state. It is used for executing most application programs
- Fast Interrupt (FIQ) mode is used for handling fast interrupts. It is suitable for high-speed data transfer or channel process
- Interrupt (IRQ) mode is used for general-purpose interrupt handling

- Supervisor mode is a protected mode for the operating system
- Abort mode is entered after a data or instruction prefetch abort
- System mode is a privileged user mode for the operating system
- Undefined mode is entered when an undefined instruction exception occurs

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, known as privileged modes, are entered in order to service interrupts or exceptions or to access protected resources.

### 12.3.7 ARM9EJ-S Registers

The ARM9EJ-S core has a total of 37 registers:


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

Table 12-1 shows all the registers in all modes.

**Table 12-1.** ARM9TDMI™ 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

The ARM state register set contains 16 directly-accessible registers, r0 to r15, and an additional register, the Current Program Status Register (CPSR). Registers r0 to r13 are general-purpose

registers used to hold either data or address values. Register r14 is used as a Link register that holds a value (return address) of r15 when BL or BLX is executed. Register r15 is used as a program counter (PC), whereas the Current Program Status Register (CPSR) contains condition code flags and the current mode bits.

In privileged modes (FIQ, Supervisor, Abort, IRQ, Undefined), mode-specific banked registers (r8 to r14 in FIQ mode or r13 to r14 in the other modes) become available. The corresponding banked registers r14\_fiq, r14\_svc, r14\_abt, r14\_irq, r14\_und are similarly used to hold the values (return address for each mode) of r15 (PC) when interrupts and exceptions arise, or when BL or BLX instructions are executed within interrupt or exception routines. There is another register called Saved Program Status Register (SPSR) that becomes available in privileged modes instead of CPSR. This register contains condition code flags and the current mode bits saved as a result of the exception that caused entry to the current (privileged) mode.

In all modes and due to a software agreement, register r13 is used as stack pointer.

The use and the function of all the registers described above should obey ARM Procedure Call Standard (APCS) which defines:

- constraints on the use of registers
- stack conventions
- argument passing and result return

The Thumb state register set is a subset of the ARM state set. The programmer has direct access to:

- Eight general-purpose registers r0-r7
- Stack pointer, SP
- Link register, LR (ARM r14)
- PC
- CPSR

There are banked registers SPs, LRs and SPSRs for each privileged mode (for more details see the ARM9EJ-S Technical Reference Manual, ref. DDI0222B, revision r1p2 page 2-12).

### 12.3.7.1 Status Registers

The ARM9EJ-S core contains one CPSR, and five SPSRs for exception handlers to use. The program status registers:

- hold information about the most recently performed ALU operation
- control the enabling and disabling of interrupts
- set the processor operation mode

**Figure 12-2.** Status Register Format

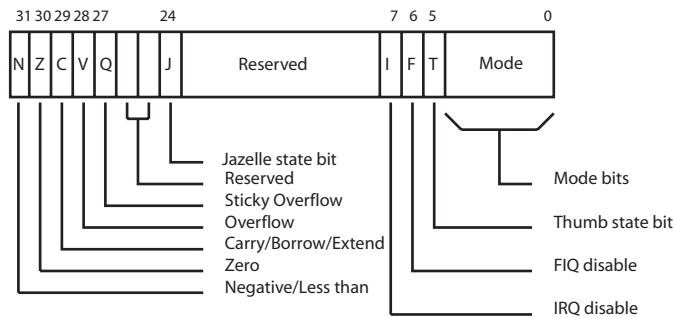


Figure 12-2 shows the status register format, where:

- N: Negative, Z: Zero, C: Carry, and V: Overflow are the four ALU flags
- The Sticky Overflow (Q) flag can be set by certain multiply and fractional arithmetic instructions like QADD, QDADD, QSUB, QDSUB, SMLAxy, and SMLAWy needed to achieve DSP operations.  
The Q flag is sticky in that, when set by an instruction, it remains set until explicitly cleared by an MSR instruction writing to the CPSR. Instructions cannot execute conditionally on the status of the Q flag.
- The J bit in the CPSR indicates when the ARM9EJ-S core is in Jazelle state, where:
  - J = 0: The processor is in ARM or Thumb state, depending on the T bit
  - J = 1: The processor is in Jazelle state.
- Mode: five bits to encode the current processor mode

### 12.3.7.2 Exceptions Exception Types and Priorities

The ARM9EJ-S supports five types of exceptions. Each type drives the ARM9EJ-S in a privileged mode. The types of exceptions are:

- Fast interrupt (FIQ)
- Normal interrupt (IRQ)
- Data and Prefetched aborts (Abort)
- Undefined instruction (Undefined)
- Software interrupt and Reset (Supervisor)

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

More than one exception can happen at a time, therefore the ARM9EJ-S takes the arisen exceptions according to the following priority order:

- Reset (highest priority)
- Data Abort
- FIQ
- IRQ
- Prefetch Abort
- BKPT, Undefined instruction, and Software Interrupt (SWI) (Lowest priority)

The BKPT, or Undefined instruction, and SWI exceptions are mutually exclusive.

There is one exception in the priority scheme though, when FIQs are enabled and a Data Abort occurs at the same time as an FIQ, the ARM9EJ-S core enters the Data Abort handler, and proceeds immediately to FIQ vector. A normal return from the FIQ causes the Data Abort handler to resume execution. Data Aborts must have higher priority than FIQs to ensure that the transfer error does not escape detection.

## *Exception Modes and Handling*

Exceptions arise whenever the normal flow of a program must be halted temporarily, for example, to service an interrupt from a peripheral.

When handling an ARM exception, the ARM9EJ-S core performs the following operations:

1. Preserves the address of the next instruction in the appropriate Link Register that corresponds to the new mode that has been entered. When the exception entry is from:
  - ARM and Jazelle states, the ARM9EJ-S copies the address of the next instruction into LR (current PC(r15) + 4 or PC + 8 depending on the exception).
  - THUMB state, the ARM9EJ-S writes the value of the PC into LR, offset by a value (current PC + 2, PC + 4 or PC + 8 depending on the exception) that causes the program to resume from the correct place on return.
2. Copies the CPSR into the appropriate SPSR.
3. Forces the CPSR mode bits to a value that depends on the exception.
4. Forces the PC to fetch the next instruction from the relevant exception vector.

The register r13 is also banked across exception modes to provide each exception handler with private stack pointer.

The ARM9EJ-S can also set the interrupt disable flags to prevent otherwise unmanageable nesting of exceptions.

When an exception has completed, the exception handler must move both the return value in the banked LR minus an offset to the PC and the SPSR to the CPSR. The offset value varies according to the type of exception. This action restores both PC and the CPSR.

The fast interrupt mode has seven private registers r8 to r14 (banked registers) to reduce or remove the requirement for register saving which minimizes the overhead of context switching.

The Prefetch Abort is one of the aborts that indicates that the current memory access cannot be completed. When a Prefetch Abort occurs, the ARM9EJ-S marks the prefetched instruction as invalid, but does not take the exception until the instruction reaches the Execute stage in the pipeline. If the instruction is not executed, for example because a branch occurs while it is in the pipeline, the abort does not take place.

The breakpoint (BKPT) instruction is a new feature of ARM9EJ-S that is destined to solve the problem of the Prefetch Abort. A breakpoint instruction operates as though the instruction caused a Prefetch Abort.

A breakpoint instruction does not cause the ARM9EJ-S to take the Prefetch Abort exception until the instruction reaches the Execute stage of the pipeline. If the instruction is not executed, for example because a branch occurs while it is in the pipeline, the breakpoint does not take place.

### **12.3.8 ARM Instruction Set Overview**

The ARM instruction set is divided into:

- Branch instructions

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

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

Table 12-2 gives the ARM instruction mnemonic list.

**Table 12-2.** ARM Instruction Mnemonic List

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

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



## 12.3.9 New ARM Instruction Set

**Table 12-3.** New ARM Instruction Mnemonic List

Mnemonic	Operation	Mnemonic	Operation
BXJ	Branch and exchange to Java	MRRC	Move double from coprocessor
BLX <sup>(1)</sup>	Branch, Link and exchange	MCR2	Alternative move of ARM reg to coprocessor
SMLAxy	Signed Multiply Accumulate 16 * 16 bit	MCRR	Move double to coprocessor
SMLAL	Signed Multiply Accumulate Long	CDP2	Alternative Coprocessor Data Processing
SMLAWy	Signed Multiply Accumulate 32 * 16 bit	BKPT	Breakpoint
SMULxy	Signed Multiply 16 * 16 bit	PLD	Soft Preload, Memory prepare to load from address
SMULWy	Signed Multiply 32 * 16 bit	STRD	Store Double
QADD	Saturated Add	STC2	Alternative Store from Coprocessor
QDADD	Saturated Add with Double	LDRD	Load Double
QSUB	Saturated subtract	LDC2	Alternative Load to Coprocessor
QDSUB	Saturated Subtract with double	CLZ	Count Leading Zeroes

Notes: 1. A Thumb BLX contains two consecutive Thumb instructions, and takes four cycles.

## 12.3.10 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

Table 5 shows the Thumb instruction set. [Table 12-4](#) gives the Thumb instruction mnemonic list.

**Table 12-4.** 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

**Table 12-4.** Thumb Instruction Mnemonic List (Continued)

Mnemonic	Operation	Mnemonic	Operation
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	BLX	Branch, Link, and Exchange
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
LDRSH	Load Signed Halfword	LDRSB	Load Signed Byte
LDMIA	Load Multiple	STMIA	Store Multiple
PUSH	Push Register to stack	POP	Pop Register from stack
BCC	Conditional Branch	BKPT	Breakpoint

## 12.4 CP15 Coprocessor

Coprocessor 15, or System Control Coprocessor CP15, is used to configure and control all the items in the list below:

- ARM9EJ-S
- Caches (ICache, DCache and write buffer)
- TCM
- MMU
- Other system options

To control these features, CP15 provides 16 additional registers. See [Table 12-5](#).

**Table 12-5.** CP15 Registers

Register	Name	Read/Write
0	ID Code <sup>(1)</sup>	Read/Unpredictable
0	Cache type <sup>(1)</sup>	Read/Unpredictable
0	TCM status <sup>(1)</sup>	Read/Unpredictable
1	Control	Read/write
2	Translation Table Base	Read/write
3	Domain Access Control	Read/write
4	Reserved	None
5	Data fault Status <sup>(1)</sup>	Read/write
5	Instruction fault status <sup>(1)</sup>	Read/write
6	Fault Address	Read/write
7	Cache Operations	Read/Write

**Table 12-5. CP15 Registers**

Register	Name	Read/Write
8	TLB operations	Unpredictable/Write
9	cache lockdown <sup>(2)</sup>	Read/write
9	TCM region	Read/write
10	TLB lockdown	Read/write
11	Reserved	None
12	Reserved	None
13	FCSE PID <sup>(1)</sup>	Read/write
13	Context ID <sup>(1)</sup>	Read/Write
14	Reserved	None
15	Test configuration	Read/Write

- Notes:
1. Register locations 0,5, and 13 each provide access to more than one register. The register accessed depends on the value of the opcode\_2 field.
  2. Register location 9 provides access to more than one register. The register accessed depends on the value of the CRm field.

### 12.4.1 CP15 Registers Access

CP15 registers can only be accessed in privileged mode by:

- MCR (Move to Coprocessor from ARM Register) instruction is used to write an ARM register to CP15.
- MRC (Move to ARM Register from Coprocessor) instruction is used to read the value of CP15 to an ARM register.

Other instructions like CDP, LDC, STC can cause an undefined instruction exception.

The assembler code for these instructions is:

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

The MCR, MRC instructions bit pattern is shown below:

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

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

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

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

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

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

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

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

Determines the destination coprocessor register.

- **L: Instruction Bit**

0 = MCR instruction

1 = MRC instruction

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

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

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

For more details, see Chapter 2 in ARM926EJ-S TRM, ref. DDI0198B.

## 12.5 Memory Management Unit (MMU)

The ARM926EJ-S processor implements an enhanced ARM architecture v5 MMU to provide virtual memory features required by operating systems like Symbian OS®, WindowsCE, and Linux. These virtual memory features are memory access permission controls and virtual to physical address translations.

The Virtual Address generated by the CPU core is converted to a Modified Virtual Address (MVA) by the FCSE (Fast Context Switch Extension) using the value in CP15 register13. The MMU translates modified virtual addresses to physical addresses by using a single, two-level page table set stored in physical memory. Each entry in the set contains the access permissions and the physical address that correspond to the virtual address.

The first level translation tables contain 4096 entries indexed by bits [31:20] of the MVA. These entries contain a pointer to either a 1 MB section of physical memory along with attribute information (access permissions, domain, etc.) or an entry in the second level translation tables; coarse table and fine table.

The second level translation tables contain two subtables, coarse table and fine table. An entry in the coarse table contains a pointer to both large pages and small pages along with access permissions. An entry in the fine table contains a pointer to large, small and tiny pages.

Table 7 shows the different attributes of each page in the physical memory.

**Table 12-6.** Mapping Details

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

The MMU consists of:

- Access control logic
- Translation Look-aside Buffer (TLB)
- Translation table walk hardware

### 12.5.1 Access Control Logic

The access control logic controls access information for every entry in the translation table. The access control logic checks two pieces of access information: domain and access permissions. The domain is the primary access control mechanism for a memory region; there are 16 of them. It defines the conditions necessary for an access to proceed. The domain determines whether the access permissions are used to qualify the access or whether they should be ignored.

The second access control mechanism is access permissions that are defined for sections and for large, small and tiny pages. Sections and tiny pages have a single set of access permissions whereas large and small pages can be associated with 4 sets of access permissions, one for each subpage (quarter of a page).

### 12.5.2 Translation Look-aside Buffer (TLB)

The Translation Look-aside Buffer (TLB) caches translated entries and thus avoids going through the translation process every time. When the TLB contains an entry for the MVA (Modified Virtual Address), the access control logic determines if the access is permitted and outputs the appropriate physical address corresponding to the MVA. If access is not permitted, the MMU signals the CPU core to abort.

If the TLB does not contain an entry for the MVA, the translation table walk hardware is invoked to retrieve the translation information from the translation table in physical memory.

### 12.5.3 Translation Table Walk Hardware

The translation table walk hardware is a logic that traverses the translation tables located in physical memory, gets the physical address and access permissions and updates the TLB.

The number of stages in the hardware table walking is one or two depending whether the address is marked as a section-mapped access or a page-mapped access.

There are three sizes of page-mapped accesses and one size of section-mapped access. Page-mapped accesses are for large pages, small pages and tiny pages. The translation process always begins with a level one fetch. A section-mapped access requires only a level one fetch, but a page-mapped access requires an additional level two fetch. For further details on the MMU, please refer to chapter 3 in ARM926EJ-S Technical Reference Manual, ref. DDI0198B.

### 12.5.4 MMU Faults

The MMU generates an abort on the following types of faults:

- Alignment faults (for data accesses only)
- Translation faults
- Domain faults
- Permission faults

The access control mechanism of the MMU detects the conditions that produce these faults. If the fault is a result of memory access, the MMU aborts the access and signals the fault to the CPU core. The MMU retains status and address information about faults generated by the data accesses in the data fault status register and fault address register. It also retains the status of faults generated by instruction fetches in the instruction fault status register.

The fault status register (register 5 in CP15) indicates the cause of a data or prefetch abort, and the domain number of the aborted access when it happens. The fault address register (register 6 in CP15) holds the MVA associated with the access that caused the Data Abort. For further details on MMU faults, please refer to chapter 3 in ARM926EJ-S Technical Reference Manual, ref. DDI0198B.

## 12.6 Caches and Write Buffer

The ARM926EJ-S contains a 4 KB Instruction Cache (ICache), a 4 KB Data Cache (DCache), and a write buffer. Although the ICache and DCache share common features, each still has some specific mechanisms.

The caches (ICache and DCache) are four-way set associative, addressed, indexed and tagged using the Modified Virtual Address (MVA), with a cache line length of eight words with two dirty bits for the DCache. The ICache and DCache provide mechanisms for cache lockdown, cache pollution control, and line replacement.

A new feature is now supported by ARM926EJ-S caches called allocate on read-miss commonly known as wrapping. This feature enables the caches to perform critical word first cache refilling. This means that when a request for a word causes a read-miss, the cache performs an AHB access. Instead of loading the whole line (eight words), the cache loads the critical word first, so the processor can reach it quickly, and then the remaining words, no matter where the word is located in the line.

The caches and the write buffer are controlled by the CP15 register 1 (Control), CP15 register 7 (cache operations) and CP15 register 9 (cache lockdown).

## 12.6.1 Instruction Cache (ICache)

The ICache caches fetched instructions to be executed by the processor. The ICache can be enabled by writing 1 to I bit of the CP15 Register 1 and disabled by writing 0 to this same bit.

When the MMU is enabled, all instruction fetches are subject to translation and permission checks. If the MMU is disabled, all instructions fetches are cachable, no protection checks are made and the physical address is flat-mapped to the modified virtual address. With the MVA use disabled, context switching incurs ICache cleaning and/or invalidating.

When the ICache is disabled, all instruction fetches appear on external memory (AHB) (see Tables 4-1 and 4-2 in page 4-4 in ARM926EJ-S TRM, ref. DDI0198B).

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

## 12.6.2 Data Cache (DCache) and Write Buffer

ARM926EJ-S includes a DCache and a write buffer to reduce the effect of main memory bandwidth and latency on data access performance. The operations of DCache and write buffer are closely connected.

### 12.6.2.1 DCache

The DCache needs the MMU to be enabled. All data accesses are subject to MMU permission and translation checks. Data accesses that are aborted by the MMU do not cause linefills or data accesses to appear on the AMBA ASB interface. If the MMU is disabled, all data accesses are noncachable, nonbufferable, with no protection checks, and appear on the AHB bus. All addresses are flat-mapped,  $VA = MVA = PA$ , which incurs DCache cleaning and/or invalidating every time a context switch occurs.

The DCache stores the Physical Address Tag (PA Tag) from which every line was loaded and uses it when writing modified lines back to external memory. This means that the MMU is not involved in write-back operations.

Each line (8 words) in the DCache has two dirty bits, one for the first four words and the other one for the second four words. These bits, if set, mark the associated half-lines as dirty. If the cache line is replaced due to a linefill or a cache clean operation, the dirty bits are used to decide whether all, half or none is written back to memory.

DCache can be enabled or disabled by writing either 1 or 0 to bit C in register 1 of CP15 (see Tables 4-3 and 4-4 on page 4-5 in ARM926EJ-S TRM, ref. DDI0222B).

The DCache supports write-through and write-back cache operations, selected by memory region using the C and B bits in the MMU translation tables.

The DCache contains an eight data word entry, single address entry write-back buffer used to hold write-back data for cache line eviction or cleaning of dirty cache lines.

The Write Buffer can hold up to 16 words of data and four separate addresses. DCache and Write Buffer operations are closely connected as their configuration is set in each section by the page descriptor in the MMU translation table.

### 12.6.2.2 Write Buffer

The ARM926EJ-S contains a write buffer that has a 16-word data buffer and a four- address buffer. The write buffer is used for all writes to a bufferable region, write-through region and write-back region. It also allows to avoid stalling the processor when writes to external memory are performed. When a store occurs, data is written to the write buffer at core speed (high speed). The write buffer then completes the store to external memory at bus speed (typically slower than the core speed). During this time, the ARM9EJ-S processor can preform other tasks.

DCache and Write Buffer support write-back and write-through memory regions, controlled by C and B bits in each section and page descriptor within the MMU translation tables.

#### *Write-through Operation*

When a cache write hit occurs, the DCache line is updated. The updated data is then written to the write buffer which transfers it to external memory.

When a cache write miss occurs, a line, chosen by round robin or another algorithm, is stored in the write buffer which transfers it to external memory.

#### *Write-back Operation*

When a cache write hit occurs, the cache line or half line is marked as dirty, meaning that its contents are not up-to-date with those in the external memory.

When a cache write miss occurs, a line, chosen by round robin or another algorithm, is stored in the write buffer which transfers it to external memory.

## 12.7 Tightly-Coupled Memory Interface

### 12.7.1 TCM Description

The ARM926EJ-S processor features a Tightly-Coupled Memory (TCM) interface, which enables separate instruction and data TCMs (ITCM and DTCM) to be directly reached by the processor. TCMs are used to store real-time and performance critical code, they also provide a DMA support mechanism. Unlike AHB accesses to external memories, accesses to TCMs are fast and deterministic and do not incur bus penalties.

The user has the possibility to independently configure each TCM size with values within the following ranges, [0KB, 32 KB] for ITCM size and [0KB, 32 KB] for DTCM size.

TCMs can be configured by two means: HMATRIX TCM register and TCM region register (register 9) in CP15 and both steps should be performed. HMATRIX TCM register sets TCM size whereas TCM region register (register 9) in CP15 maps TCMs and enables them.

The data side of the ARM9EJ-S core is able to access the ITCM. This is necessary to enable code to be loaded into the ITCM, for SWI and emulated instruction handlers, and for accesses to PC-relative literal pools.



## 12.7.2 Enabling and Disabling TCMs

Prior to any enabling step, the user should configure the TCM sizes in HMATRIX TCM register. Then enabling TCMs is performed by using TCM region register (register 9) in CP15. The user should use the same sizes as those put in HMATRIX TCM register. For further details and programming tips, please refer to chapter 2.3 in ARM926EJ-S TRM, ref. DDI0222B.

## 12.7.3 TCM Mapping

The TCMs can be located anywhere in the memory map, with a single region available for ITCM and a separate region available for DTCM. The TCMs are physically addressed and can be placed anywhere in physical address space. However, the base address of a TCM must be aligned to its size, and the DTCM and ITCM regions must not overlap. TCM mapping is performed by using TCM region register (register 9) in CP15. The user should input the right mapping address for TCMs.

## 12.8 Bus Interface Unit

The ARM926EJ-S features a Bus Interface Unit (BIU) that arbitrates and schedules AHB requests. The BIU 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. This is achieved by using a more complex interconnection matrix and gives the benefit of increased overall bus bandwidth, and a more flexible system architecture.

The multi-master bus architecture has a number of benefits:

- It allows the development of multi-master systems with an increased bus bandwidth and a flexible architecture.
- Each AHB layer becomes simple because it only has one master, so no arbitration or master-to-slave muxing is required. AHB layers, implementing AHB-Lite protocol, do not have to support request and grant, nor do they have to support retry and split transactions.
- The arbitration becomes effective when more than one master wants to access the same slave simultaneously.

### 12.8.1 Supported Transfers

The ARM926EJ-S processor performs all AHB accesses as single word, bursts of four words, or bursts of eight words. Any ARM9EJ-S core request that is not 1, 4, 8 words in size is split into packets of these sizes. Note that the Atmel bus is AHB-Lite protocol compliant, hence it does not support split and retry requests.

Table 8 gives an overview of the supported transfers and different kinds of transactions they are used for.

**Table 12-7.** Supported Transfers

HBurst[2:0]	Description	
SINGLE	Single transfer	Single transfer of word, half word, or byte: <ul style="list-style-type: none"> <li>• data write (NCNB, NCB, WT, or WB that has missed in DCache)</li> <li>• data read (NCNB or NCB)</li> <li>• NC instruction fetch (prefetched and non-prefetched)</li> <li>• page table walk read</li> </ul>
INCR4	Four-word incrementing burst	Half-line cache write-back, Instruction prefetch, if enabled. Four-word burst NCNB, NCB, WT, or WB write.
INCR8	Eight-word incrementing burst	Full-line cache write-back, eight-word burst NCNB, NCB, WT, or WB write.
WRAP8	Eight-word wrapping burst	Cache linefill

### 12.8.2 Thumb Instruction Fetches

All instructions fetches, regardless of the state of ARM9EJ-S core, are made as 32-bit accesses on the AHB. If the ARM9EJ-S is in Thumb state, then two instructions can be fetched at a time.

### 12.8.3 Address Alignment

The ARM926EJ-S BIU performs address alignment checking and aligns AHB addresses to the necessary boundary. 16-bit accesses are aligned to halfword boundaries, and 32-bit accesses are aligned to word boundaries.

## 13. AT91SAM9R64/RL64 Debug and Test

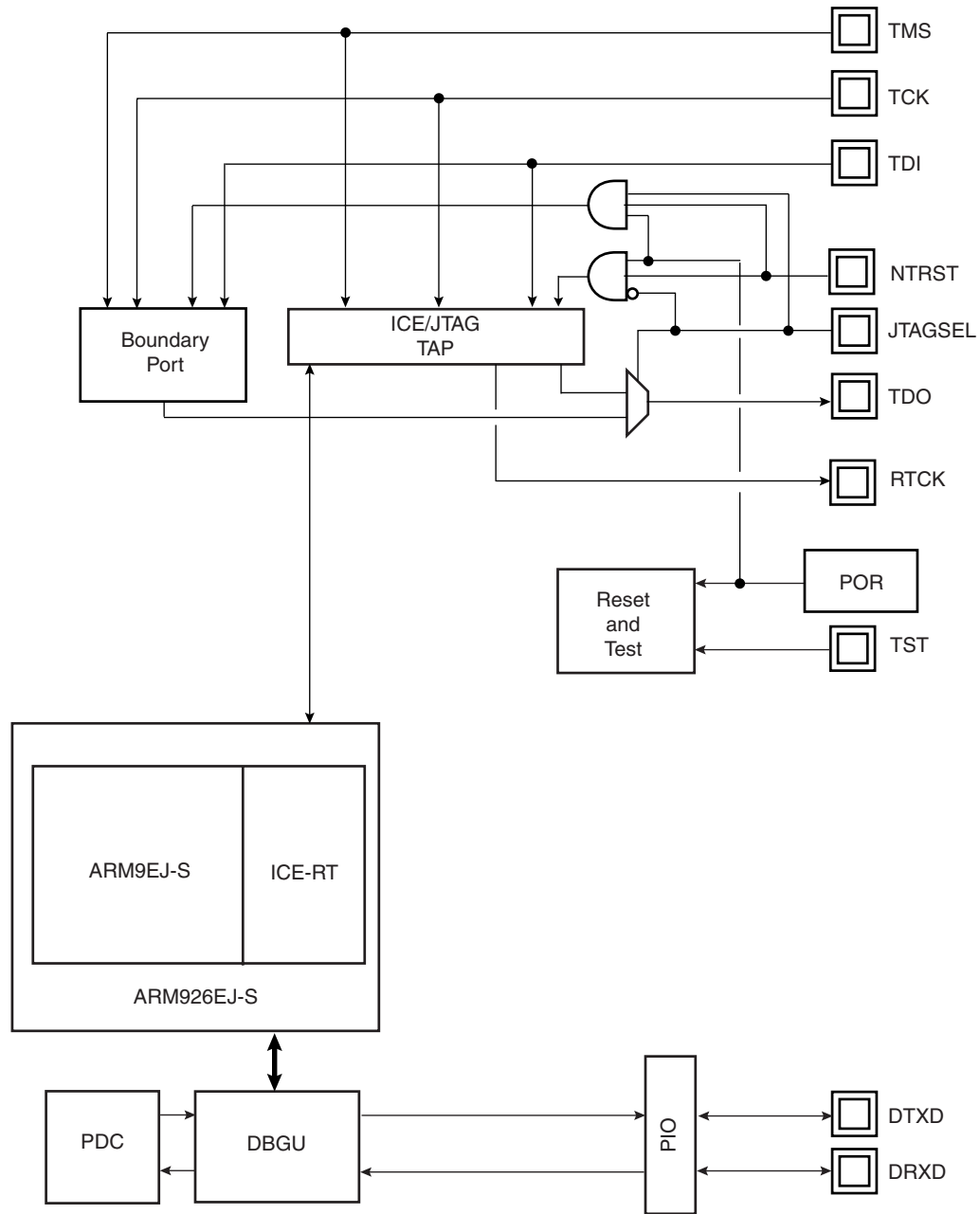
### 13.1 Description

The AT91SAM9R64/RL64 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



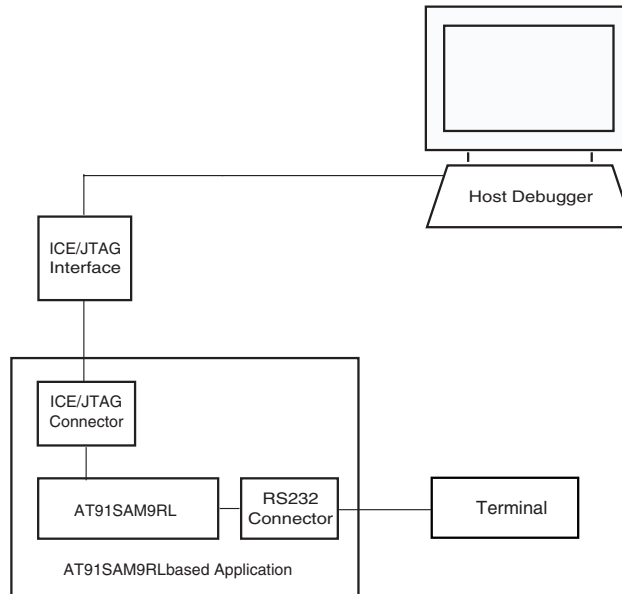
TAP: Test Access Port

## 13.3 Application Examples

### 13.3.1 Debug Environment

Figure 13-2 on page 69 shows a complete debug environment example. The ICE/JTAG interface is used for standard debugging functions, such as downloading code and single-stepping through the program. The Trace Port interface is used for tracing information. A software debugger running on a personal computer provides the user interface for configuring a Trace Port interface utilizing the ICE/JTAG interface.

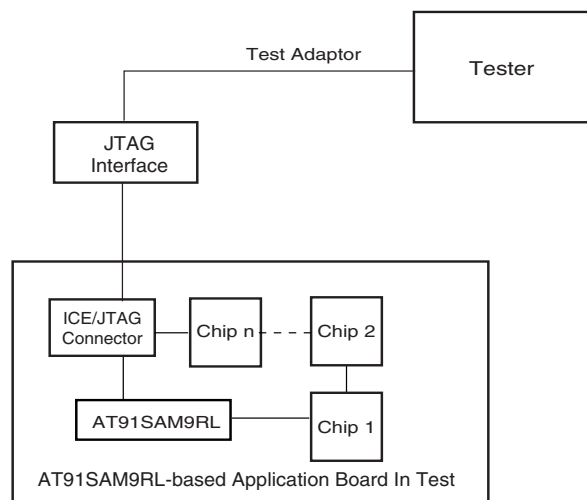
**Figure 13-2.** Application Debug and Trace Environment Example



### 13.3.2 Test Environment

Figure 13-3 on page 69 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



## 13.4 Debug and Test Pin Description

**Table 13-1.** Debug and Test Pin List

Pin Name	Function	Type	Active Level
<b>Reset/Test</b>			
NTRST	Test Reset Signal	Input	Low
NRST	Microcontroller Reset	Input/Output	Low
TST	Test Mode Select	Input	High
<b>ICE and JTAG</b>			
TCK	Test Clock	Input	
TDI	Test Data In	Input	
TDO	Test Data Out	Output	
TMS	Test Mode Select	Input	
RTCK	Returned Test Clock	Output	
JTAGSEL	JTAG Selection	Input	
<b>Debug Unit</b>			
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 ARM9EJ-S Embedded In-Circuit Emulator-RT is supported via the ICE/JTAG port. It is connected to a host computer via an ICE interface. Debug support is implemented using an ARM9EJ-S core embedded within the ARM926EJ-S. The internal state of the ARM926EJ-S is examined through an ICE/JTAG port which allows instructions to be serially inserted into the pipeline of the core without using the external data bus. Therefore, when in debug state, a store-multiple (STM) can be inserted into the instruction pipeline. This exports the contents of the ARM9EJ-S registers. This data can be serially shifted out without affecting the rest of the system.

There are two scan chains inside the ARM9EJ-S processor which support testing, debugging, and programming of the Embedded ICE-RT™. 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-RT, see the ARM document:

ARM9EJ-S Technical Reference Manual (DDI 0222A).

## 13.5.3 JTAG Signal Description

TMS is the Test Mode Select input which controls the transitions of the test interface state machine.

TDI is the Test Data Input line which supplies the data to the JTAG registers (Boundary Scan Register, Instruction Register, or other data registers).

TDO is the Test Data Output line which is used to serially output the data from the JTAG registers to the equipment controlling the test. It carries the sampled values from the boundary scan chain (or other JTAG registers) and propagates them to the next chip in the serial test circuit.

NTRST (optional in IEEE Standard 1149.1) is a Test-ReSeT input which is mandatory in ARM cores and used to reset the debug logic. On Atmel ARM926EJ-S-based cores, NTRST is a Power On Reset output. It is asserted on power on. If necessary, the user can also reset the debug logic with the NTRST pin assertion during 2.5 MCK periods.

TCK is the Test Clock input which enables the test interface. TCK is pulsed by the equipment controlling the test and not by the tested device. It can be pulsed at any frequency. Note the maximum JTAG clock rate on ARM926EJ-S cores is 1/6th the clock of the CPU. This gives 5.45 kHz maximum initial JTAG clock rate for an ARM9E running from the 32.768 kHz slow clock.

RTCK is the Return Test Clock. Not an IEEE Standard 1149.1 signal added for a better clock handling by emulators. From some ICE Interface probes, this return signal can be used to synchronize the TCK clock and take not care about the given ratio between the ICE Interface clock and system clock equal to 1/6th. This signal is only available in JTAG ICE Mode and not in boundary scan mode.

## 13.5.4 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 data 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, gives information about the product version and its internal configuration.

The AT91SAM9R64/RL64 Debug Unit Chip ID value is 0x0196 07A0 on 32-bit width.

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

## 13.5.5 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.

## 13.5.6 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

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

Set to 0x01F.

Bit[0] Required by IEEE Std. 1149.1.

Set to 0x1.

JTAG ID Code value is 0x05B2\_003F.

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

Product part Number is 0x5B20.

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

Set to 0x0.



## 14. AT91SAM9R64/RL64 Boot Program

### 14.1 Description

The Boot Program integrates different programs that manage download and/or upload into the different memories of the product.

First, it initializes the Debug Unit serial port (DBGU) and the USB High Speed Device Port.

Then the SD Card Boot program is executed. It looks for a boot.bin file in the root directory of a FAT12/16/32 formatted SD Card. If such a file is found, code is downloaded into the internal SRAM. This is followed by a remap and a jump to the first address of the SRAM.

If the SD Card is not formatted or if boot.bin file is not found, NAND Flash Boot program is then executed. The NAND Flash Boot program searches for a valid application in the NAND Flash memory. If a valid application is found, this application is loaded into internal SRAM and executed by branching at address 0x0000\_0000 after remap. See [“DataFlash Boot” on page 75](#) for more information on Valid Image Detection.

If no valid ARM vector sequence is found, the DataFlash Boot program is then executed. It looks for a sequence of seven valid ARM exception vectors in a DataFlash connected to the SPI. All these vectors must be B-branch or LDR load register instructions except for the sixth vector. This vector is used to store the size of the image to download.

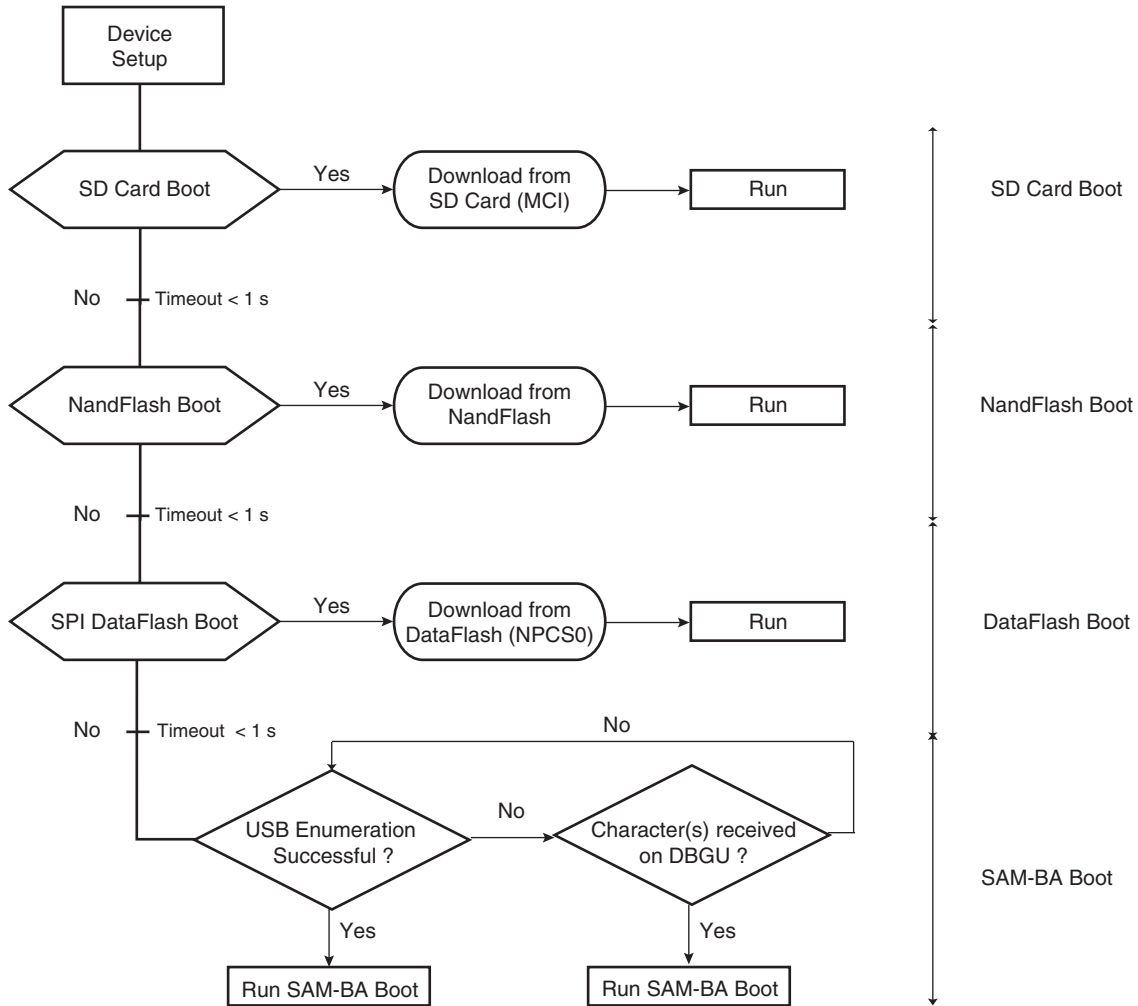
If a valid sequence is found, code is downloaded into the internal SRAM. This is followed by a remap and a jump to the first address of the SRAM.

If no valid ARM vector sequence is found, SAM-BA Boot is then executed. It waits for transactions either on the USB device, or on the DBGU serial port.

### 14.2 Flow Diagram

The Boot Program implements the algorithm in [Figure 14-1](#).

**Figure 14-1.** Boot Program Algorithm Flow Diagram



### 14.3 Device Initialization

Initialization follows the steps described below:

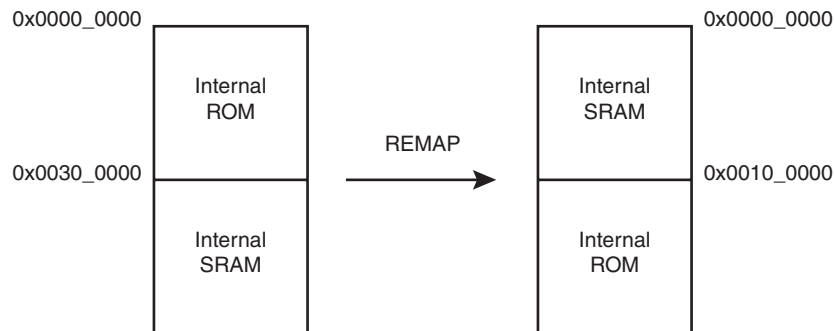
1. Stack setup for ARM supervisor mode
2. Main Oscillator Frequency Detection
3. C variable initialization
4. UTMI PLL is enabled to generate a 480MHz clock necessary to use the USB High Speed Device.
5. PLL setup: PLL is initialized to generate a 96 MHz clock.

Note: A 12 MHz Crystal is mandatory in order to generate these clocks correctly.

6. MCK is configured to generate a 48MHz clock (PLL/2).
7. Initialization of the DBGU serial port (115200 bauds, 8, N, 1)
8. Enable the user reset

9. Jump to SD Card Boot sequence. If SD Card Boot succeeds, perform a remap and jump to 0x0.
10. Jump to NAND Flash Boot sequence. If NAND Flash Boot succeeds, perform a remap and jump to 0x0.
11. Jump to DataFlash Boot sequence through NPCS0. If DataFlash Boot succeeds, perform a remap and jump to 0x0.
12. Activation of the Instruction Cache
13. Jump to SAM-BA Boot sequence
14. Disable the WatchDog
15. Initialization of the USB Device Port

**Figure 14-2.** Remap Action after Download Completion



## 14.4 DataFlash Boot

The DataFlash Boot program searches for a valid application in the SPI DataFlash memory. If a valid application is found, this application is loaded into internal SRAM and executed by branching at address 0x0000\_0000 after remap. This application may be the application code or a second-level bootloader.

All the calls to functions are PC relative and do not use absolute addresses.

After reset, the code in internal ROM is mapped at both addresses 0x0000\_0000 and 0x0040\_0000:

400000	ea000006	B	0x20	00ea000006B0x20
400004	eaffffffe	B	0x04	04eaffffffeB0x04
400008	ea00002f	B	_main	08ea00002fB_main
40000c	eaffffffe	B	0x0c	0ceaffffffeB0x0c
400010	eaffffffe	B	0x10	10eaffffffeB0x10
400014	eaffffffe	B	0x14	14eaffffffeB0x14
400018	eaffffffe	B	0x18	18eaffffffeB0x18

### 14.4.1 Valid Image Detection

The DataFlash Boot software looks for a valid application by analyzing the first 28 bytes corresponding to the ARM exception vectors. These bytes must implement ARM instructions for either branch or load PC with PC relative addressing.

The sixth vector, at offset 0x14, contains the size of the image to download. The user must replace this vector with his own vector (see [“Structure of ARM Vector 6”](#) on page 76).

**Figure 14-3.** LDR Opcode

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

**Figure 14-4.** B Opcode

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

Unconditional instruction: 0xE for bits 31 to 28

Load PC with PC relative addressing instruction:

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

## 14.4.2 Structure of ARM Vector 6

The ARM exception vector 6 is used to store information needed by the DataFlash boot program. This information is described below.

**Figure 14-5.** Structure of the ARM Vector 6

31	0
Size of the code to download in bytes	

### 14.4.2.1 Example

An example of valid vectors follows:

```

00 ea000006 B 0x20
04 eaffffffe B 0x04
08 ea00002f B _main
0c eaffffffe B 0x0c
10 eaffffffe B 0x10
14 00001234 <- Code size = 4660 bytes
18 eaffffffe B 0x18
    
```

The size of the image to load into SRAM is contained in the location of the sixth ARM vector. Thus the user must replace this vector by the correct vector for his application.

## 14.4.3 DataFlash Boot Sequence

The DataFlash boot program performs device initialization followed by the download procedure.

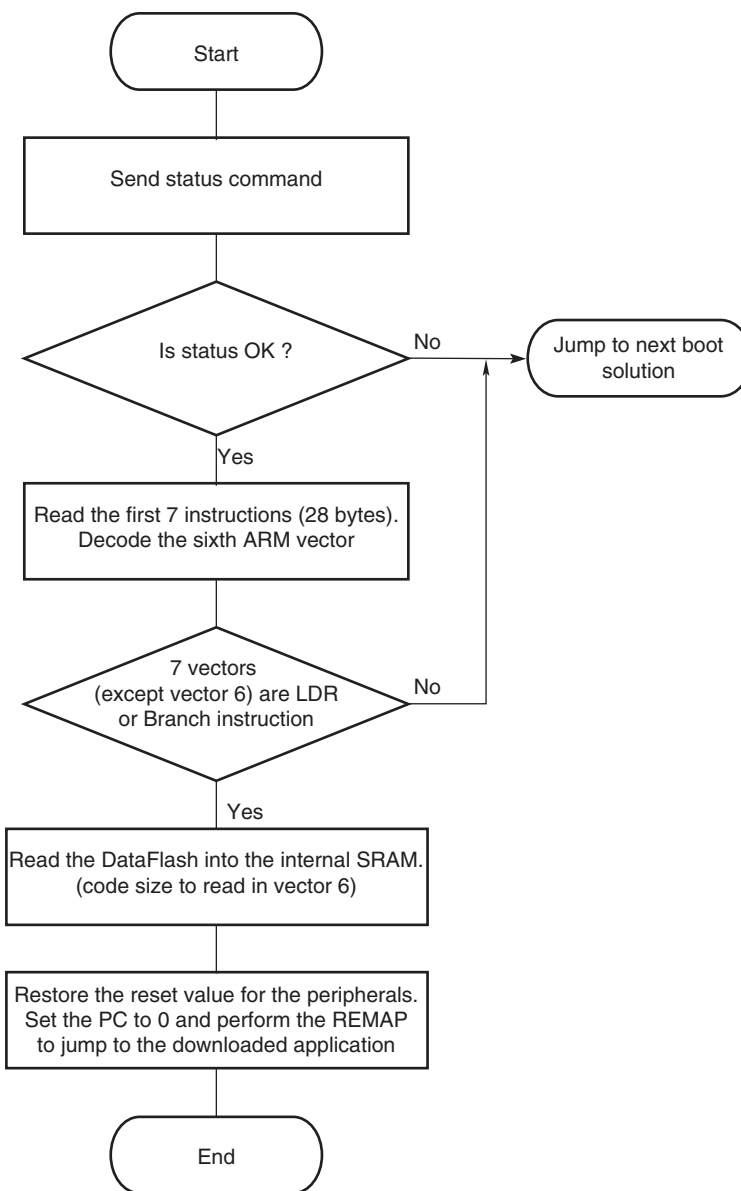
The DataFlash boot program supports all Atmel DataFlash devices. [Table 14-1](#) summarizes the parameters to include in the ARM vector 6 for all devices.

**Table 14-1.** DataFlash Device

Device	Density	Page Size (bytes)	Number of Pages
AT45DB011	1 Mbit	264	512
AT45DB021	2 Mbits	264	1024
AT45DB041	4 Mbits	264	2048
AT45DB081	8 Mbits	264	4096
AT45DB161	16 Mbits	528	4096
AT45DB321	32 Mbits	528	8192
AT45DB642	64 Mbits	1056	8192

The DataFlash has a Status Register that determines all the parameters required to access the device. The DataFlash boot is configured to be compatible with the future design of the DataFlash.

**Figure 14-6.** Serial DataFlash Download



## 14.5 SD Card Boot

The SD Card Boot program searches for a valid application in the SD Card memory.

It looks for a boot.bin file in the root directory of a FAT12/16/32 formatted SD Card. If a valid file is found, this application is loaded into internal SRAM and executed by branching at address 0x0000\_0000 after remap. This application may be the application code or a second-level bootloader.

## 14.6 NAND Flash Boot

The NAND Flash Boot program searches for a valid application in the NAND Flash memory. If a valid application is found, this application is loaded into internal SRAM and executed by branch-

ing at address 0x0000\_0000 after remap. See “DataFlash Boot” on page 75 for more information on Valid Image Detection.

## 14.6.1 Supported NAND Flash Devices

Any 8 or 16-bit NAND Flash devices.

## 14.7 SAM-BA Boot

If no valid DataFlash device has been found during the DataFlash boot sequence, the SAM-BA boot program is performed.

The SAM-BA boot principle is to:

- Check if USB High Speed Device enumeration has occurred.
- Check if characters have been received on the DBGU.
- Once the communication interface is identified, the application runs in an infinite loop waiting for different commands as in Table 14-2.

**Table 14-2.** Commands Available through the SAM-BA Boot

Command	Action	Argument(s)	Example
O	write a byte	Address, Value#	O200001,CA#
o	read a byte	Address,#	o200001,#
H	write a half word	Address, Value#	H200002,CAFE#
h	read a half word	Address,#	h200002,#
W	write a word	Address, Value#	W200000,CAFEDCA#
w	read a word	Address,#	w200000,#
S	send a file	Address,#	S200000,#
R	receive a file	Address, NbOfBytes#	R200000,1234#
G	go	Address#	G200200#
V	display version	No argument	V#

- Write commands: Write a byte (**O**), a halfword (**H**) or a word (**W**) to the target.
  - *Address*: Address in hexadecimal.
  - *Value*: Byte, halfword or word to write in hexadecimal.
  - *Output*: '>'.
- Read commands: Read a byte (**o**), a halfword (**h**) or a word (**w**) from the target.
  - *Address*: Address in hexadecimal
  - *Output*: The byte, halfword or word read in hexadecimal following by '>'
- Send a file (**S**): Send a file to a specified address
  - *Address*: Address in hexadecimal
  - *Output*: '>'.

Note: There is a time-out on this command which is reached when the prompt '>' appears before the end of the command execution.

- Receive a file (**R**): Receive data into a file from a specified address
  - *Address*: Address in hexadecimal

- *NbOfBytes*: Number of bytes in hexadecimal to receive
- *Output*: '>'
- Go (**G**): Jump to a specified address and execute the code
  - *Address*: Address to jump in hexadecimal
  - *Output*: '>'
- Get Version (**V**): Return the SAM-BA boot version
  - *Output*: '>'

## 14.7.1 DBGU Serial Port

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

The Send and Receive File commands use the Xmodem protocol to communicate. Any terminal performing this protocol can be used to send the application file to the target. The size of the binary file to send depends on the SRAM size embedded in the product. In all cases, the size of the binary file must be lower than the SRAM size because the Xmodem protocol requires some SRAM memory to work.

## 14.7.2 Xmodem Protocol

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

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

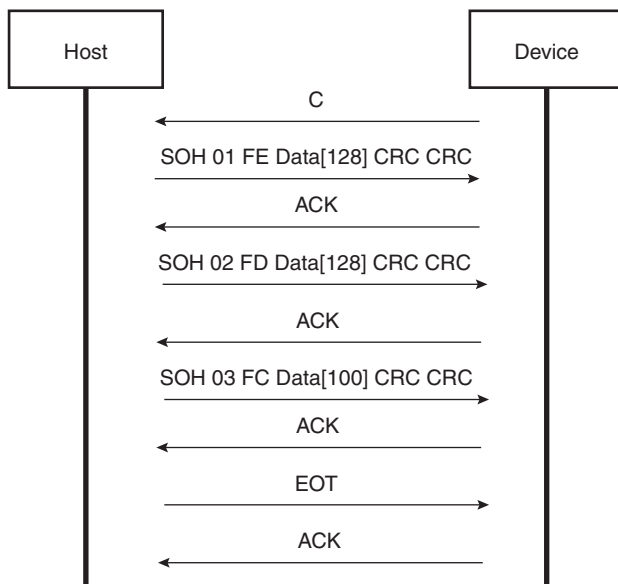
<SOH><blk #><255-blk #><--128 data bytes--><checksum> in which:

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

Figure 14-7 shows a transmission using this protocol.



**Figure 14-7.** Xmodem Transfer Example



### 14.7.3 USB High Speed Device Port

A 480 MHz USB clock is necessary to use the USB High Speed Device port. It has been programmed earlier in the device initialization procedure with UTMI PLL configuration.

The device uses the USB communication device class (CDC) drivers to take advantage of the installed PC RS-232 software to talk over the USB. The CDC class is implemented in all releases of Windows®, from Windows 98SE to Windows XP®. The CDC document, available at [www.usb.org](http://www.usb.org), describes a way to implement devices such as ISDN modems and virtual COM ports.

The Vendor ID is Atmel’s vendor ID 0x03EB. The product ID is 0x6124. These references are used by the host operating system to mount the correct driver. On Windows systems, the INF files contain the correspondence between vendor ID and product ID.

Atmel provides an INF example to see the device as a new serial port and also provides another custom driver used by the SAM-BA application: atm6124.sys. Refer to the document “USB Basic Application”, literature number 6123, for more details.

#### 14.7.3.1 Enumeration Process

The USB protocol is a master/slave protocol. This is the host that starts the enumeration sending requests to the device through the control endpoint. The device handles standard requests as defined in the USB Specification.

**Table 14-3.** Handled Standard Requests

Request	Definition
GET_DESCRIPTOR	Returns the current device configuration value.
SET_ADDRESS	Sets the device address for all future device access.
SET_CONFIGURATION	Sets the device configuration.
GET_CONFIGURATION	Returns the current device configuration value.

**Table 14-3.** Handled Standard Requests (Continued)

Request	Definition
GET_STATUS	Returns status for the specified recipient.
SET_FEATURE	Used to set or enable a specific feature.
CLEAR_FEATURE	Used to clear or disable a specific feature.

The device also handles some class requests defined in the CDC class.

**Table 14-4.** Handled Class Requests

Request	Definition
SET_LINE_CODING	Configures DTE rate, stop bits, parity and number of character bits.
GET_LINE_CODING	Requests current DTE rate, stop bits, parity and number of character bits.
SET_CONTROL_LINE_STATE	RS-232 signal used to tell the DCE device the DTE device is now present.

Unhandled requests are STALLED.

### 14.7.3.2 Communication Endpoints

There are two communication endpoints and endpoint 0 is used for the enumeration process. Endpoint 1 is a 512-byte Bulk OUT endpoint and endpoint 2 is a 512-byte Bulk IN endpoint. SAM-BA Boot commands are sent by the host through the endpoint 1. If required, the message is split by the host into several data payloads by the host driver.

If the command requires a response, the host can send IN transactions to pick up the response.

## 14.8 Hardware and Software Constraints

- A 12 MHz Crystal is mandatory in order to generate correctly 480 MHz clock necessary for the USB High Speed Device and to generate the 48 MHz System clock.
- No Bypass Mode.
- The SD Card, NAND Flash and DataFlash downloaded code size must be inferior to 56 K bytes.
- The code is always downloaded from the DataFlash or NAND Flash device address 0x0000\_0000 to the address 0x0000\_0000 of the internal SRAM (after remap).
- The downloaded code must be position-independent or linked at address 0x0000\_0000.
- The DataFlash must be connected to NPCS0 of the SPI.

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

To assure correct functionality, it is recommended to plug in critical devices to other pins.

Table 14-5 contains a list of pins that are driven during the boot program execution. These pins are driven during the boot sequence for a period of less than 1 second if no correct boot program is found.

For the DataFlash driven by the SPCK signal at 8 MHz, the time to download 60 K bytes is reduced to 200 ms.

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

**Table 14-5.** Pins Driven during Boot Program Execution

Peripheral	Pin	PIO Line
MCI	MCDA0	PIOA0
MCI	MCCDA	PIOA1
MCI	MCCK	PIOA2
MCI	MCDA1	PIOA3
MCI	MCDA2	PIOA4
MCI	MCDA3	PIOA5
SPI	MISO	PIOA25
SPI	MOSI	PIOA26
SPI	SPCK	PIOA27
SPI	NPCS0	PIOA28
PIO Controller B	NAND OE	PIOB4
PIO Controller B	NAND WE	PIOB5
PIO Controller B	NANDCS	PIOB6
Address Bus	NAND ALE	A21
Address Bus	NAND CLE	A22
DBGU	DRXD	PIOA21
DBGU	DTXD	PIOA22



## 15. Reset Controller (RSTC)

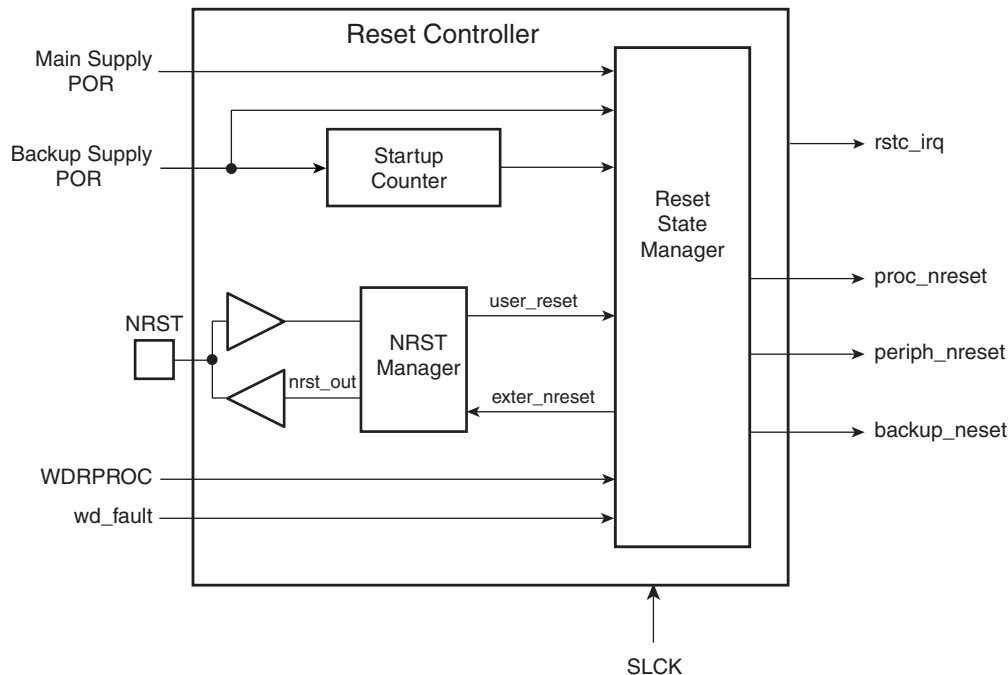
### 15.1 Description

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.

### 15.2 Block Diagram

Figure 15-1. Reset Controller Block Diagram



### 15.3 Functional Description

#### 15.3.1 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.

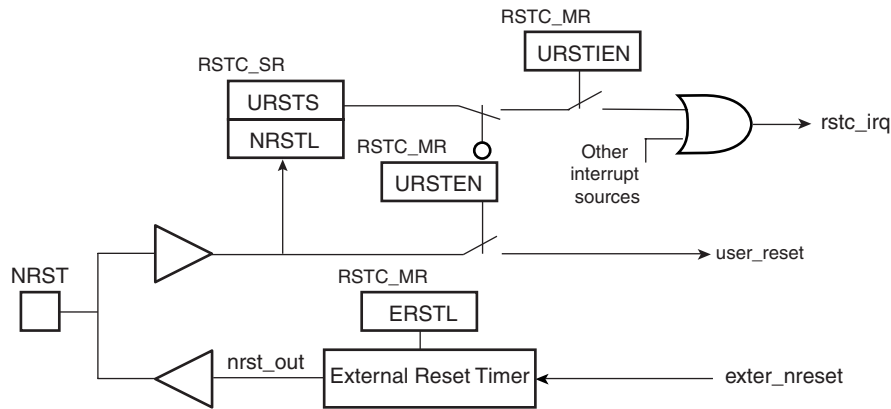
The startup counter waits for the complete crystal oscillator startup. The wait delay is given by the crystal oscillator startup time maximum value that can be found in the section Crystal Oscillator Characteristics in the Electrical Characteristics section of the product documentation.

The Reset Controller Mode Register (RSTC\_MR), allowing the configuration of the Reset Controller, is powered with VDDDBU, so that its configuration is saved as long as VDDDBU is on.

### 15.3.2 NRST Manager

The NRST Manager samples the NRST input pin and drives this pin low when required by the Reset State Manager. Figure 15-2 shows the block diagram of the NRST Manager.

Figure 15-2. NRST Manager



#### 15.3.2.1 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.

#### 15.3.2.2 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  $\mu$ s and 2 seconds. Note that ERSTL at 0 defines a two-cycle duration for the NRST pulse.

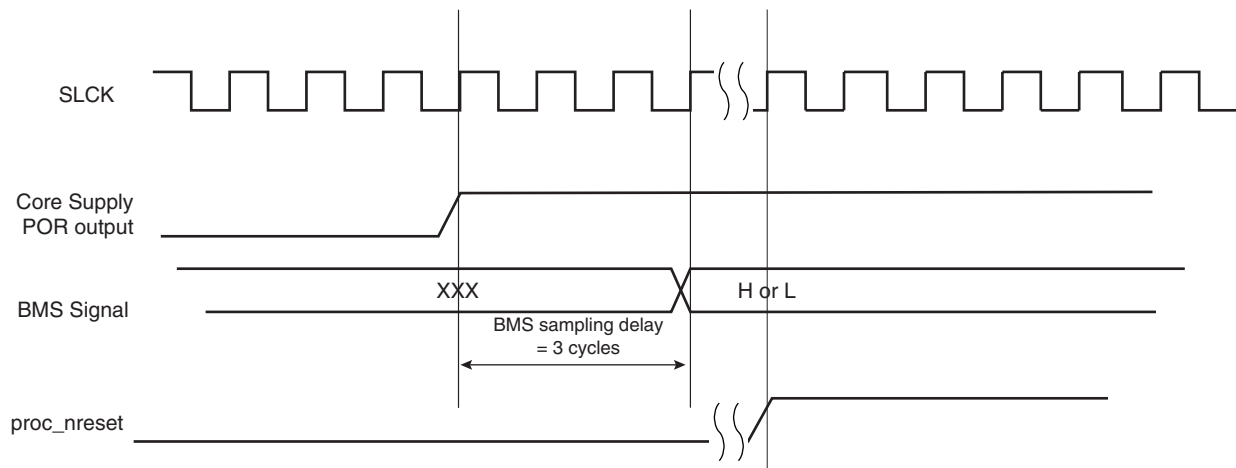
This feature allows the Reset Controller to shape the NRST pin level, and thus to guarantee that the NRST line is driven low for a time compliant with potential external devices connected on the system reset.

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.

### 15.3.3 BMS Sampling

The product matrix manages a boot memory that depends on the level on the BMS pin at reset. The BMS signal is sampled three slow clock cycles after the Core Power-On-Reset output rising edge.

**Figure 15-3.** BMS Sampling



### 15.3.4 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.

#### 15.3.4.1 General Reset

A general reset occurs when VDDBU and VDDCORE are 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 Slow Clock Oscillator startup time.

After this time, the processor clock is released at Slow Clock and all the other signals remain valid for 3 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.

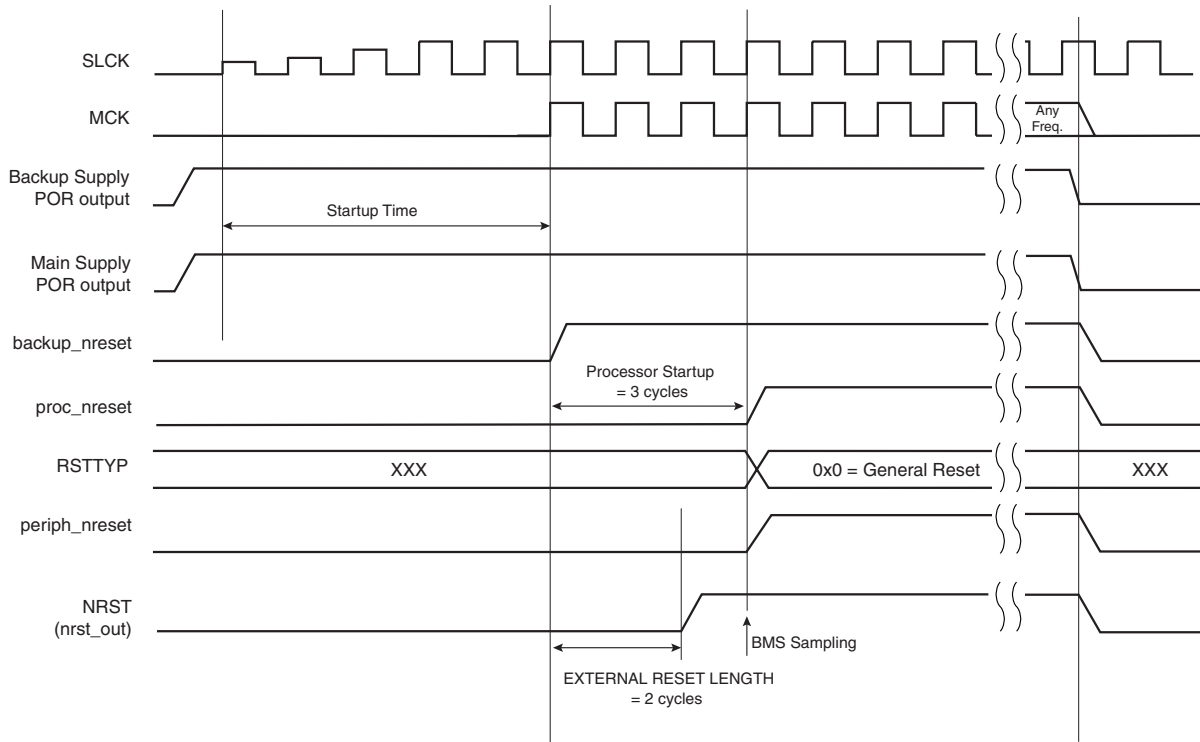
When VDDBU is detected low by the Backup Supply POR Cell, all resets signals are immediately asserted, even if the Main Supply POR Cell does not report a Main Supply shutdown.

VDDBU only activates the backup\_nreset signal.

The backup\_nreset must be released so that any other reset can be generated by VDDCORE (Main Supply POR output).

Figure 15-4 shows how the General Reset affects the reset signals.

**Figure 15-4. General Reset State**





## 15.3.4.2 Wake-up Reset

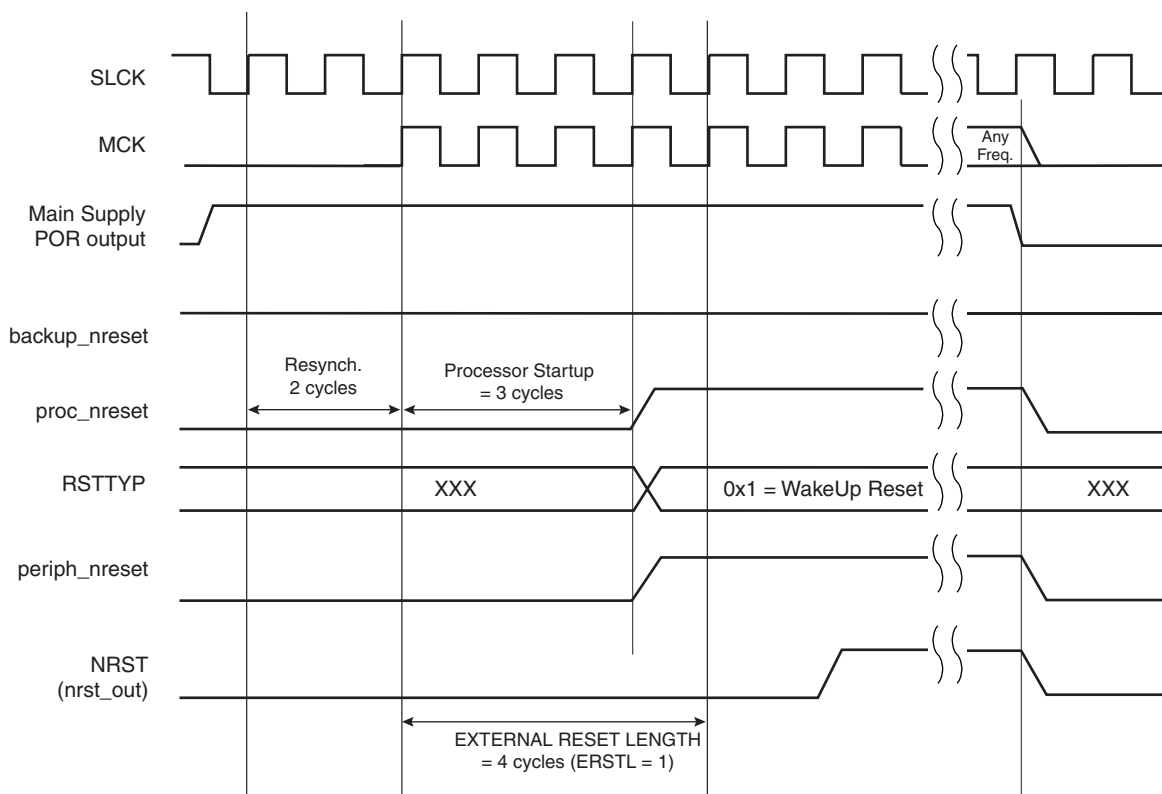
The Wake-up Reset occurs when the Main Supply is down. When the Main Supply POR output is active, all the reset signals are asserted except backup\_nreset. When the Main Supply powers up, the POR output is resynchronized on Slow Clock. The processor clock is then re-enabled during 3 Slow Clock cycles, 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 15-5.** Wake-up State



## 15.3.4.3 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.

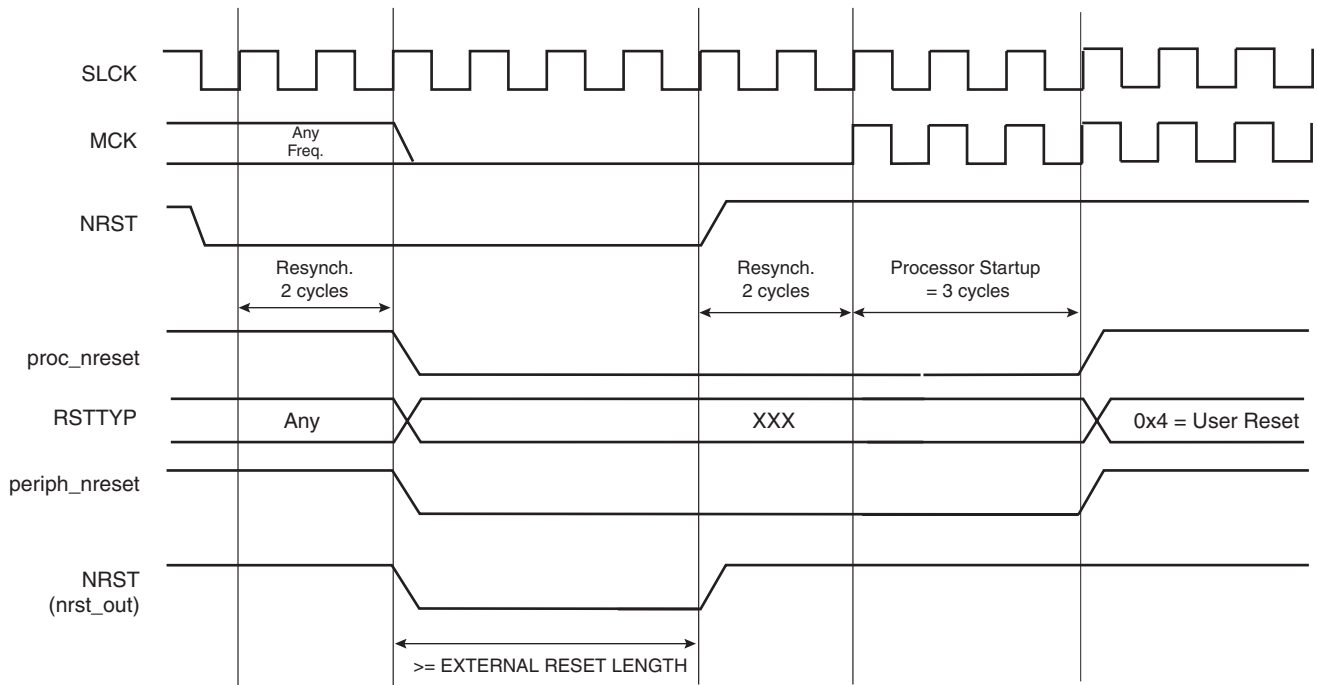
The User Reset is entered as soon as a low level is detected on NRST. The Processor Reset and the Peripheral Reset are asserted.

The User Reset is left when NRST rises, after a two-cycle resynchronization time and a 3-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 15-6. User Reset State**



#### 15.3.4.4 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 3 Slow Clock cycles.

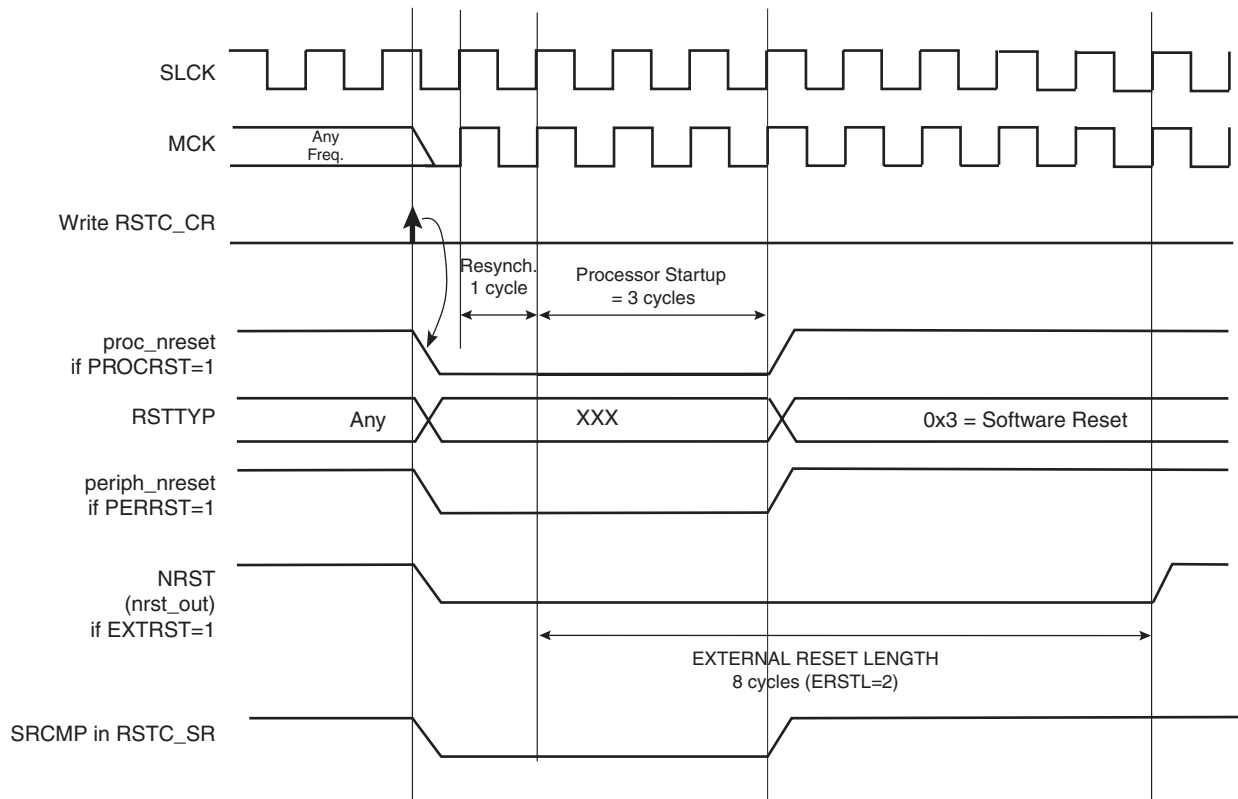
The internal reset signals are asserted as soon as the register write is performed. This is detected on the Master Clock (MCK). They are released when the software reset is left, i.e.; synchronously to SLCK.

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 15-7.** Software Reset



### 15.3.4.5 Watchdog Reset

The Watchdog Reset is entered when a watchdog fault occurs. This state lasts 3 Slow Clock cycles.

When in Watchdog Reset, assertion of the reset signals depends on the WDRPROC bit in WDT\_MR:

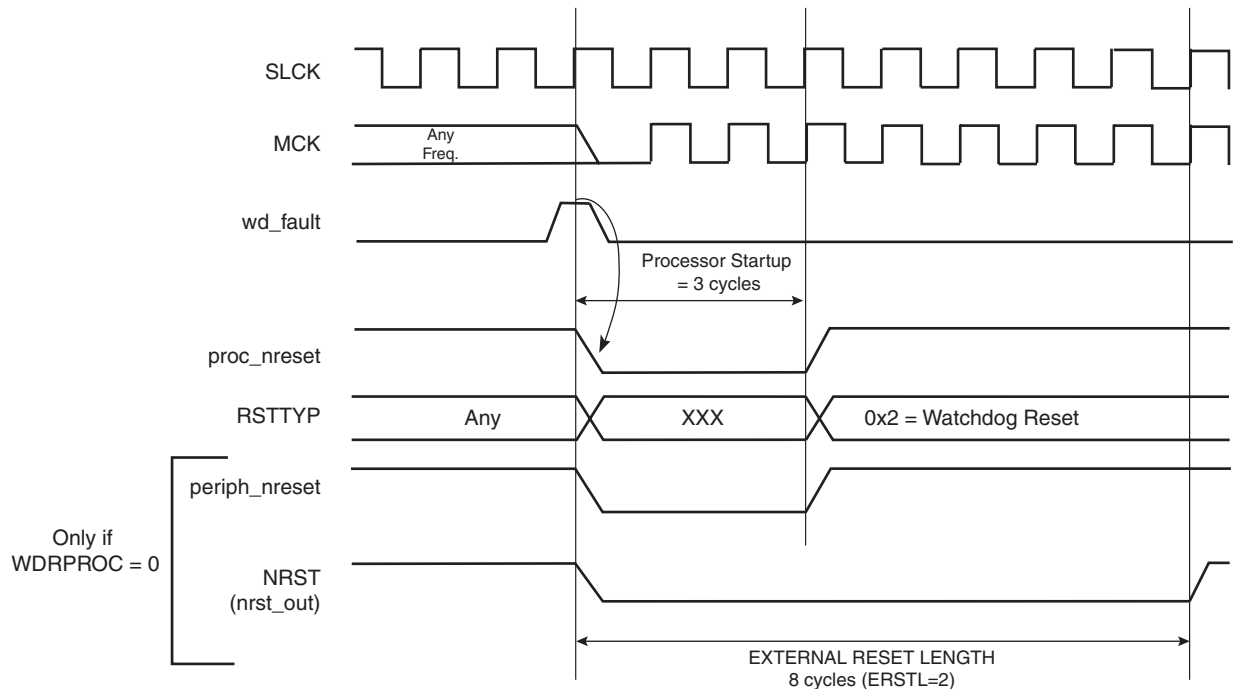
- If WDRPROC is 0, the Processor Reset and the Peripheral Reset are asserted. The NRST line is also asserted, depending on the programming of the field ERSTL. However, the resulting low level on NRST does not result in a User Reset state.

- 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 15-8.** Watchdog Reset



### 15.3.5 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.

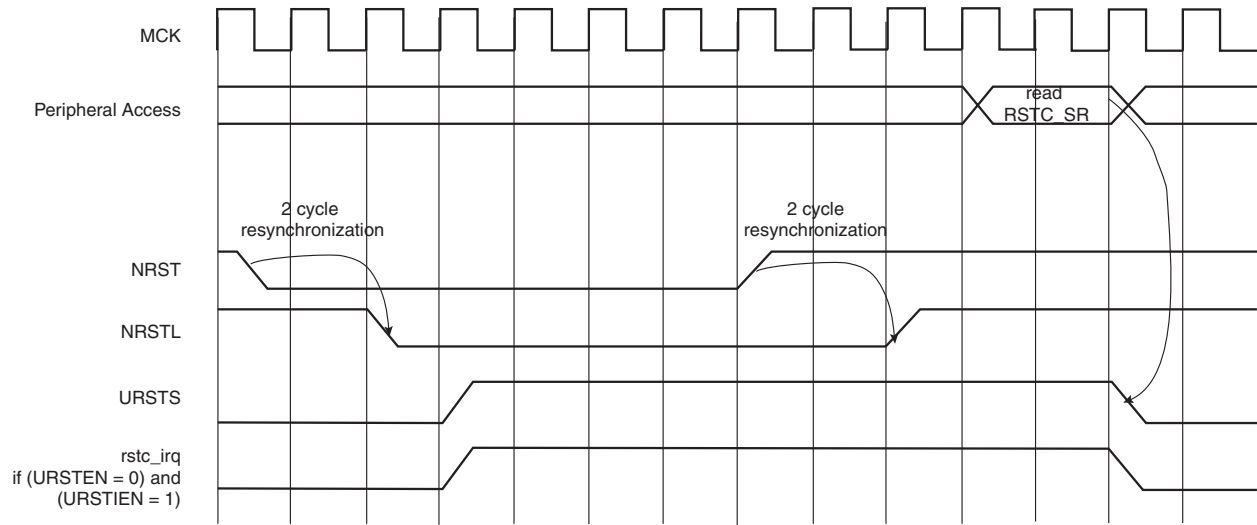
- When in Watchdog Reset:
  - The processor reset is active and so a Software Reset cannot be programmed.
  - A User Reset cannot be entered.

## 15.3.6 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 15-9](#)). 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 15-9.** Reset Controller Status and Interrupt



## 15.4 Reset Controller (RSTC) User Interface

**Table 15-1.** 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.

## 15.4.1 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.

### 15.4.2 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.



## 15.4.3 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  $\mu$ s and 2 seconds.

- **KEY: Password**

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



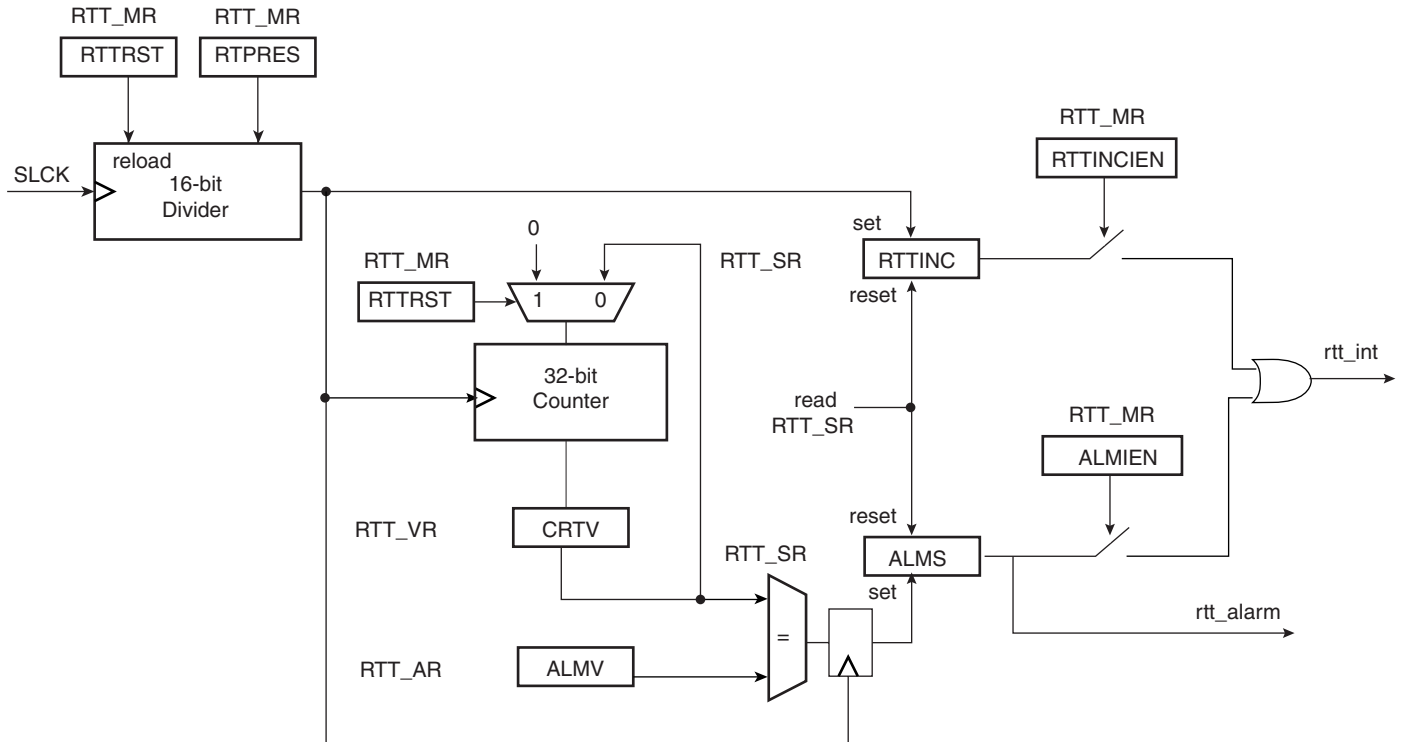
## 16. Real-time Timer (RTT)

### 16.1 Overview

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.

### 16.2 Block Diagram

Figure 16-1. Real-time Timer



### 16.3 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.

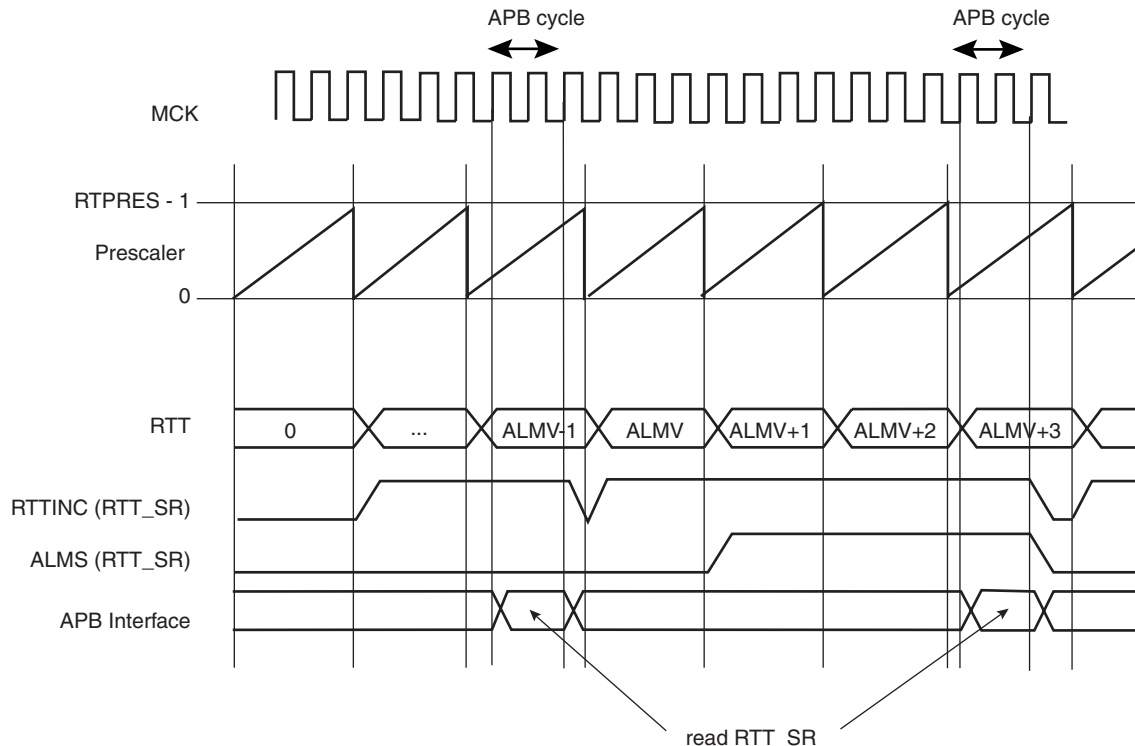
The bit RTTINC in RTT\_SR is set each time the Real-time Timer counter is incremented. This bit can be used to start a periodic interrupt, the period being one second when the RTPRES is programmed with 0x8000 and Slow Clock equal to 32.768 Hz.

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 16-2.** RTT Counting



**16.4 Real-time Timer (RTT) User Interface****16.4.1 Register Mapping****Table 16-1.** Real-time Timer Register Mapping

<b>Offset</b>	<b>Register</b>	<b>Name</b>	<b>Access</b>	<b>Reset Value</b>
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

### 16.4.2 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
–	–	–	–	–	RTRRST	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  $\neq$  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.

- **RTRRST: Real-time Timer Restart**

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

## 16.4.3 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.

## 16.4.4 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.



### 16.4.5 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 occurred since the last read of RTT\_SR.

1 = The Real-time Alarm occurred 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.



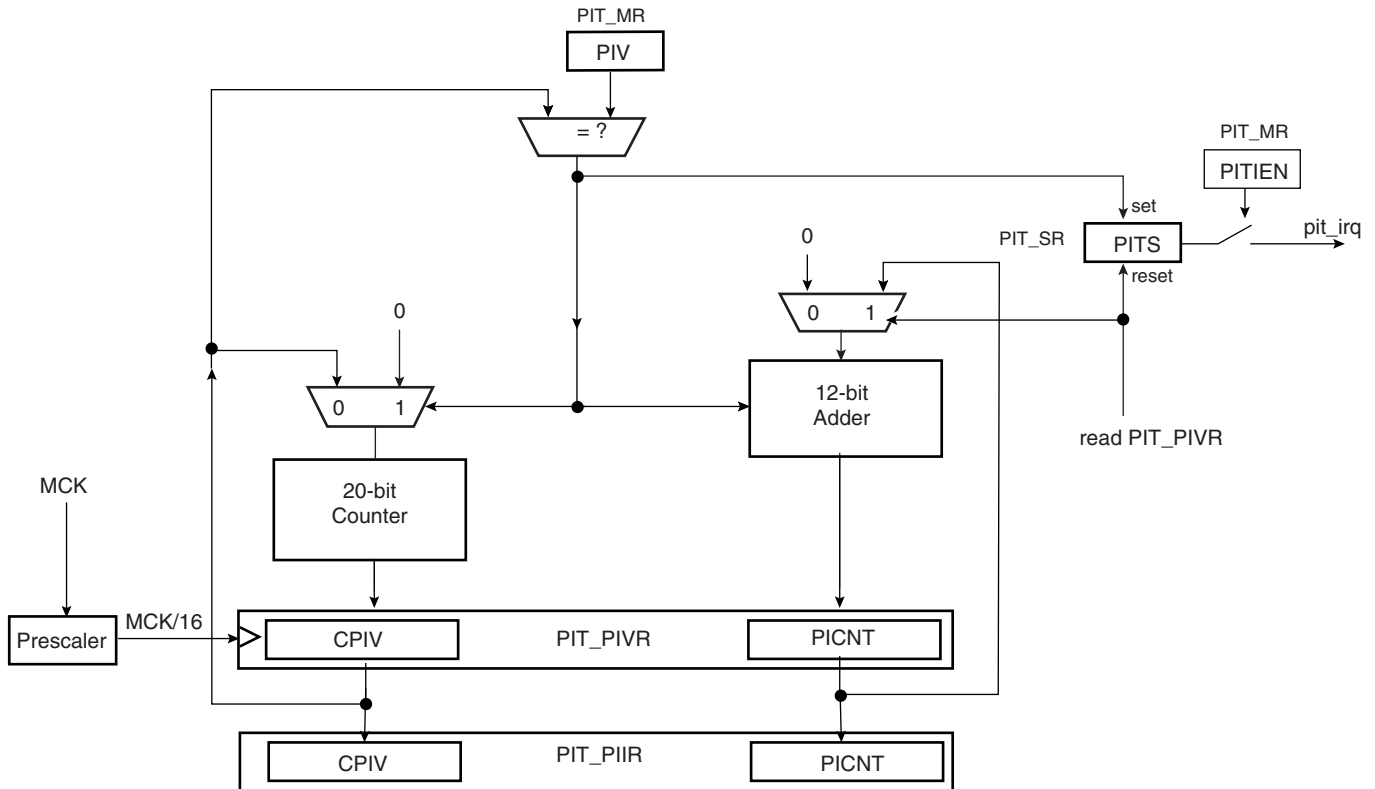
## 17. Periodic Interval Timer (PIT)

### 17.1 Overview

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.

### 17.2 Block Diagram

Figure 17-1. Periodic Interval Timer



## 17.3 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.

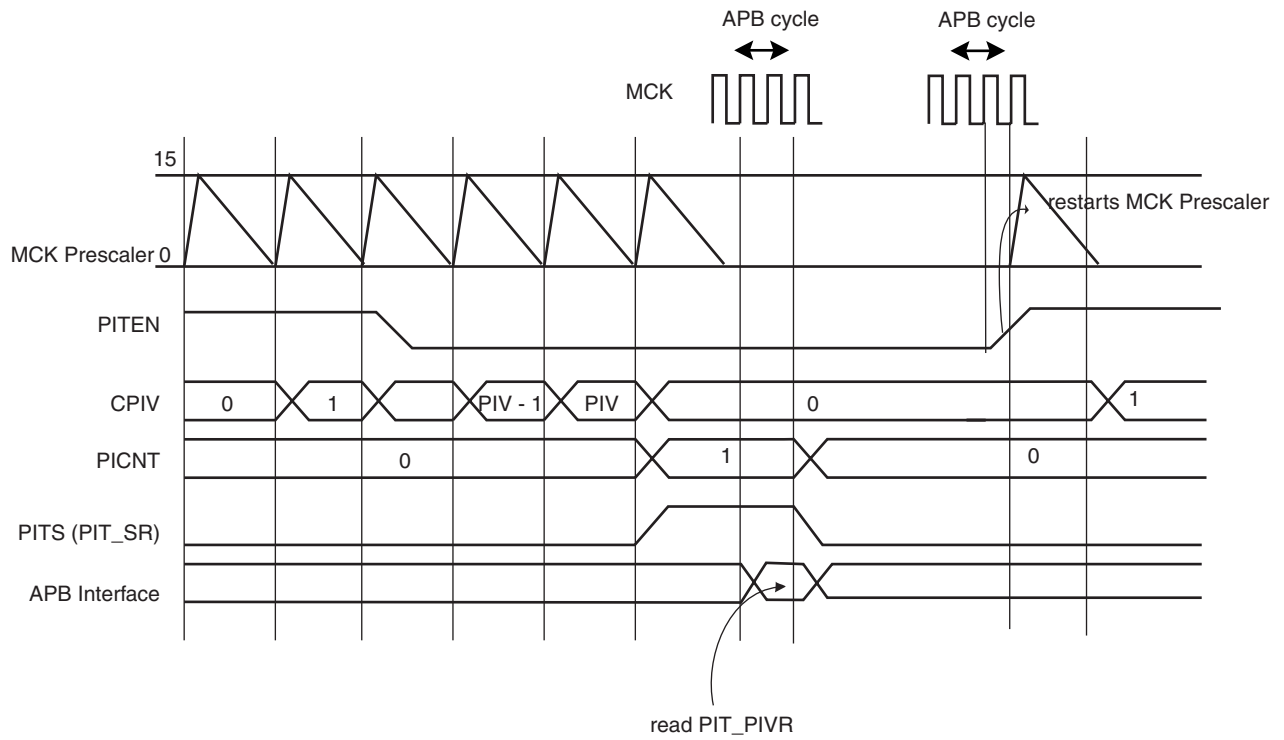
When CPIV and PICNT values are obtained by reading the Periodic Interval Value Register (PIT\_PIVR), the overflow counter (PICNT) is reset and the PITS is cleared, thus acknowledging the interrupt. The value of PICNT gives the number of periodic intervals elapsed since the last read of PIT\_PIVR.

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 17-2](#) 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 17-2.** Enabling/Disabling PIT with PITEN



## 17.4 Periodic Interval Timer (PIT) User Interface

**Table 17-1.** 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

## 17.4.1 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.

## 17.4.2 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**

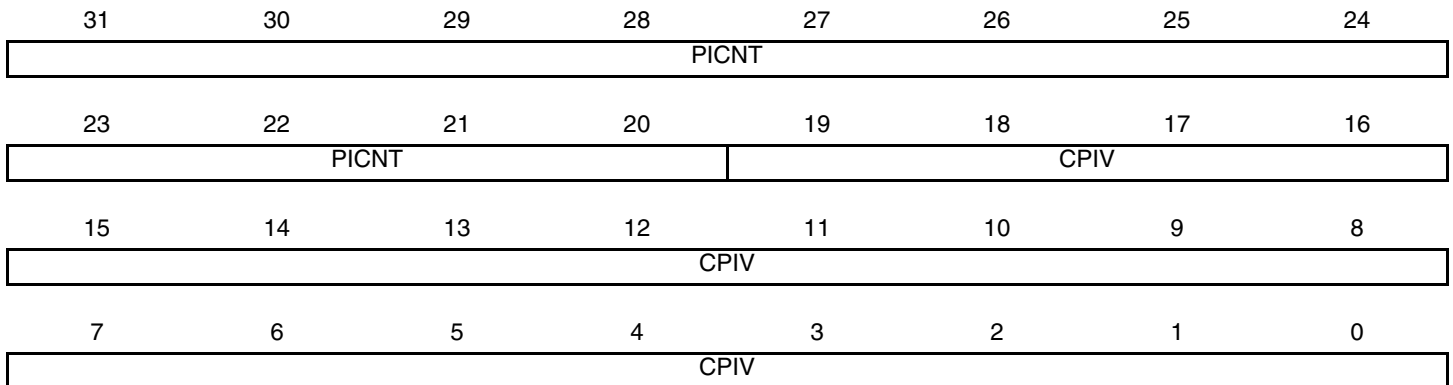
0 = The Periodic Interval timer has not reached PIV since the last read of PIT\_PIVR.

1 = The Periodic Interval timer has reached PIV since the last read of PIT\_PIVR.

### 17.4.3 Periodic Interval Timer Value Register

**Register Name:** PIT\_PIVR

**Access Type:** Read-only



Reading this register clears PITS in PIT\_SR.

- **CPIV: Current Periodic Interval Value**

Returns the current value of the periodic interval timer.

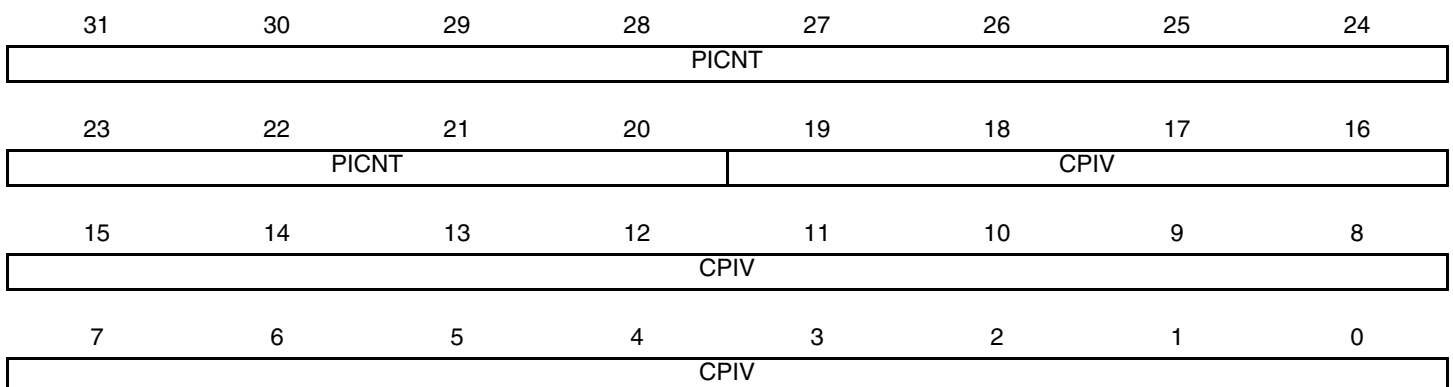
- **PICNT: Periodic Interval Counter**

Returns the number of occurrences of periodic intervals since the last read of PIT\_PIVR.

### 17.4.4 Periodic Interval Timer Image Register

**Register Name:** PIT\_PIIIR

**Access Type:** Read-only



- **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.

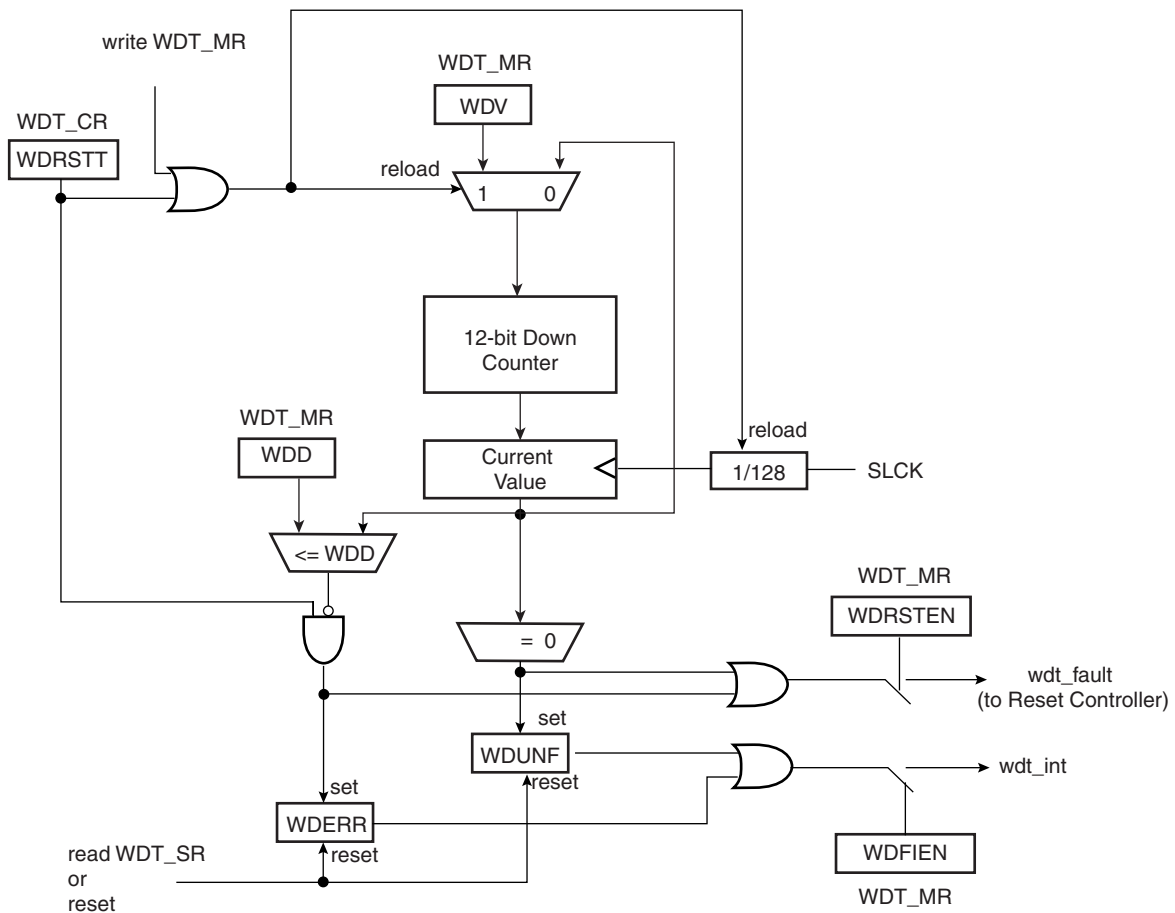
## 18. Watchdog Timer (WDT)

### 18.1 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.

### 18.2 Block Diagram

Figure 18-1. Watchdog Timer Block Diagram



## 18.3 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 0xFFFF, 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.

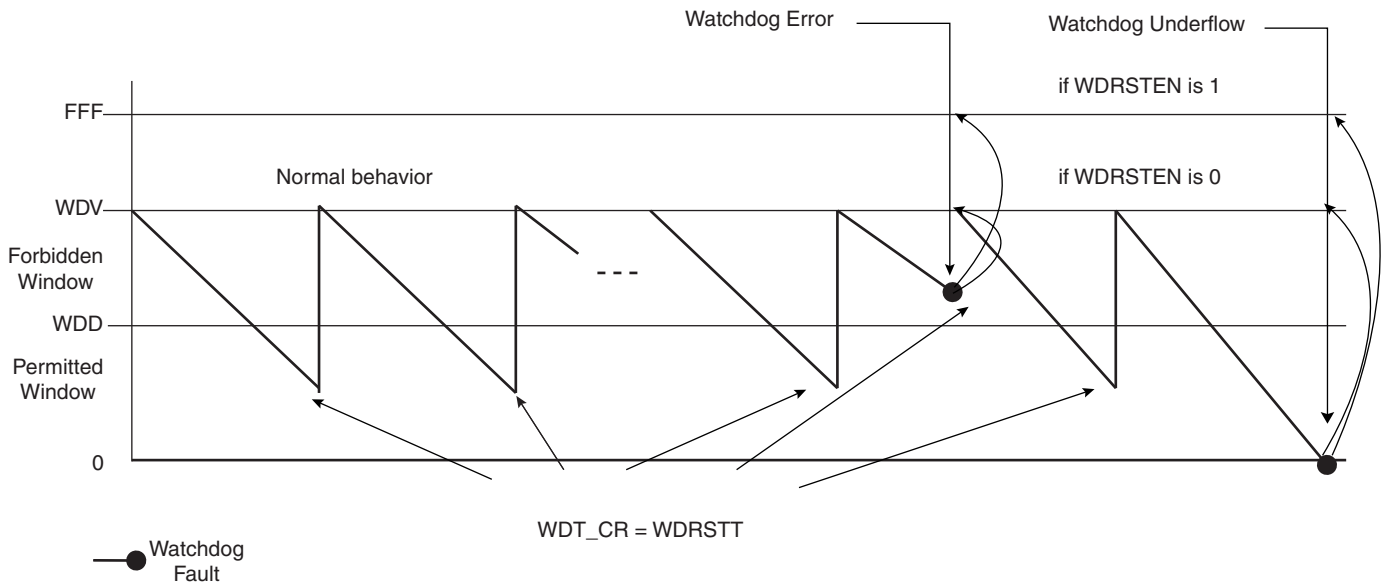
If a reset is generated or if WDT\_SR is read, the status bits are reset, the interrupt is cleared, and the “wdt\_fault” signal to the reset controller is deasserted.

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 18-2. Watchdog Behavior**



## 18.4 Watchdog Timer (WDT) User Interface

**Table 18-1.** 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

### 18.4.1 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.

### 18.4.2 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.

### 18.4.3 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.



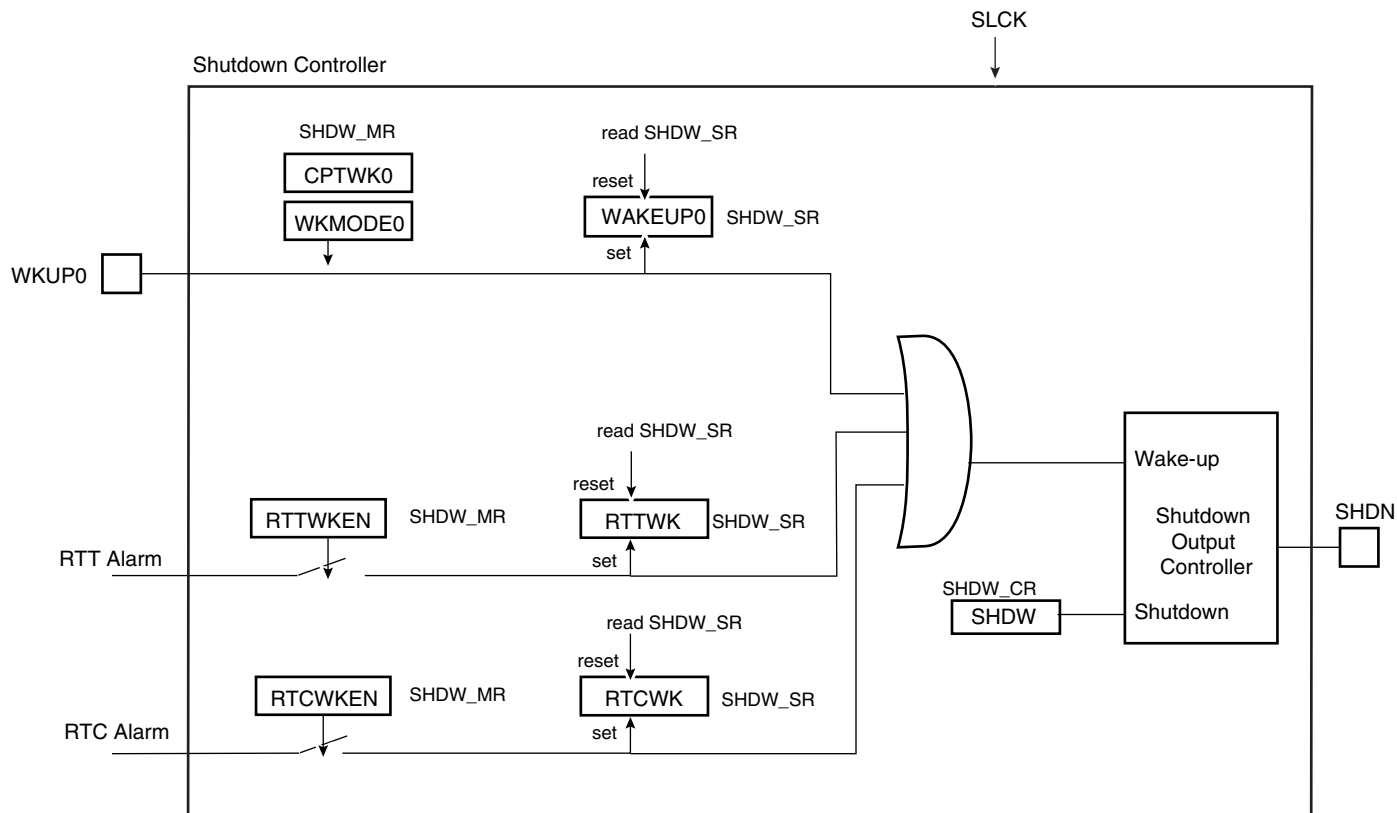
## 19. Shutdown Controller (SHDWC)

### 19.1 Description

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

### 19.2 Block Diagram

Figure 19-1. Shutdown Controller Block Diagram



### 19.3 I/O Lines Description

Table 19-1. I/O Lines Description

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

### 19.4 Product Dependencies

#### 19.4.1 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.

## 19.5 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.

The Shutdown Controller can be programmed so as to activate the wake-up using the RTC alarm (the detection of the rising edge of the RTC alarm is synchronized with SLCK). This is done by writing the SHDW\_MR register using the RTCWKEN field. When enabled, the detection of the RTC alarm is reported in the RTCWK bit of the SHDW\_SR Status register. It is reset after the read of SHDW\_SR. When using the RTC alarm to wake up the system, the user must ensure that the RTC 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.



## 19.6 Shutdown Controller (SHDWC) User Interface

**Table 19-2.** Register Mapping

Offset	Register	Name	Access	Reset
0x00	Shutdown Control Register	SHDW_CR	Write-only	-
0x04	Shutdown Mode Register	SHDW_MR	Read-write	0x0000_0003
0x08	Shutdown Status Register	SHDW_SR	Read-only	0x0000_0000

### 19.6.1 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.

## 19.6.2 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
–	–	–	–	–	–	RTCWKEN	RTTWKEN
15	14	13	12	11	10	9	8
–				–	–	–	
7	6	5	4	3	2	1	0
CPTWK0				–	–	WKMODE0	

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

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.

- **RTCWKEN: Real-time Clock Wake-up Enable**

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

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

## 19.6.3 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
–	–	–	–	–	–	RTCWK	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.

- **RTCWK: Real-time Clock Wake-up**

0 = No wake-up alarm from the RTC occurred since the last read of SHDW\_SR.

1 = At least one wake-up alarm from the RTC occurred since the last read of SHDW\_SR.



## 20. Real-time Clock (RTC)

### 20.1 Description

The Real-time Clock (RTC) peripheral is designed for very low power consumption.

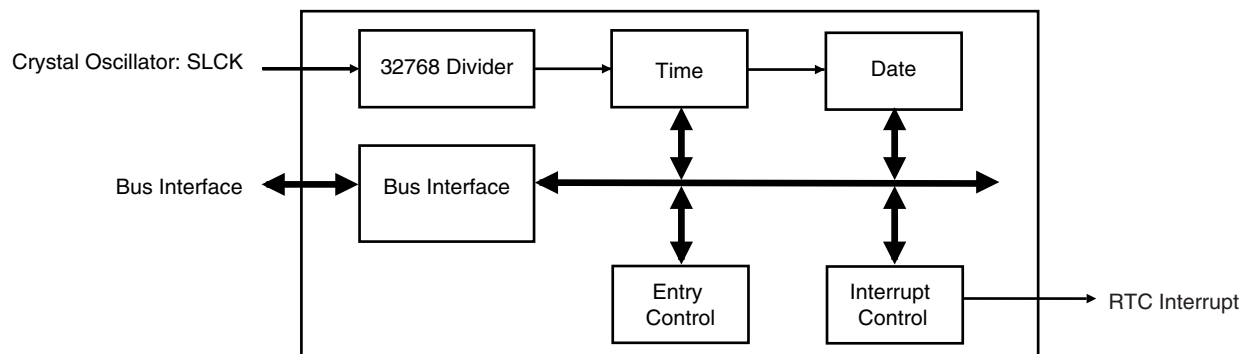
It combines a complete time-of-day clock with alarm and a two-hundred-year Gregorian calendar, complemented by a programmable periodic interrupt. The alarm and calendar registers are accessed by a 32-bit data bus.

The time and calendar values are coded in binary-coded decimal (BCD) format. The time format can be 24-hour mode or 12-hour mode with an AM/PM indicator.

Updating time and calendar fields and configuring the alarm fields are performed by a parallel capture on the 32-bit data bus. An entry control is performed to avoid loading registers with incompatible BCD format data or with an incompatible date according to the current month/year/century.

### 20.2 Block Diagram

Figure 20-1. Block Diagram



### 20.3 Product Dependencies

#### 20.3.1 Power Management

The Real-time Clock is continuously clocked at 32768 Hz. The Power Management Controller has no effect on RTC behavior.

#### 20.3.2 Interrupt

The RTC Interrupt is connected to interrupt source 1 (IRQ1) of the advanced interrupt controller. This interrupt line is due to the OR-wiring of the system peripheral interrupt lines (System Timer, Real Time Clock, Power Management Controller, Memory Controller, etc.). When a system interrupt occurs, the service routine must first determine the cause of the interrupt. This is done by reading the status registers of the above system peripherals successively.

### 20.4 Functional Description

The RTC provides a full binary-coded decimal (BCD) clock that includes century (19/20), year (with leap years), month, date, day, hours, minutes and seconds.

The valid year range is 1900 to 2099, a two-hundred-year Gregorian calendar achieving full Y2K compliance.

The RTC can operate in 24-hour mode or in 12-hour mode with an AM/PM indicator.

Corrections for leap years are included (all years divisible by 4 being leap years, including year 2000). This is correct up to the year 2099.

After hardware reset, the calendar is initialized to Thursday, January 1, 1998.

#### 20.4.1 Reference Clock

The reference clock is Slow Clock (SLCK).

During low power modes of the processor (idle mode), the oscillator runs and power consumption is critical. The crystal selection has to take into account the current consumption for power saving and the frequency drift due to temperature effect on the circuit for time accuracy.

#### 20.4.2 Timing

The RTC is updated in real time at one-second intervals in normal mode for the counters of seconds, at one-minute intervals for the counter of minutes and so on.

Due to the asynchronous operation of the RTC with respect to the rest of the chip, to be certain that the value read in the RTC registers (century, year, month, date, day, hours, minutes, seconds) are valid and stable, it is necessary to read these registers twice. If the data is the same both times, then it is valid. Therefore, a minimum of two and a maximum of three accesses are required.

#### 20.4.3 Alarm

The RTC has five programmable fields: month, date, hours, minutes and seconds.

Each of these fields can be enabled or disabled to match the alarm condition:

- If all the fields are enabled, an alarm flag is generated (the corresponding flag is asserted and an interrupt generated if enabled) at a given month, date, hour/minute/second.
- If only the “seconds” field is enabled, then an alarm is generated every minute.

Depending on the combination of fields enabled, a large number of possibilities are available to the user ranging from minutes to 365/366 days.

#### 20.4.4 Error Checking

Verification on user interface data is performed when accessing the century, year, month, date, day, hours, minutes, seconds and alarms. A check is performed on illegal BCD entries such as illegal date of the month with regard to the year and century configured.

If one of the time fields is not correct, the data is not loaded into the register/counter and a flag is set in the validity register. The user can not reset this flag. It is reset as soon as an acceptable value is programmed. This avoids any further side effects in the hardware. The same procedure is done for the alarm.

The following checks are performed:

1. Century (check if it is in range 19 - 20)
2. Year (BCD entry check)
3. Date (check range 01 - 31)
4. Month (check if it is in BCD range 01 - 12, check validity regarding “date”)

5. Day (check range 1 - 7)
6. Hour (BCD checks: in 24-hour mode, check range 00 - 23 and check that AM/PM flag is not set if RTC is set in 24-hour mode; in 12-hour mode check range 01 - 12)
7. Minute (check BCD and range 00 - 59)
8. Second (check BCD and range 00 - 59)

Note: If the 12-hour mode is selected by means of the RTC\_MODE register, a 12-hour value can be programmed and the returned value on RTC\_TIME will be the corresponding 24-hour value. The entry control checks the value of the AM/PM indicator (bit 22 of RTC\_TIME register) to determine the range to be checked.

## 20.4.5 Updating Time/Calendar

To update any of the time/calendar fields, the user must first stop the RTC by setting the corresponding field in the Control Register. Bit UPDTIM must be set to update time fields (hour, minute, second) and bit UPDCAL must be set to update calendar fields (century, year, month, date, day).

Then the user must poll or wait for the interrupt (if enabled) of bit ACKUPD in the Status Register. Once the bit reads 1, the user can write to the appropriate register.

Once the update is finished, the user must reset (0) UPDTIM and/or UPDCAL in the Control Register.

When programming the calendar fields, the time fields remain enabled. This avoids a time slip in case the user stays in the calendar update phase for several tens of seconds or more. In successive update operations, the user must wait at least one second after resetting the UPDTIM/UPDCAL bit in the RTC\_CR (Control Register) before setting these bits again. This is done by waiting for the SEC flag in the Status Register before setting UPDTIM/UPDCAL bit. After resetting UPDTIM/UPDCAL, the SEC flag must also be cleared.

## 20.5 Real-time Clock (RTC) User Interface

Table 20-1. Register Mapping

Offset	Register	Register Name	Read/Write	Reset
0x00	Control Register	RTC_CR	Read/Write	0x0
0x04	Mode Register	RTC_MR	Read/Write	0x0
0x08	Time Register	RTC_TIMR	Read/Write	0x0
0x0C	Calendar Register	RTC_CALR	Read/Write	0x01819819
0x10	Time Alarm Register	RTC_TIMALR	Read/Write	0x0
0x14	Calendar Alarm Register	RTC_CALALR	Read/Write	0x01010000
0x18	Status Register	RTC_SR	Read-only	0x0
0x1C	Status Clear Command Register	RTC_SCCR	Write-only	---
0x20	Interrupt Enable Register	RTC_IER	Write-only	---
0x24	Interrupt Disable Register	RTC_IDR	Write-only	---
0x28	Interrupt Mask Register	RTC_IMR	Read-only	0x0
0x2C	Valid Entry Register	RTC_VER	Read-only	0x0
0xFC	Version Register <sup>(1)</sup>	RTC-VERSION	Read-only	0x-

Note: 1. Values in the Version Register vary with the version of the IP block implementation.





## 20.5.1 RTC Control Register

**Name:** RTC\_CR

**Access Type:** Read/Write

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

- **UPDTIM: Update Request Time Register**

0 = No effect.

1 = Stops the RTC time counting.

Time counting consists of second, minute and hour counters. Time counters can be programmed once this bit is set and acknowledged by the bit ACKUPD of the Status Register.

- **UPDCAL: Update Request Calendar Register**

0 = No effect.

1 = Stops the RTC calendar counting.

Calendar counting consists of day, date, month, year and century counters. Calendar counters can be programmed once this bit is set.

- **TIMEVSEL: Time Event Selection**

The event that generates the flag TIMEV in RTC\_SR (Status Register) depends on the value of TIMEVSEL.

0 = Minute change.

1 = Hour change.

2 = Every day at midnight.

3 = Every day at noon.

- **CALEVSEL: Calendar Event Selection**

The event that generates the flag CALEV in RTC\_SR depends on the value of CALEVSEL.

0 = Week change (every Monday at time 00:00:00).

1 = Month change (every 01 of each month at time 00:00:00).

2, 3 = Year change (every January 1 at time 00:00:00).

## 20.5.2 RTC Mode Register

**Name:** RTC\_MR

**Access Type:** Read/Write

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

- **HRMOD: 12-/24-hour Mode**

0 = 24-hour mode is selected.

1 = 12-hour mode is selected.

All non-significant bits read zero.

## 20.5.3 RTC Time Register

**Name:** RTC\_TIMR

**Access Type:** Read/Write

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

- **SEC: Current Second**

The range that can be set is 0 - 59 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **MIN: Current Minute**

The range that can be set is 0 - 59 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

- **HOUR: Current Hour**

The range that can be set is 1 - 12 (BCD) in 12-hour mode or 0 - 23 (BCD) in 24-hour mode.

- **AMPM: Ante Meridiem Post Meridiem Indicator**

This bit is the AM/PM indicator in 12-hour mode.

0 = AM.

1 = PM.

All non-significant bits read zero.



### 20.5.4 RTC Calendar Register

Name: RTC\_CALR

Access Type: Read/Write

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

• **CENT: Current Century**

The range that can be set is 19 - 20 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

• **YEAR: Current Year**

The range that can be set is 00 - 99 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

• **MONTH: Current Month**

The range that can be set is 01 - 12 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

• **DAY: Current Day**

The range that can be set is 1 - 7 (BCD).

The coding of the number (which number represents which day) is user-defined as it has no effect on the date counter.

• **DATE: Current Date**

The range that can be set is 01 - 31 (BCD).

The lowest four bits encode the units. The higher bits encode the tens.

All non-significant bits read zero.

## 20.5.5 RTC Time Alarm Register

**Name:** RTC\_TIMALR

**Access Type:** Read/Write

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

- **SEC: Second Alarm**

This field is the alarm field corresponding to the BCD-coded second counter.

- **SECEN: Second Alarm Enable**

0 = The second-matching alarm is disabled.

1 = The second-matching alarm is enabled.

- **MIN: Minute Alarm**

This field is the alarm field corresponding to the BCD-coded minute counter.

- **MINEN: Minute Alarm Enable**

0 = The minute-matching alarm is disabled.

1 = The minute-matching alarm is enabled.

- **HOUR: Hour Alarm**

This field is the alarm field corresponding to the BCD-coded hour counter.

- **AMPM: AM/PM Indicator**

This field is the alarm field corresponding to the BCD-coded hour counter.

- **HOUREN: Hour Alarm Enable**

0 = The hour-matching alarm is disabled.

1 = The hour-matching alarm is enabled.



### 20.5.6 RTC Calendar Alarm Register

Name: RTC\_CALALR

Access Type: Read/Write

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

• **MONTH: Month Alarm**

This field is the alarm field corresponding to the BCD-coded month counter.

• **MTHEN: Month Alarm Enable**

0 = The month-matching alarm is disabled.

1 = The month-matching alarm is enabled.

• **DATE: Date Alarm**

This field is the alarm field corresponding to the BCD-coded date counter.

• **DATEEN: Date Alarm Enable**

0 = The date-matching alarm is disabled.

1 = The date-matching alarm is enabled.

## 20.5.7 RTC Status Register

**Name:** RTC\_SR

**Access Type:** Read-only

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

- **ACKUPD: Acknowledge for Update**

0 = Time and calendar registers cannot be updated.

1 = Time and calendar registers can be updated.

- **ALARM: Alarm Flag**

0 = No alarm matching condition occurred.

1 = An alarm matching condition has occurred.

- **SEC: Second Event**

0 = No second event has occurred since the last clear.

1 = At least one second event has occurred since the last clear.

- **TIMEV: Time Event**

0 = No time event has occurred since the last clear.

1 = At least one time event has occurred since the last clear.

The time event is selected in the TIMEVSEL field in RTC\_CTRL (Control Register) and can be any one of the following events: minute change, hour change, noon, midnight (day change).

- **CALEV: Calendar Event**

0 = No calendar event has occurred since the last clear.

1 = At least one calendar event has occurred since the last clear.

The calendar event is selected in the CALEVSEL field in RTC\_CR and can be any one of the following events: week change, month change and year change.

## 20.5.8 RTC Status Clear Command Register

**Name:** RTC\_SCCR

**Access Type:** Write-only

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

- **ACKCLR: Acknowledge Clear**

0 = No effect.

1 = Clears corresponding status flag in the Status Register (RTC\_SR).

- **ALRCLR: Alarm Clear**

0 = No effect.

1 = Clears corresponding status flag in the Status Register (RTC\_SR).

- **SECCLR: Second Clear**

0 = No effect.

1 = Clears corresponding status flag in the Status Register (RTC\_SR).

- **TIMCLR: Time Clear**

0 = No effect.

1 = Clears corresponding status flag in the Status Register (RTC\_SR).

- **CALCLR: Calendar Clear**

0 = No effect.

1 = Clears corresponding status flag in the Status Register (RTC\_SR).



## 20.5.9 RTC Interrupt Enable Register

**Name:** RTC\_IER

**Access Type:** Write-only

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

- **ACKEN: Acknowledge Update Interrupt Enable**

0 = No effect.

1 = The acknowledge for update interrupt is enabled.

- **ALREN: Alarm Interrupt Enable**

0 = No effect.

1 = The alarm interrupt is enabled.

- **SECEN: Second Event Interrupt Enable**

0 = No effect.

1 = The second periodic interrupt is enabled.

- **TIMEN: Time Event Interrupt Enable**

0 = No effect.

1 = The selected time event interrupt is enabled.

- **CALEN: Calendar Event Interrupt Enable**

0 = No effect.

- **1 = The selected calendar event interrupt is enabled.**

### 20.5.10 RTC Interrupt Disable Register

**Name:** RTC\_IDR

**Access Type:** Write-only

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

- **ACKDIS: Acknowledge Update Interrupt Disable**

0 = No effect.

1 = The acknowledge for update interrupt is disabled.

- **ALRDIS: Alarm Interrupt Disable**

0 = No effect.

1 = The alarm interrupt is disabled.

- **SECDIS: Second Event Interrupt Disable**

0 = No effect.

1 = The second periodic interrupt is disabled.

- **TIMDIS: Time Event Interrupt Disable**

0 = No effect.

1 = The selected time event interrupt is disabled.

- **CALDIS: Calendar Event Interrupt Disable**

0 = No effect.

1 = The selected calendar event interrupt is disabled.

## 20.5.11 RTC Interrupt Mask Register

**Name:** RTC\_IMR

**Access Type:** Read-only

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

- **ACK: Acknowledge Update Interrupt Mask**

0 = The acknowledge for update interrupt is disabled.

1 = The acknowledge for update interrupt is enabled.

- **ALR: Alarm Interrupt Mask**

0 = The alarm interrupt is disabled.

1 = The alarm interrupt is enabled.

- **SEC: Second Event Interrupt Mask**

0 = The second periodic interrupt is disabled.

1 = The second periodic interrupt is enabled.

- **TIM: Time Event Interrupt Mask**

0 = The selected time event interrupt is disabled.

1 = The selected time event interrupt is enabled.

- **CAL: Calendar Event Interrupt Mask**

0 = The selected calendar event interrupt is disabled.

1 = The selected calendar event interrupt is enabled.

### 20.5.12 RTC Valid Entry Register

**Name:** RTC\_VER

**Access Type:** Read-only

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

- **NVTIM: Non valid Time**

0 = No invalid data has been detected in RTC\_TIMR (Time Register).

1 = RTC\_TIMR has contained invalid data since it was last programmed.

- **NVCAL: Non valid Calendar**

0 = No invalid data has been detected in RTC\_CALR (Calendar Register).

1 = RTC\_CALR has contained invalid data since it was last programmed.

- **NVTIMALR: Non valid Time Alarm**

0 = No invalid data has been detected in RTC\_TIMALR (Time Alarm Register).

1 = RTC\_TIMALR has contained invalid data since it was last programmed.

- **NVCALALR: Non valid Calendar Alarm**

0 = No invalid data has been detected in RTC\_CALALR (Calendar Alarm Register).

1 = RTC\_CALALR has contained invalid data since it was last programmed.

## 21. AT91SAM9R64/RL64 Bus Matrix

### 21.1 Description

Bus Matrix implements a multi-layer AHB, based on AHB-Lite protocol, that enables parallel access paths between multiple AHB masters and slaves in a system, which increases the overall bandwidth. Bus Matrix interconnects 6 AHB Masters to 6 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 Advanced Peripheral Bus and provides a Chip Configuration User Interface with Registers that allow the Bus Matrix to support application specific features.

### 21.2 Memory Mapping

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 allows to perform remap action for every master independently.

### 21.3 Special Bus Granting Techniques

The Bus Matrix provides some speculative bus granting techniques in order to anticipate access requests from some masters. This mechanism allows to reduce latency at first accesses of a burst or single transfer. The bus granting mechanism allows to set a 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.

#### 21.3.1 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.

#### 21.3.2 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.

#### 21.3.3 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 doesn't 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 allow to 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 allows to choose the default master type (no default, last access master, fixed default master) whereas the 4-bit

FIXED\_DEFMSTR field allows to choose a fixed default master provided that DEFMSTR\_TYPE is set to fixed default master. Please refer to the Bus Matrix user interface description.

## 21.4 Arbitration

The Bus Matrix provides an arbitration mechanism that allows to reduce latency when conflict cases occur, basically when two or more masters try to access the same slave at the same time. One arbiter per AHB slave is provided, allowing to arbitrate each slave differently.

The Bus Matrix provides to the user the possibility to choose between 2 arbitration types, and this for each slave:

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

This choice is given through 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 has to be done, it is realized only under some specific conditions detailed in the following paragraph.

### 21.4.1 Arbitration rules

Each arbiter has the ability to arbitrate between two or more different master's 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 iv.).
4. Slot Cycle Limit: when the slot cycle counter has reach the limit value indicating that the current master access is too long and must be broken (See "Slot Cycle Limit Arbitration" on page iv.).

#### 21.4.1.1 Undefined Length Burst Arbitration

In order to avoid too 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 for defined length burst transfer, which is selected between the following:

1. Infinite: no predicted end of burst is generated and therefore INCR burst transfer will never be broken.
2. Four beat bursts: predicted end of burst is generated at the end of each four beat boundary inside INCR transfer.
3. Eight beat bursts: predicted end of burst is generated at the end of each eight beat boundary inside INCR transfer.
4. 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).

## 21.4.1.2 Slot Cycle Limit Arbitration

The Bus Matrix contains specific logic to break too 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.

## 21.4.2 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's 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 algorithm implemented:

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

### 21.4.2.1 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.

### 21.4.2.2 Round-Robin Arbitration with Last Access 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 performs the access. Other non privileged masters will 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.

### 21.4.2.3 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.

## 21.4.3 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's requests are active at the same time, the master with the highest priority number is serviced first. If two or



more master's 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).



## 21.5 Bus Matrix User Interface

**Table 21-1.** Register Mapping

Offset	Register	Name	Access	Reset Value
0x0000	Master Configuration Register 0	MATRIX_MCFG0	Read/Write	0x00000000
0x0004	Master Configuration Register 1	MATRIX_MCFG1	Read/Write	0x00000000
0x0008	Master Configuration Register 2	MATRIX_MCFG2	Read/Write	0x00000000
0x000C	Master Configuration Register 3	MATRIX_MCFG3	Read/Write	0x00000000
0x0010	Master Configuration Register 4	MATRIX_MCFG4	Read/Write	0x00000000
0x0014	Master Configuration Register 5	MATRIX_MCFG5	Read/Write	0x00000000
0x0018 - 0x003C	Reserved	-	-	-
0x0040	Slave Configuration Register 0	MATRIX_SCFG0	Read/Write	0x00010010
0x0044	Slave Configuration Register 1	MATRIX_SCFG1	Read/Write	0x00050010
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 - 0x007C	Reserved	-	-	-
0x0080	Priority Register A for Slave 0	MATRIX_PRAS0	Write	0x00000000
0x0084	Reserved	-	-	-
0x0088	Priority Register A for Slave 1	MATRIX_PRAS1	Write	0x00000000
0x008C	Reserved	-	-	-
0x0090	Priority Register A for Slave 2	MATRIX_PRAS2	Write	0x00000000
0x0094	Reserved	-	-	-
0x0098	Priority Register A for Slave 3	MATRIX_PRAS3	Write	0x00000000
0x009C	Reserved	-	-	-
0x00A0	Priority Register A for Slave 4	MATRIX_PRAS4	Write	0x00000000
0x00A4	Reserved	-	-	-
0x00A8	Priority Register A for Slave 5	MATRIX_PRAS5	Write	0x00000000
0x00A8 - 0x00FC	Reserved	-	-	-
0x0100	Master Remap Control Register	MATRIX_MRCR	Read/Write	0x00000000
0x0104 - 0x010C	Reserved	-	-	-
0x0110	Reserved	-	-	-
0x0114	Bus Matrix TCM Configuration Register	MATRIX_TCR	Read/Write	0x00000000
0x0118-0x11C	Reserved	-	-	-
0x0120	EBI Chip Select Assignment Register	EBI_CSA	Read/Write	0x00010000
0x0124 - 0x01FC	Reserved	-	-	-

## 21.5.1 Bus Matrix Master Configuration Registers

**Register Name:** MATRIX\_MCFG0...MATRIX\_MCFG5

**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 four-beat burst allowing re arbitration at each four-beat burst end.

3: Eight-beat Burst

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

4: Sixteen-beat Burst

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

## 21.5.2 Bus Matrix Slave Configuration Registers

**Register Name:** MATRIX\_SCFG0...MATRIX\_SCFG5

**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.

Note that an unreasonably small value breaks every burst and the Bus Matrix then 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 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 current slave access, if no other master request is pending, the slave remains connected to the last master that accessed it.

This results in not having the one cycle latency when the last master tries access to 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 of which has been written in the FIXED\_DEFMSTR field.

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

- **FIXED\_DEFMSTR: Fixed Default Master**

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

- **ARBT: Arbitration Type**

0: Round-Robin Arbitration

1: Fixed Priority Arbitration

2: Reserved

3: Reserved



## 21.5.3 Bus Matrix Priority Registers A For Slaves

**Register Name:** MATRIX\_PRAS0...MATRIX\_PRAS5

**Access Type:** Write

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
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 to the selected slave. The higher the number, the higher the priority.

## 21.5.4 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
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	RCB5	RCB4	RCB3	RCB2	RCB1	RCB0

- **RCBx: Remap Command Bit for AHB Master x**

0: Disable remapped address decoding for the selected Master.

1: Enable remapped address decoding for the selected Master.

**Table 21-2.** AT91SAM9R/RL64 Remap Control Bits

RCBx	Master
RCB0	ARM926 Instruction
RCB1	ARM926 Data
RCB2	Peripheral DMA Controller
RCB3	USB Device High Speed DMA
RCB4	LCD Controller DMA
RCB5	DMA Controller

## 21.5.5 Bus Matrix TCM Configuration Register

**Register Name:** MATRIX\_TCR

**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
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
DTCM_SIZE				ITCM_SIZE			

- **ITCM\_SIZE: Size of ITCM enabled memory block**

0000: 0 KB (No ITCM Memory)

0101: 16 KB

0110: 32 KB

Others: Reserved

- **DTCM\_SIZE: Size of DTCM enabled memory block**

0000: 0 KB (No DTCM Memory)

0101: 16 KB

0110: 32 KB

Others: Reserved

## 21.5.6 EBI0 Chip Select Assignment Register

**Register Name:** EBI\_CSA

**Access Type:** Read/Write

**Reset:** 0x0001\_0000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	VDDIOMSEL
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: EBI0 Chip Select 1 Assignment**

0 = EBI0 Chip Select 1 is assigned to the Static Memory Controller.

1 = EBI0 Chip Select 1 is assigned to the SDRAM Controller.

- **EBI\_CS3A: EBI0 Chip Select 3 Assignment**

0 = EBI0 Chip Select 3 is only assigned to the Static Memory Controller and EBI0\_NCS3 behaves as defined by the SMC.

1 = EBI0 Chip Select 3 is assigned to the Static Memory Controller and the NAND Flash Logic is activated.

- **EBI\_CS4A: EBI0 Chip Select 4 Assignment**

0 = EBI0 Chip Select 4 is only assigned to the Static Memory Controller and EBI0\_NCS4 behaves as defined by the SMC.

1 = EBI0 Chip Select 4 is assigned to the Static Memory Controller and the CompactFlash Logic (first slot) is activated.

- **EBI\_CS5A: EBI0 Chip Select 5 Assignment**

0 = EBI0 Chip Select 5 is only assigned to the Static Memory Controller and EBI0\_NCS5 behaves as defined by the SMC.

1 = EBI0 Chip Select 5 is assigned to the Static Memory Controller and the CompactFlash Logic (second slot) is activated.

- **EBI\_DBPUC: EBI0 Data Bus Pull-Up Configuration**

0 = EBI0 D0 - D15 Data Bus bits are internally pulled-up to the VDDIOM0 power supply.

1 = EBI0 D0 - D15 Data Bus bits are not internally pulled-up.

- **VDDIOMSEL: Memory voltage selection**

0 = Memories are 1.8V powered.

1 = Memories are 3.3V powered.





## 22. External Bus Interface (EBI)

### 22.1 Description

The External Bus Interface (EBI) is designed to ensure the successful data transfer between several external devices and the embedded Memory Controller of an ARM-based device. The Static Memory, SDRAM and ECC 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.

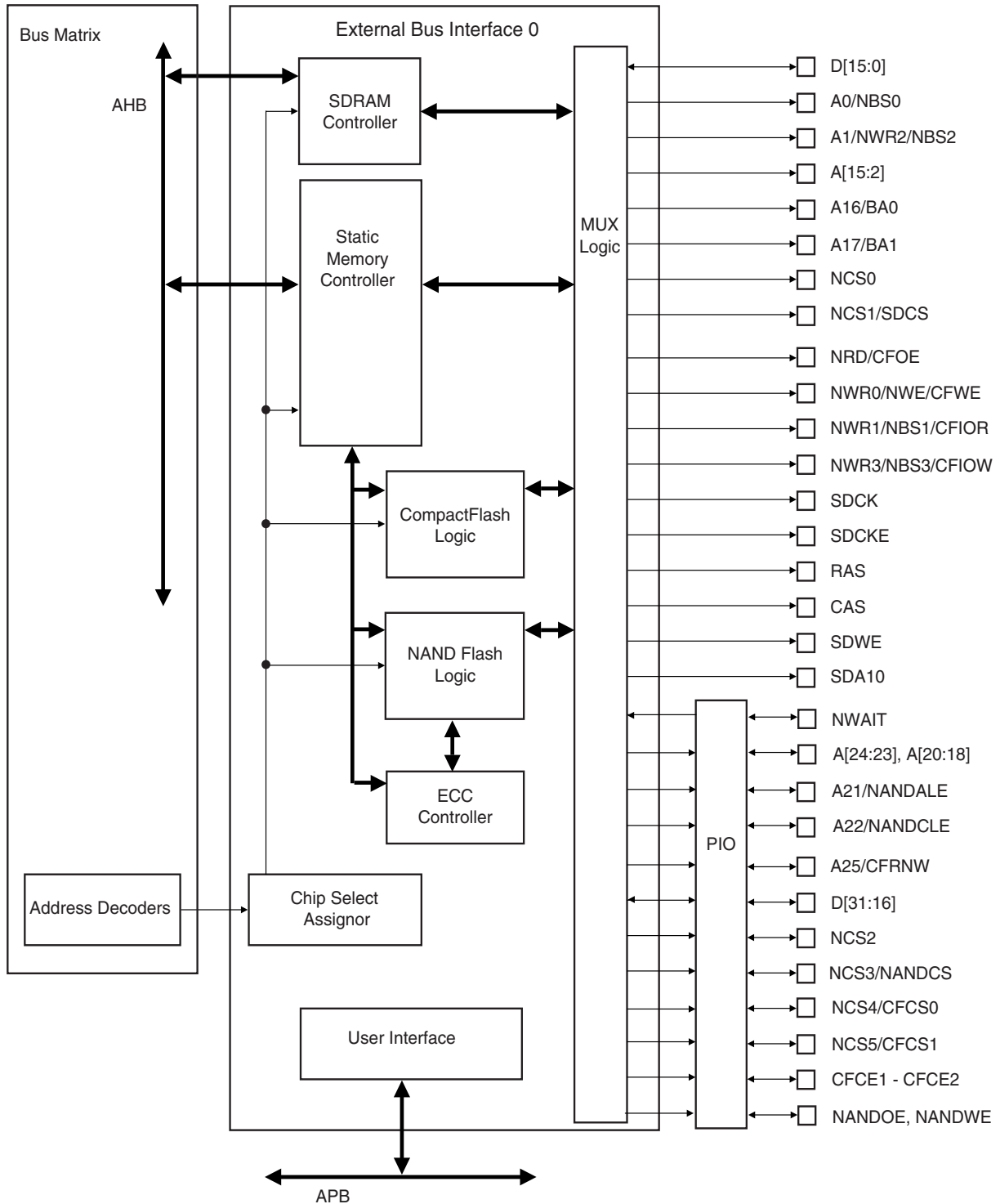
The EBI also supports the CompactFlash and the NAND Flash protocols via integrated circuitry that greatly reduces the requirements for external components. Furthermore, the EBI handles data transfers with up to six external devices, each assigned to six address spaces defined by the embedded Memory Controller. Data transfers are performed through a 16-bit or 32-bit data bus, an address bus of up to 26 bits, up to six chip select lines (NCS[5:0]) and several control pins that are generally multiplexed between the different external Memory Controllers.

## 22.2 Block Diagram

### 22.2.1 External Bus Interface 0

Figure 22-1 shows the organization of the External Bus Interface 0.

Figure 22-1. Organization of the External Bus Interface 0



## 22.3 I/O Lines Description

Table 22-1. EBI I/O Lines Description

Name	Function	Type	Active Level
<b>EBI</b>			
EBI_D0 - EBI_D31	Data Bus	I/O	
EBI_A0 - EBI_A25	Address Bus	Output	
EBI_NWAIT	External Wait Signal	Input	Low
<b>SMC</b>			
EBI_NCS0 - EBI_NCS5	Chip Select Lines	Output	Low
EBI_NWR0 - EBI_NWR3	Write Signals	Output	Low
EBI_NRD	Read Signal	Output	Low
EBI_NWE	Write Enable	Output	Low
EBI_NBS0 - EBI_NBS3	Byte Mask Signals	Output	Low
<b>EBI for CompactFlash Support</b>			
EBI_CFCE1 - EBI_CFCE2	CompactFlash Chip Enable	Output	Low
EBI_CFOE	CompactFlash Output Enable	Output	Low
EBI_CFWE	CompactFlash Write Enable	Output	Low
EBI_CFIOR	CompactFlash I/O Read Signal	Output	Low
EBI_CFIOW	CompactFlash I/O Write Signal	Output	Low
EBI_CFRNW	CompactFlash Read Not Write Signal	Output	
EBI_CFCS0 - EBI_CFCS1	CompactFlash Chip Select Lines	Output	Low
<b>EBI for NAND Flash Support</b>			
EBI_NANDCS	NAND Flash Chip Select Line	Output	Low
EBI_NANDOE	NAND Flash Output Enable	Output	Low
EBI_NANDWE	NAND Flash Write Enable	Output	Low
<b>SDRAM Controller</b>			
EBI_SDCK	SDRAM Clock	Output	
EBI_SDCKE	SDRAM Clock Enable	Output	High
EBI_SDACS	SDRAM Controller Chip Select Line	Output	Low
EBI_BA0 - EBI_BA1	Bank Select	Output	
EBI_SDWE	SDRAM Write Enable	Output	Low
EBI_RAS - EBI_CAS	Row and Column Signal	Output	Low
EBI_NWR0 - EBI_NWR3	Write Signals	Output	Low
EBI_NBS0 - EBI_NBS3	Byte Mask Signals	Output	Low
EBI_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 22-2 on page 156](#) details the connections between the two Memory Controllers and the EBI pins.

**Table 22-2.** EBI Pins and Memory Controllers I/O Lines Connections

EBIx Pins <sup>(1)</sup>	SDRAMC I/O Lines	SMC I/O Lines
EBI_NWR1/NBS1/CFIOR	NBS1	NWR1/NUB
EBI_A0/NBS0	Not Supported	SMC_A0/NLB
EBI_A1/NBS2/NWR2	Not Supported	SMC_A1
EBI_A[11:2]	SDRAMC_A[9:0]	SMC_A[11:2]
EBI_SDA10	SDRAMC_A10	Not Supported
EBI_A12	Not Supported	SMC_A12
EBI_A[14:13]	SDRAMC_A[12:11]	SMC_A[14:13]
EBI_A[22:15]	Not Supported	SMC_A[22:15]
EBI_A[25:23]	Not Supported	SMC_A[25:23]
EBI_D[31:0]	D[31:0]	D[31:0]

## 22.4 Application Example

### 22.4.1 Hardware Interface

Table 22-3 on page 157 details the connections to be applied between the EBI pins and the external devices for each Memory Controller.

**Table 22-3.** EBI Pins and External Static Devices Connections

Signals: EBI_	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					
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 <sup>(3)</sup>	BE0 <sup>(5)</sup>
A1/NWR2/NBS2	A1	A0	A0	WE <sup>(2)</sup>	NLB <sup>(4)</sup>	BE2 <sup>(5)</sup>
A2 - A22	A[2:22]	A[1:21]	A[1:21]	A[0:20]	A[0:20]	A[0:20]
A23 - A25	A[23:25]	A[22:24]	A[22:24]	A[21:23]	A[21:23]	A[21:23]
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
NCS5/CFCS1	CS	CS	CS	CS	CS	CS
NRD/CFOE	OE	OE	OE	OE	OE	OE
NWR0/NWE	WE	WE <sup>(1)</sup>	WE	WE <sup>(2)</sup>	WE	WE
NWR1/NBS1	–	WE <sup>(1)</sup>	NUB	WE <sup>(2)</sup>	NUB <sup>(3)</sup>	BE1 <sup>(5)</sup>
NWR3/NBS3	–	–	–	WE <sup>(2)</sup>	NUB <sup>(4)</sup>	BE3 <sup>(5)</sup>

- 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 22-4.** EBI Pins and External Devices Connections

Signals: EBI_	Pins of the Interfaced Device			
	SDRAM	CompactFlash (EBI only)	CompactFlash True IDE Mode (EBI only)	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/NANDALE	–	–	–	ALE
A22/NANDCLE	–	REG	REG	CLE
A23 - A24	–	–	–	–
A25	–	CFRNW <sup>(1)</sup>	CFRNW <sup>(1)</sup>	–
NCS0	–	–	–	–
NCS1/SDCS	CS	–	–	–
NCS2	–	–	–	–
NCS3/NANDCS	–	–	–	–
NCS4/CFCS0	–	CFCS0 <sup>(1)</sup>	CFCS0 <sup>(1)</sup>	–
NCS5/CFCS1	–	CFCS1 <sup>(1)</sup>	CFCS1 <sup>(1)</sup>	–
NANDOE	–	–	–	OE
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	–	–	–

**Table 22-4.** EBI Pins and External Devices Connections (Continued)

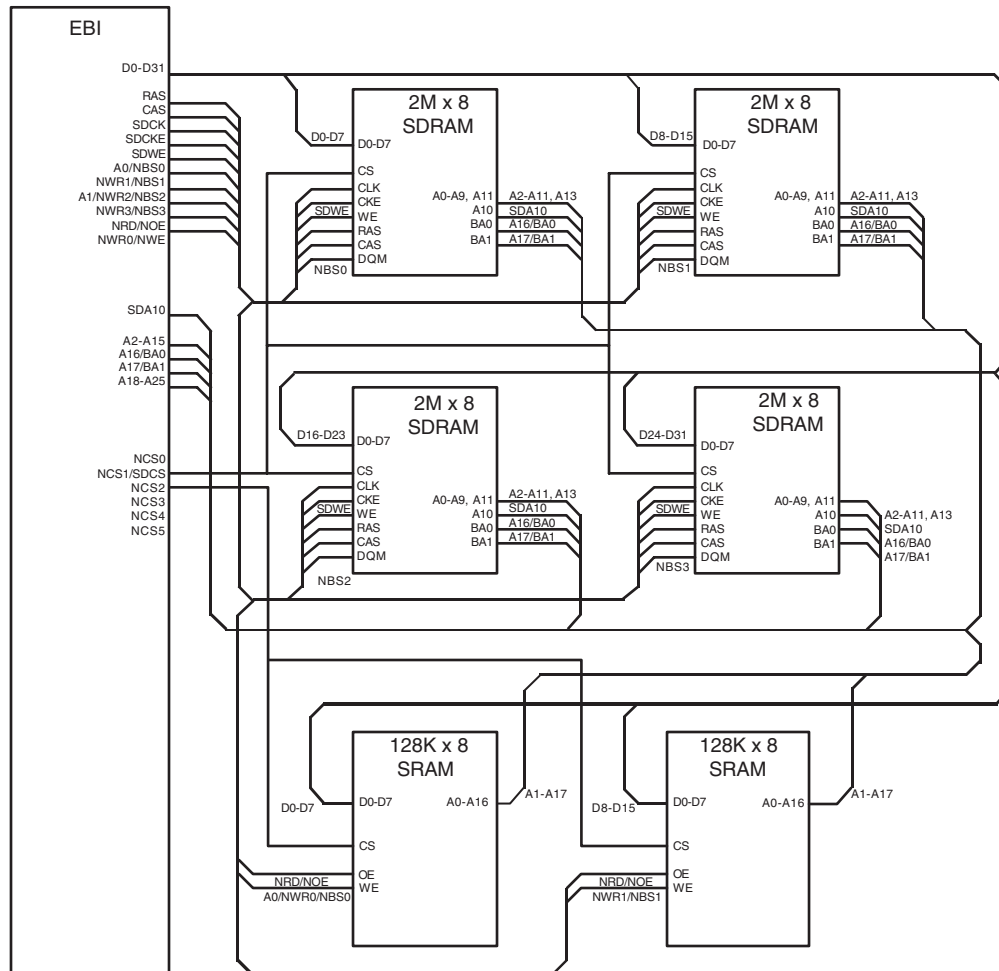
Signals: EBI_	Pins of the Interfaced Device			
	SDRAM	CompactFlash (EBI only)	CompactFlash True IDE Mode (EBI only)	NAND Flash
Controller	SDRAMC	SMC		
SDCKE	CKE	–	–	–
RAS	RAS	–	–	–
CAS	CAS	–	–	–
SDWE	WE	–	–	–
NWAIT	–	WAIT	WAIT	–
Pxx <sup>(2)</sup>	–	CD1 or CD2	CD1 or CD2	–
Pxx <sup>(2)</sup>	–	–	–	CE
Pxx <sup>(2)</sup>	–	–	–	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.

## 22.4.2 Connection Examples

Figure 22-2 shows an example of connections between the EBI and external devices.

Figure 22-2. EBI Connections to Memory Devices



## 22.5 Product Dependencies

### 22.5.1 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.

## 22.6 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 buses and is composed of the following elements:

- the Static Memory Controller (SMC)
- the SDRAM Controller (SDRAMC)



- the ECC Controller (ECC)
- 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

## 22.6.1 Bus Multiplexing

The EBI and EBI1 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.

## 22.6.2 Pull-up Control

The EBI\_CSA register in the Chip Configuration User Interface permit 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 EBI\_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.

## 22.6.3 Static Memory Controller

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

## 22.6.4 SDRAM Controller

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

## 22.6.5 ECC Controller

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

## 22.6.6 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 EBI\_CS4A and/or EBI\_CS5A bit of the EBI\_CSA Register in the Chip Configuration User Interface to the appropriate value enables this logic. For details on this register, refer to the in the Bus Matrix 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).

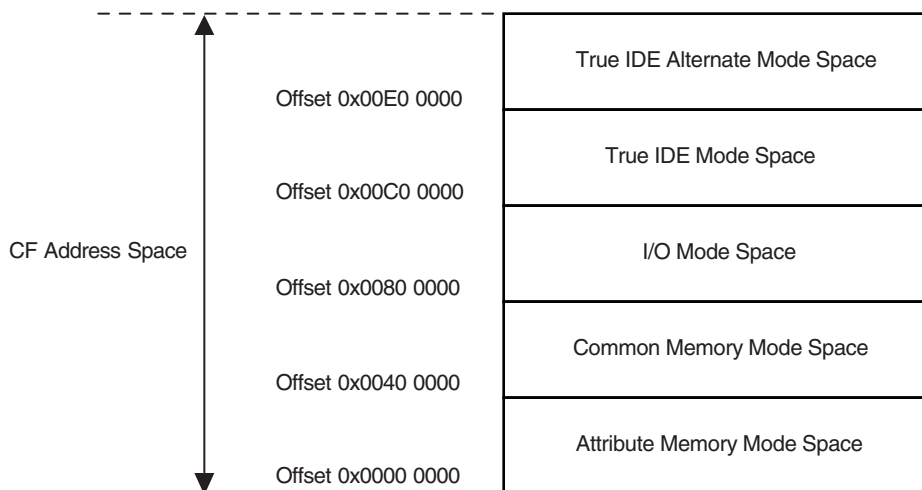
All CompactFlash modes (Attribute Memory, Common Memory, I/O and True IDE) are supported but the signals \_IOIS16 (I/O and True IDE modes) and \_ATA SEL (True IDE mode) are not handled.

## 22.6.6.1 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 22-3](#). A[23:21] bits of the transfer address are used to select the desired mode as described in [Table 22-5](#) on page 162.

**Figure 22-3.** CompactFlash Memory Mapping



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

**Table 22-5.** 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

## 22.6.6.2 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 or NCS5). The Chip Select Register (DBW field in the corresponding Chip Select Register) of the NCS4 and/or NCS5 address space must be set as shown in [Table 22-6](#) 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 22-6.** CFCE1 and CFCE2 Truth Table

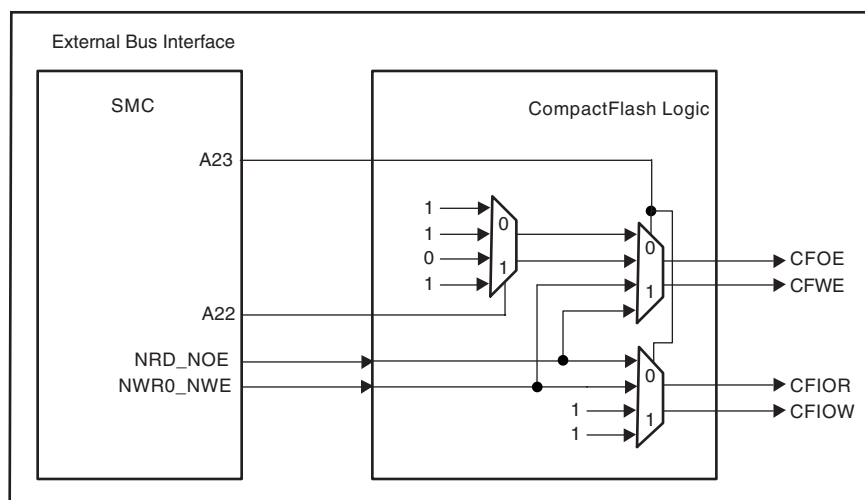
Mode	CFCE2	CFCE1	DBW	Comment	SMC Access Mode
Attribute Memory	NBS1	NBS0	16 bits	Access to Even Byte on D[7:0]	Byte Select
Common Memory	NBS1	NBS0	16bits	Access to Even Byte on D[7:0] Access to Odd Byte on D[15:8]	Byte Select
	1	0	8 bits	Access to Odd Byte on D[7:0]	
I/O Mode	NBS1	NBS0	16 bits	Access 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	–	–	–

### 22.6.6.3 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 CFLOW 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 CFLOW are deactivated. [Figure 22-4 on page 164](#) 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. For details on these signal waveforms, please refer to the section: Setup and Hold Cycles of the Static Memory Controller section.

**Figure 22-4.** CompactFlash Read/Write Control Signals



**Table 22-7.** 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

#### 22.6.6.4 Multiplexing of CompactFlash Signals on EBI Pins

Table 22-8 on page 164 and Table 22-9 on page 165 illustrate the multiplexing of the CompactFlash logic signals with other EBI signals on the EBI pins. The EBI pins in Table 22-8 are strictly dedicated to the CompactFlash interface as soon as the EBI\_CS4A and/or EBI\_CS5A field of the EBI\_CSA Register in the Chip Configuration User Interface is set. These pins must not be used to drive any other memory devices.

The EBI pins in Table 22-9 on page 165 remain shared between all memory areas when the corresponding CompactFlash interface is enabled (EBI\_CS4A = 1 and/or EBI\_CS5A = 1).

**Table 22-8.** 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

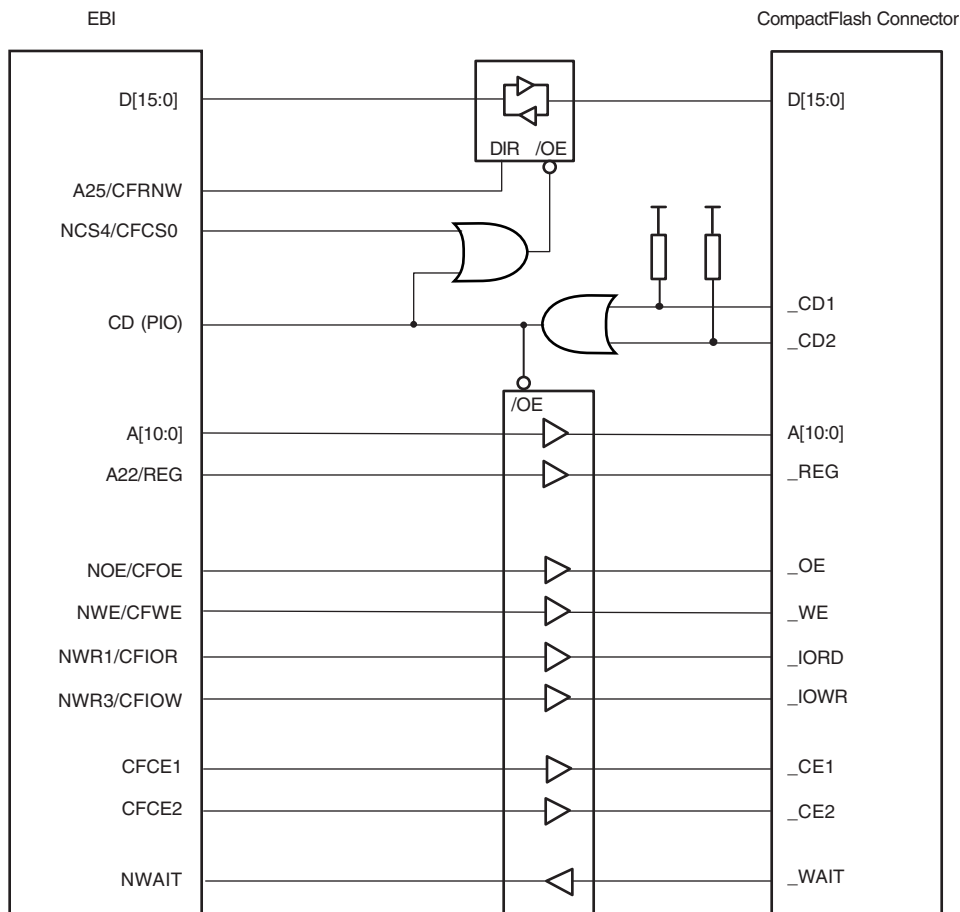
**Table 22-9.** Shared CompactFlash Interface Multiplexing

Pins	Access to CompactFlash Device	Access to Other EBI Devices
	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

### 22.6.6.5 Application Example

Figure 22-5 on page 166 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 22-5. CompactFlash Application Example



## 22.6.7 NAND Flash Support

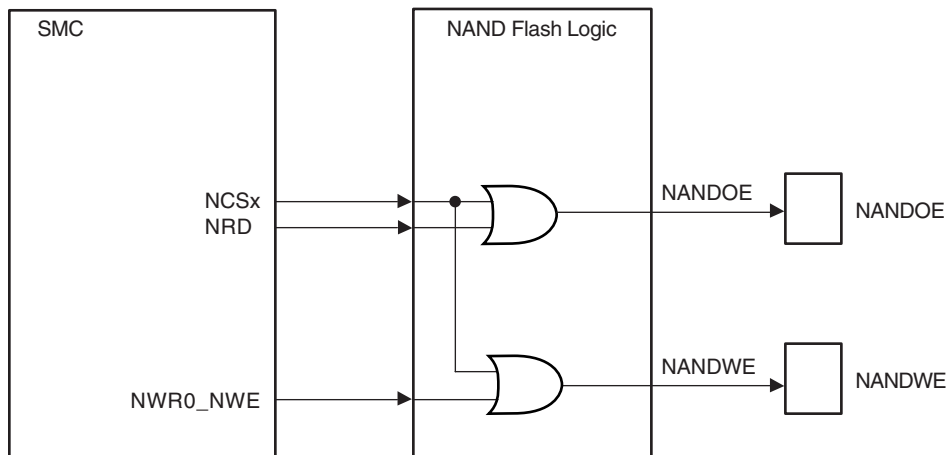
External Bus Interface integrates circuitry that interfaces to NAND Flash devices.

### 22.6.7.1 External Bus Interface

The NAND Flash logic is driven by the Static Memory Controller on the NCS3 address space. Programming the EBI\_CS3A field in the EBI\_CSA Register in the Chip Configuration User Interface to the appropriate value enables the NAND Flash logic. For details on this register, refer to the Bus Matrix Section. Access to an external NAND Flash device is then made by accessing the address space reserved to NCS3 (i.e., between 0x4000 0000 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. See Figure “NAND Flash Signal Multiplexing on EBI Pins” on page 167 for more information. For details on these waveforms, refer to the Static Memory Controller section.

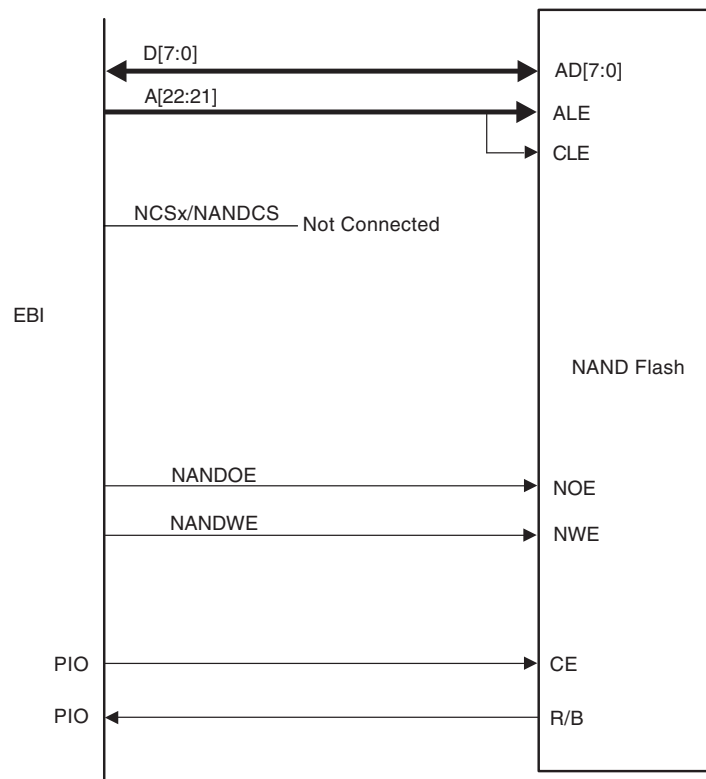
**Figure 22-6.** NAND Flash Signal Multiplexing on EBI Pins



### 22.6.7.2 NAND Flash Signals

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 command, address or data words on the data bus of the NAND Flash device are distinguished by using their address within the NCSx 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 NCSx is not selected, preventing the device from returning to standby mode.

**Figure 22-7.** NAND Flash Application Example



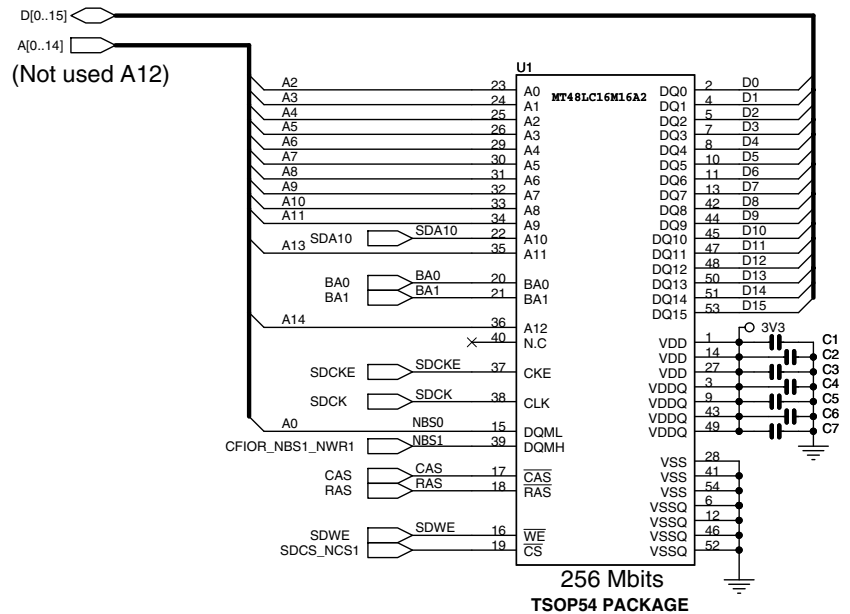
Note: The External Bus Interfaces 0 and 1 are also able to support 16-bit devices.

## 22.7 Implementation Examples

The following hardware configurations are given for illustration only. The user should refer to the memory manufacturer web site to check current device availability.

### 22.7.1 16-bit SDRAM

#### 22.7.1.1 Hardware Configuration



#### 22.7.1.2 Software Configuration

The following configuration has to be performed:

- Assign the EBI CS1 to the SDRAM controller by setting the bit EBI\_CS1A in the EBI Chip Select Assignment Register located in the bus matrix memory space.
- Initialize the SDRAM Controller depending on the SDRAM device and system bus frequency.

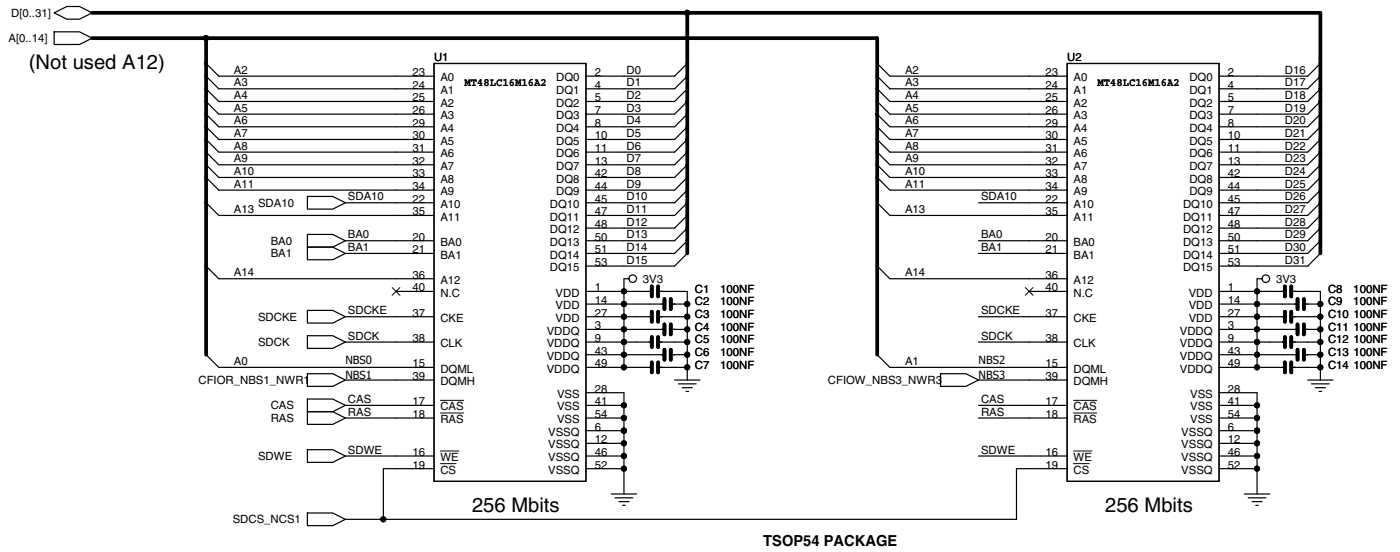
The Data Bus Width is to be programmed to 16 bits.

The SDRAM initialization sequence is described in the section “SDRAM Device Initialization” in “SDRAM Controller (SDRAMC)”.



## 22.7.2 32-bit SDRAM

### 22.7.2.1 Hardware Configuration



### 22.7.2.2 Software Configuration

The following configuration has to be performed:

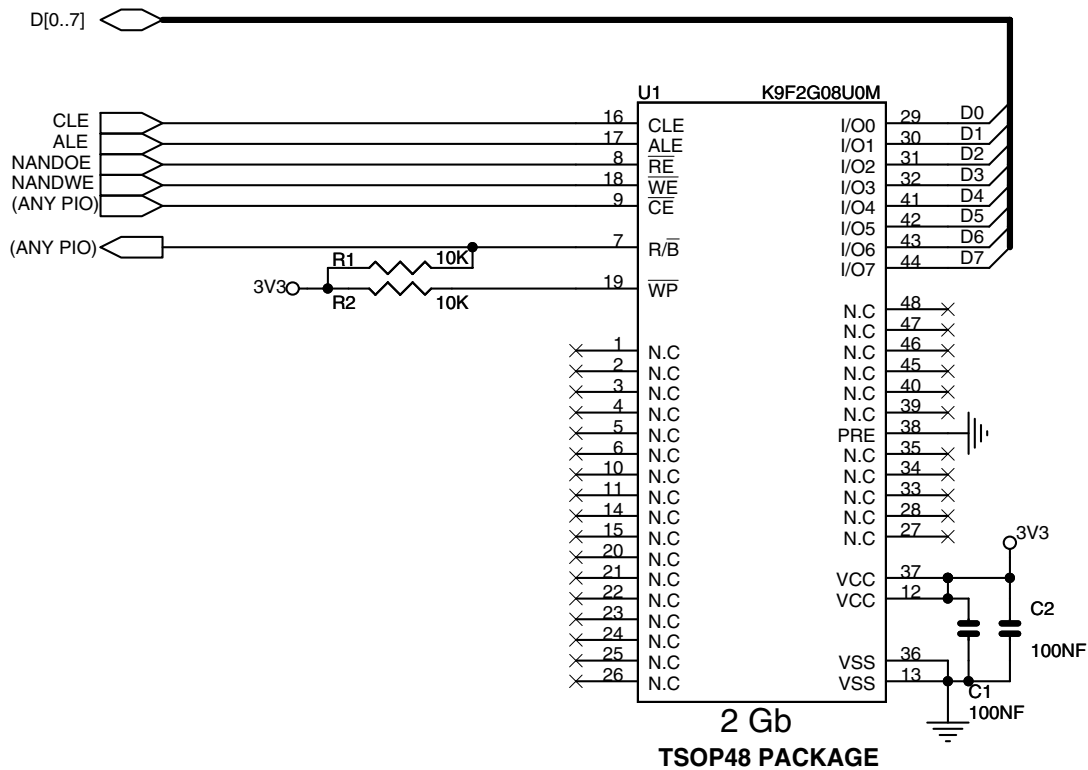
- Assign the EBI CS1 to the SDRAM controller by setting the bit EBI\_CS1A in the EBI Chip Select Assignment Register located in the bus matrix memory space.
- Initialize the SDRAM Controller depending on the SDRAM device and system bus frequency.

The Data Bus Width is to be programmed to 32 bits. The data lines D[16..31] are multiplexed with PIO lines and thus the dedicated PIOs must be programmed in peripheral mode in the PIO controller.

The SDRAM initialization sequence is described in the section “SDRAM Device Initialization” in “SDRAM Controller (SDRAMC)”.

## 22.7.3 8-bit NAND Flash

### 22.7.3.1 Hardware Configuration



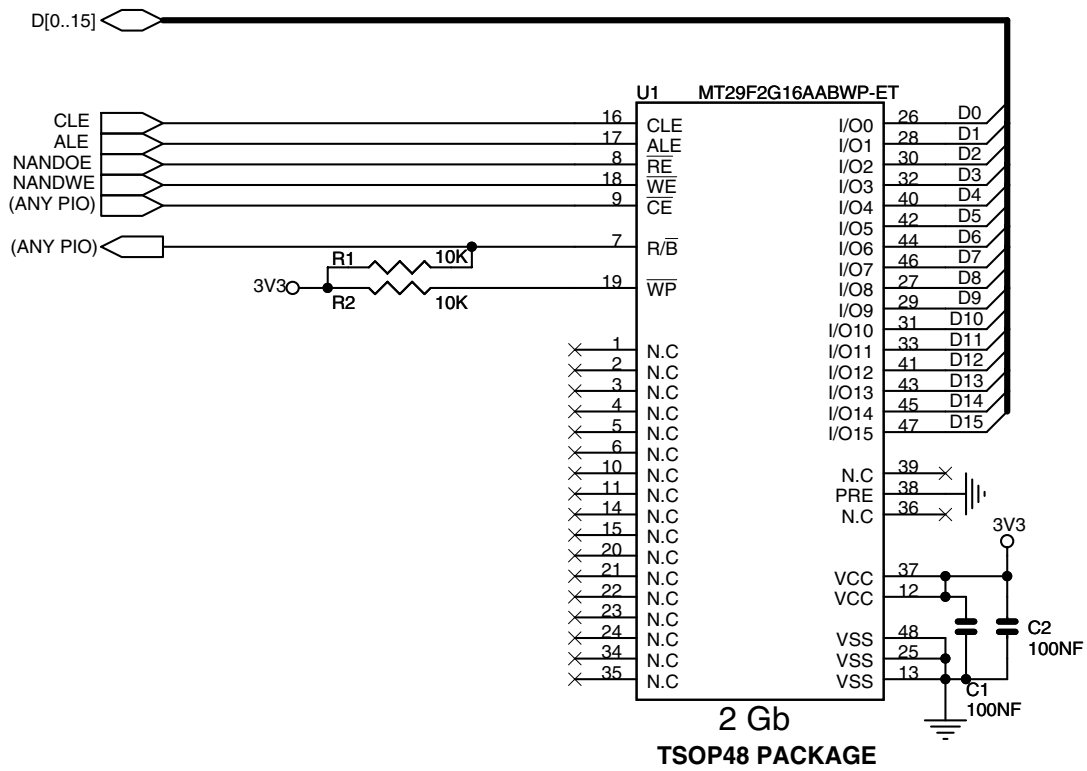
### 22.7.3.2 Software Configuration

The following configuration has to be performed:

- Assign the EBI CS3 to the NAND Flash by setting the bit EBI\_CS3A in the EBI Chip Select Assignment Register located in the bus matrix memory space
- Reserve A21 / A22 for ALE / CLE functions. Address and Command Latches are controlled respectively by setting to 1 the address bit A21 and A22 during accesses.
- NANDOE and NANDWE signals are multiplexed with PIO lines and thus the dedicated PIOs must be programmed in peripheral mode in the PIO controller.
- A21/NANDALE and A22/NANDCLE signals are multiplexed with PIO lines and thus the dedicated PIOs must be programmed in peripheral mode in the PIO controller.
- Configure a PIO line as an input to manage the Ready/Busy signal.
- Configure Static Memory Controller CS3 Setup, Pulse, Cycle and Mode accordingly to NAND Flash timings, the data bus width and the system bus frequency.

## 22.7.4 16-bit NAND Flash

### 22.7.4.1 Hardware Configuration

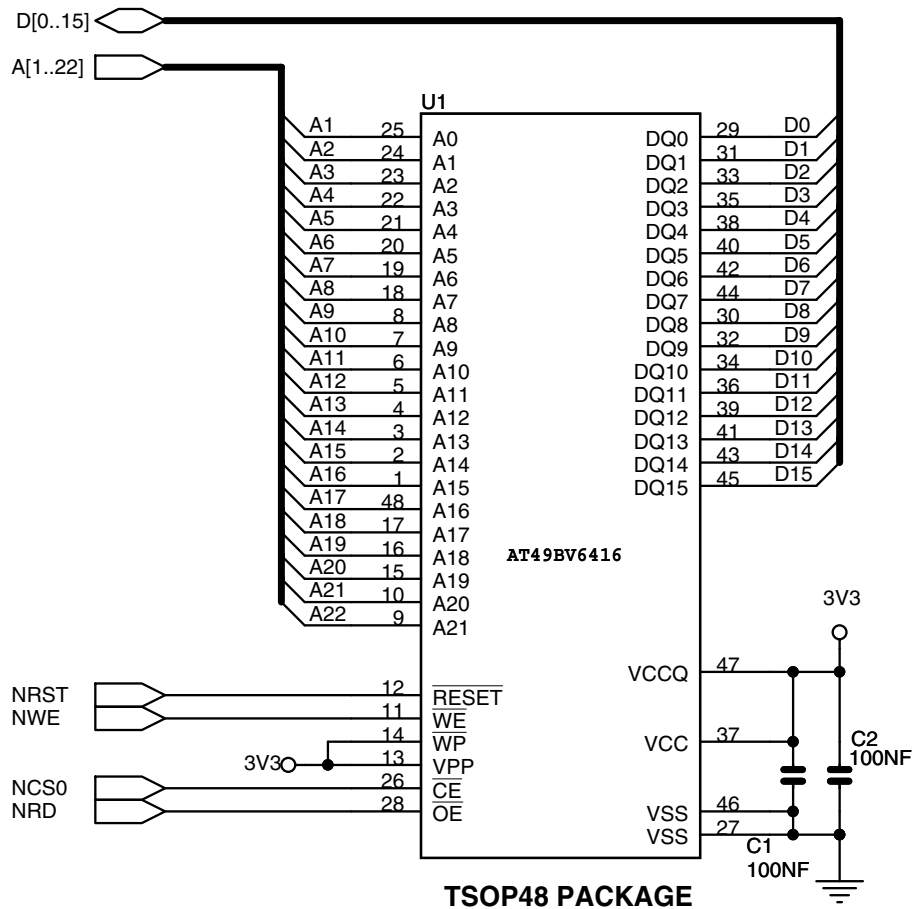


### 22.7.4.2 Software Configuration

The software configuration is the same as for an 8-bit NAND Flash except the data bus width programmed in the mode register of the Static Memory Controller.

## 22.7.5 NOR Flash on NCS0

### 22.7.5.1 Hardware Configuration



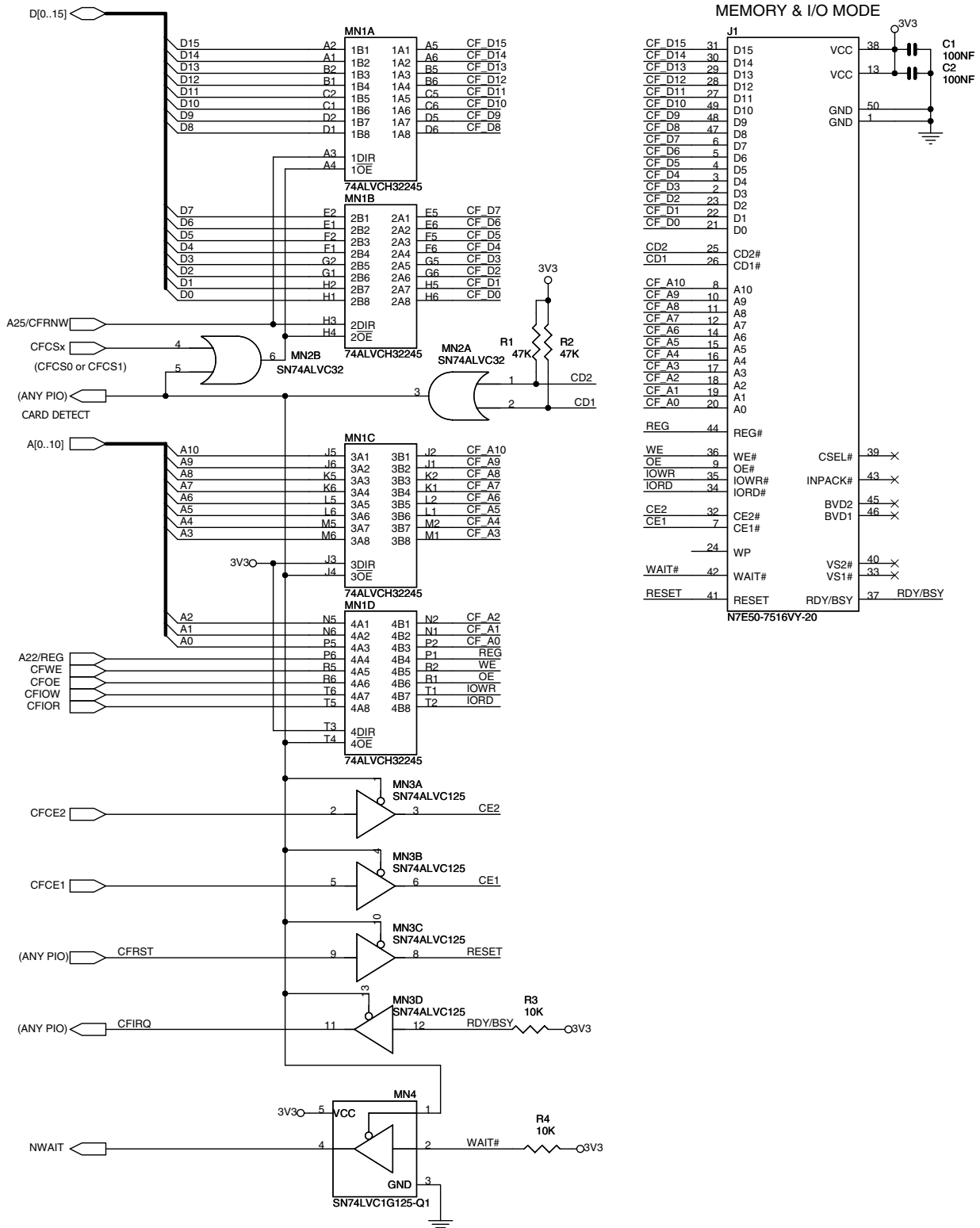
### 22.7.5.2 Software Configuration

The default configuration for the Static Memory Controller, byte select mode, 16-bit data bus, Read/Write controlled by Chip Select, allows boot on 16-bit non-volatile memory at slow clock.

For another configuration, configure the Static Memory Controller CS0 Setup, Pulse, Cycle and Mode depending on Flash timings and system bus frequency.

## 22.7.6 Compact Flash

### 22.7.6.1 Hardware Configuration



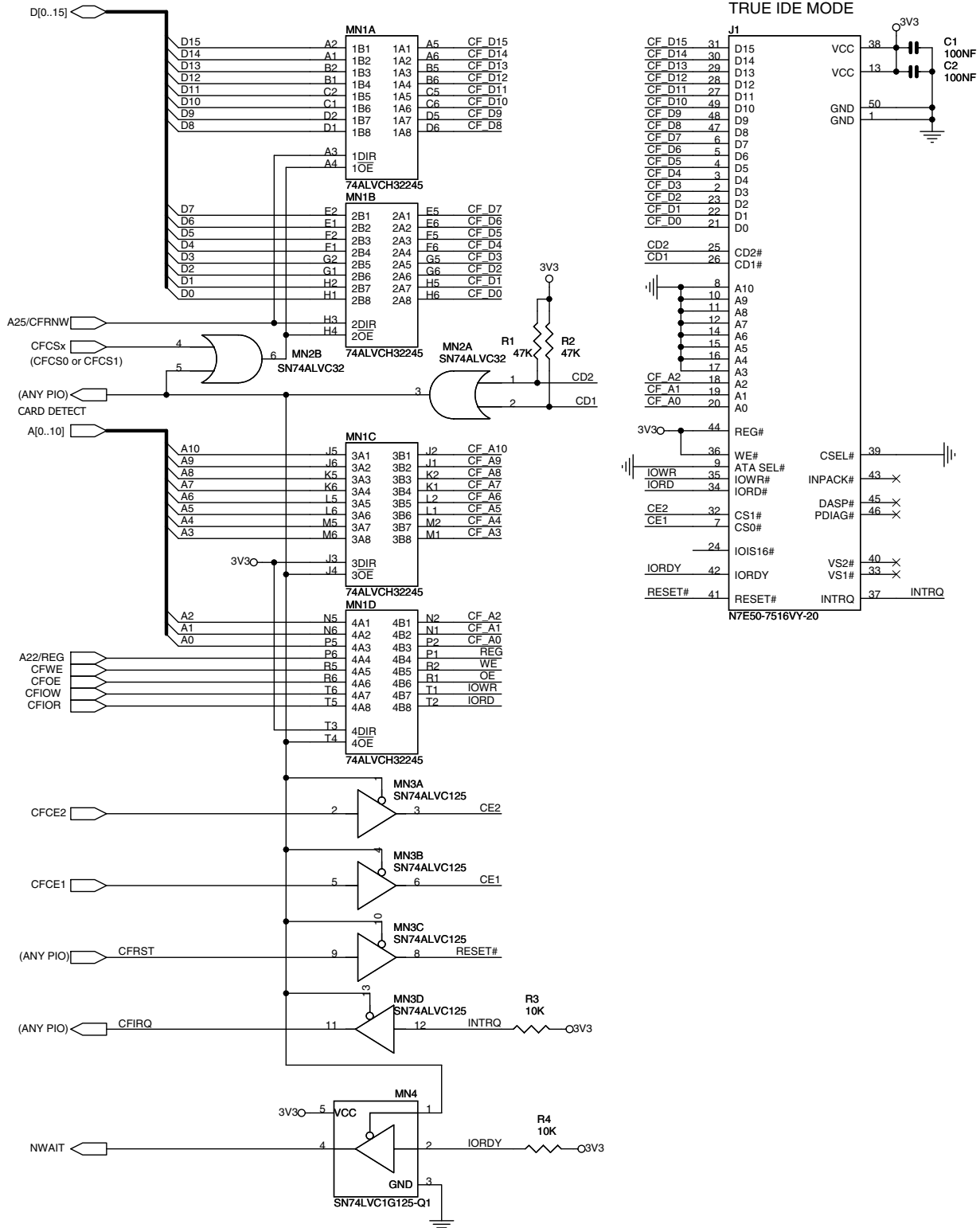
### 22.7.6.2 *Software Configuration*

The following configuration has to be performed:

- Assign the EBI CS4 and/or EBI\_CS5 to the CompactFlash Slot 0 or/and Slot 1 by setting the bit EBI\_CS4A or/and EBI\_CS5A in the EBI Chip Select Assignment Register located in the bus matrix memory space.
- The address line A23 is to select I/O (A23=1) or Memory mode (A23=0) and the address line A22 for REG function.
- A23, CFRNW, CFS0, CFCS1, CFCE1 and CFCE2 signals are multiplexed with PIO lines and thus the dedicated PIOs must be programmed in peripheral mode in the PIO controller.
- Configure a PIO line as an output for CFRST and two others as an input for CFIRQ and CARD DETECT functions respectively.
- Configure SMC CS4 and/or SMC\_CS5 (for Slot 0 or 1) Setup, Pulse, Cycle and Mode accordingly to Compact Flash timings and system bus frequency.

## 22.7.7 Compact Flash True IDE

### 22.7.7.1 Hardware Configuration



### 22.7.7.2 *Software Configuration*

The following configuration has to be performed:

- Assign the EBI CS4 and/or EBI\_CS5 to the CompactFlash Slot 0 or/and Slot 1 by setting the bit EBI\_CS4A or/and EBI\_CS5A in the EBI Chip Select Assignment Register located in the bus matrix memory space.
- The address line A21 is to select Alternate True IDE (A21=1) or True IDE (A21=0) modes.
- CFRNW, CFS0, CFCS1, CFCE1 and CFCE2 signals are multiplexed with PIO lines and thus the dedicated PIOs must be programmed in peripheral mode in the PIO controller.
- Configure a PIO line as an output for CFRST and two others as an input for CFIRQ and CARD DETECT functions respectively.
- Configure SMC CS4 and/or SMC\_CS5 (for Slot 0 or 1) Setup, Pulse, Cycle and Mode accordingly to Compact Flash timings and system bus frequency.



## 23. Static Memory Controller (SMC)

### 23.1 Description

The Static Memory Controller (SMC) generates the signals that control the access to the external memory devices or peripheral devices. It has 6 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.

### 23.2 I/O Lines Description

**Table 23-1.** 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

### 23.3 Multiplexed Signals

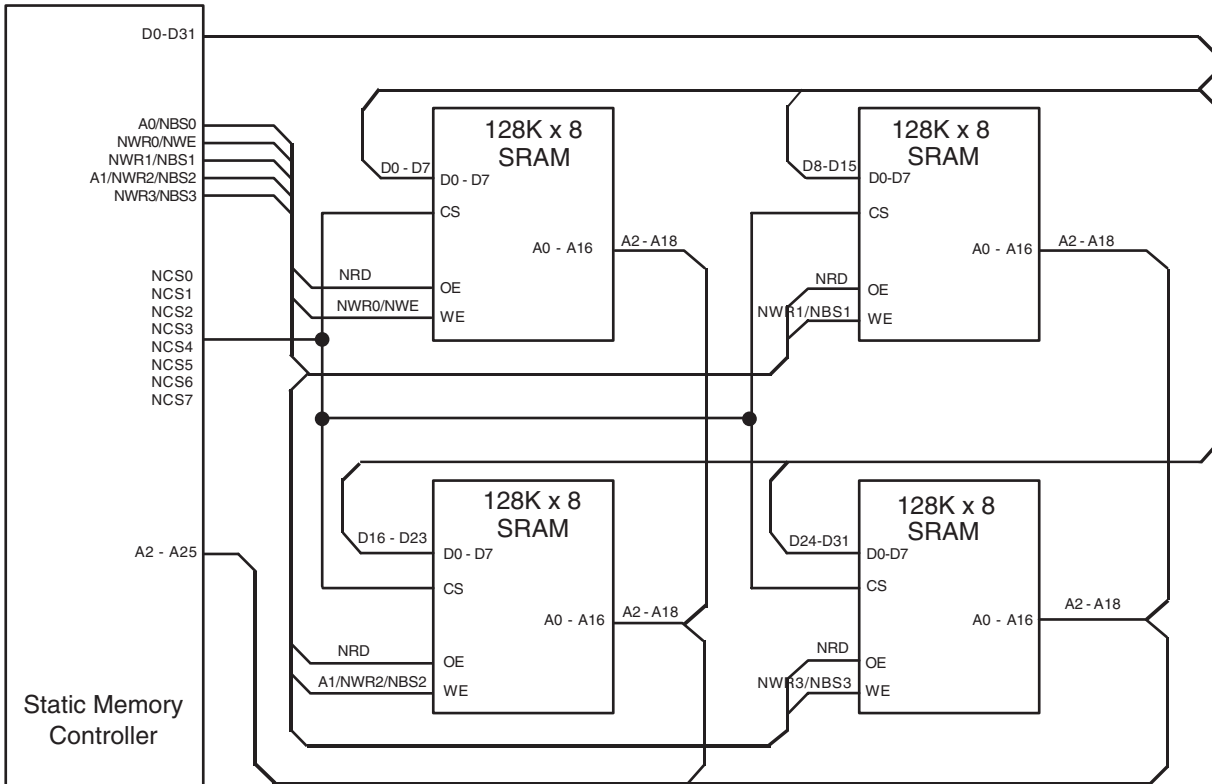
**Table 23-2.** Static Memory Controller (SMC) Multiplexed Signals

Multiplexed Signals			Related Function
NWR0	NWE		Byte-write or byte-select access, see <a href="#">“Byte Write or Byte Select Access” on page 179</a>
A0	NBS0		8-bit or 16-/32-bit data bus, see <a href="#">“Data Bus Width” on page 179</a>
NWR1	NBS1		Byte-write or byte-select access see <a href="#">“Byte Write or Byte Select Access” on page 179</a>
A1	NWR2	NBS2	8-/16-bit or 32-bit data bus, see <a href="#">“Data Bus Width” on page 179</a> . Byte-write or byte-select access, see <a href="#">“Byte Write or Byte Select Access” on page 179</a>
NWR3	NBS3		Byte-write or byte-select access see <a href="#">“Byte Write or Byte Select Access” on page 179</a>

## 23.4 Application Example

### 23.4.1 Hardware Interface

Figure 23-1. SMC Connections to Static Memory Devices



## 23.5 Product Dependencies

### 23.5.1 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 purposes by the PIO Controller.

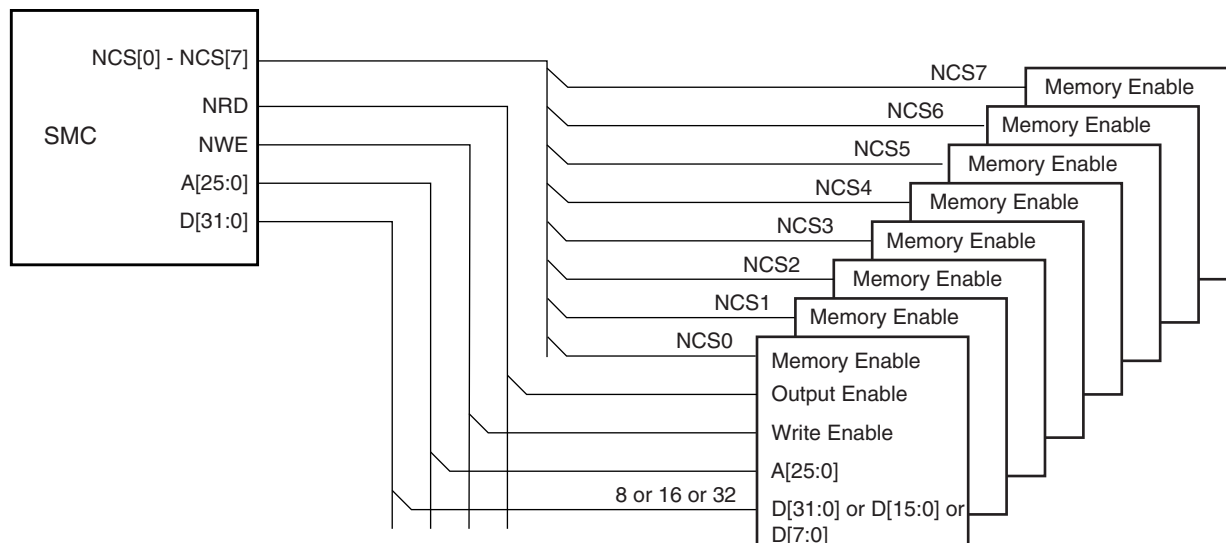
## 23.6 External Memory Mapping

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 23-2](#)).

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 23-2.** Memory Connections for Eight External Devices



## 23.7 Connection to External Devices

### 23.7.1 Data Bus Width

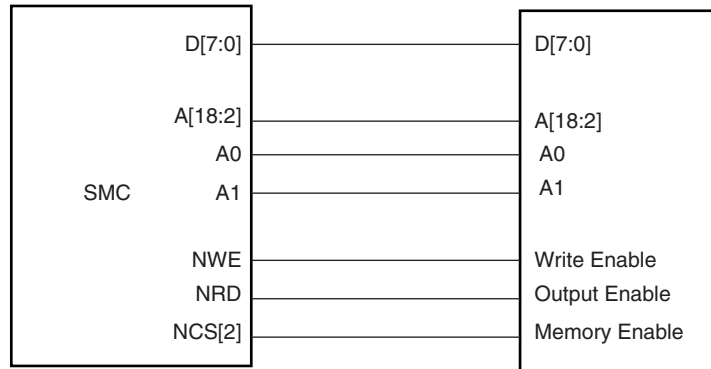
A data bus width of 8, 16, or 32 bits can be selected for each chip select. This option is controlled by the field DBW in SMC\_MODE (Mode Register) for the corresponding chip select.

[Figure 23-3](#) shows how to connect a 512K x 8-bit memory on NCS2. [Figure 23-4](#) shows how to connect a 512K x 16-bit memory on NCS2. [Figure 23-5](#) shows two 16-bit memories connected as a single 32-bit memory

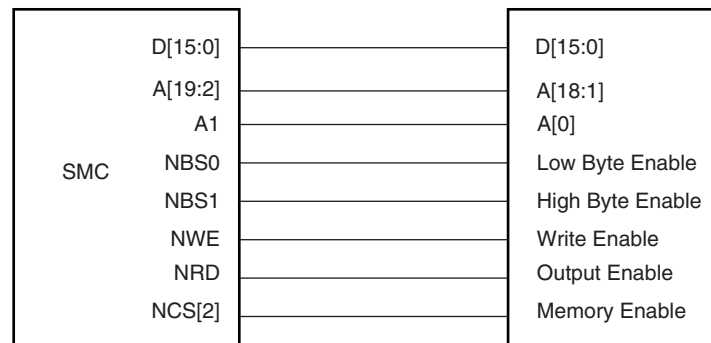
### 23.7.2 Byte Write or Byte Select Access

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.

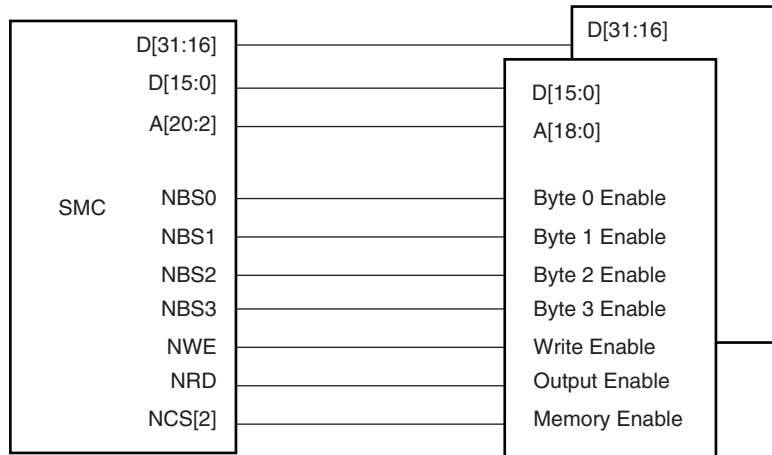
**Figure 23-3.** Memory Connection for an 8-bit Data Bus



**Figure 23-4.** Memory Connection for a 16-bit Data Bus



**Figure 23-5.** Memory Connection for a 32-bit Data Bus



## 23.7.2.1 *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 23-6](#).

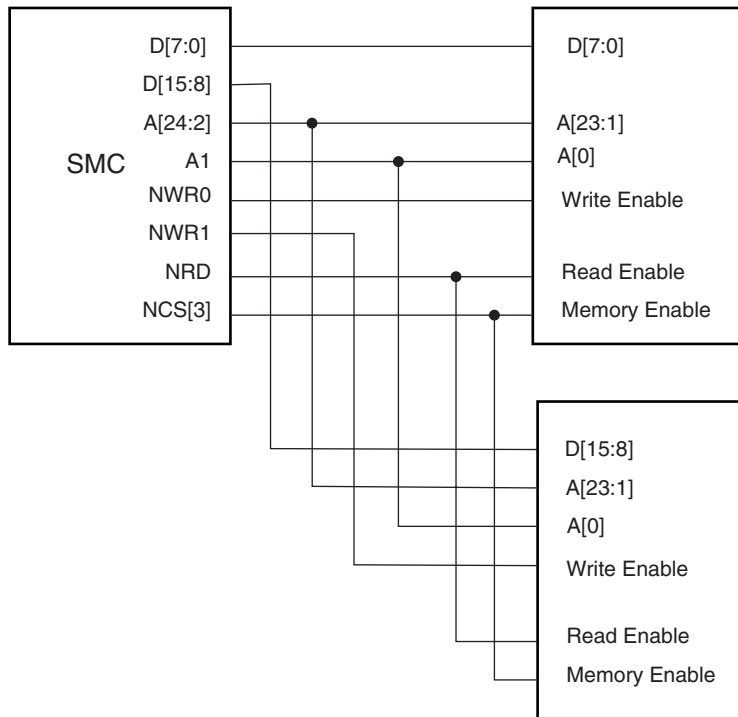
## 23.7.2.2 *Byte Select Access*

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 23-7](#) 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 23-6.** Connection of 2 x 8-bit Devices on a 16-bit Bus: Byte Write Option

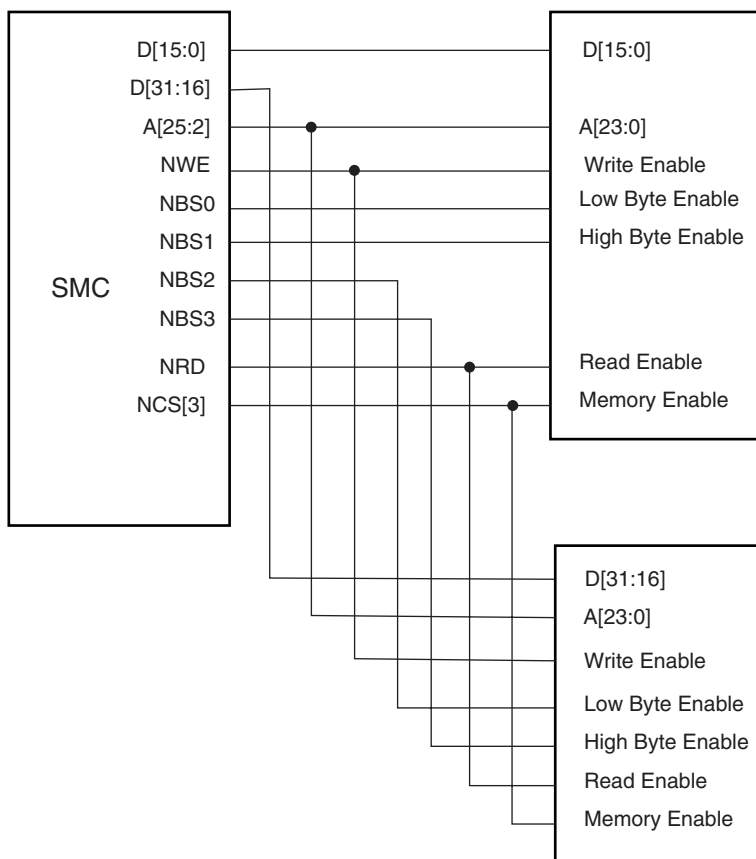


### 23.7.2.3 Signal Multiplexing

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 23-3](#) 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 23-7.** Connection of 2x16-bit Data Bus on a 32-bit Data Bus (Byte Select Option)



**Table 23-3.** SMC Multiplexed Signal Translation

Signal Name	32-bit Bus			16-bit Bus		8-bit Bus
	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			

## 23.8 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..5] chip select lines.

### 23.8.1 Read Waveforms

The read cycle is shown on [Figure 23-8](#).

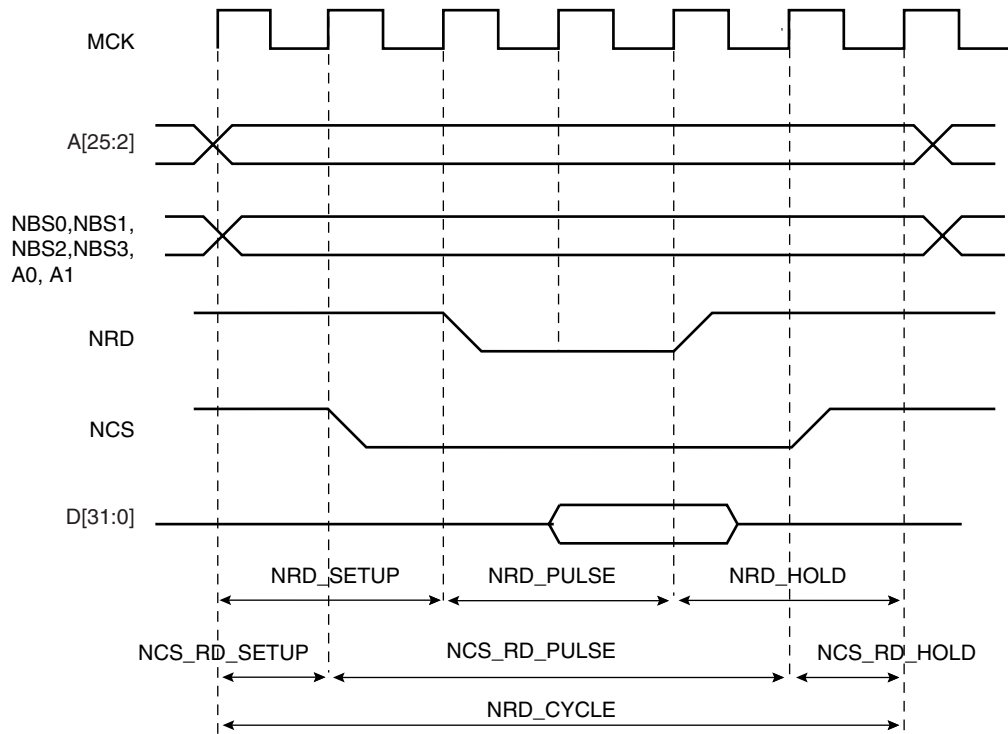
The read cycle starts with the address setting on the memory address bus, i.e.:

{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 23-8.** Standard Read Cycle



#### 23.8.1.1 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.



## 23.8.1.2 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.

## 23.8.1.3 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:

$$\begin{aligned} \text{NRD\_CYCLE} &= \text{NRD\_SETUP} + \text{NRD\_PULSE} + \text{NRD\_HOLD} \\ &= \text{NCS\_RD\_SETUP} + \text{NCS\_RD\_PULSE} + \text{NCS\_RD\_HOLD} \end{aligned}$$

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:

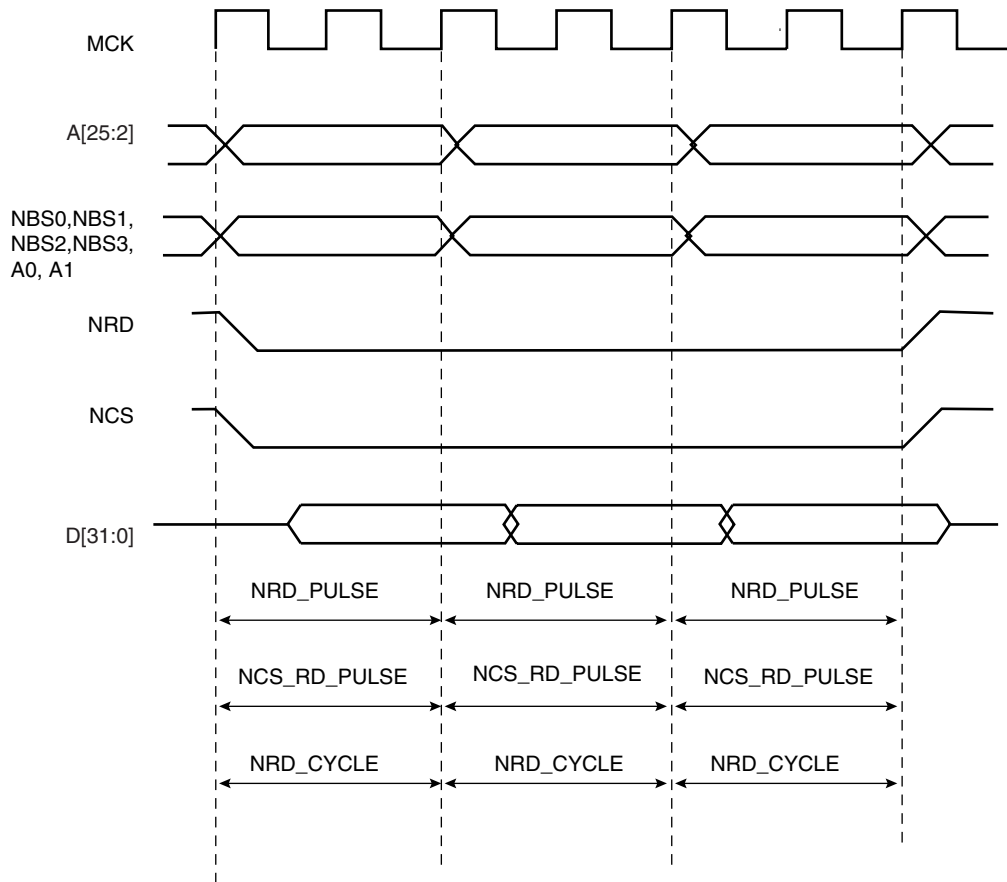
$$\text{NRD\_HOLD} = \text{NRD\_CYCLE} - \text{NRD\_SETUP} - \text{NRD\_PULSE}$$

$$\text{NCS\_RD\_HOLD} = \text{NRD\_CYCLE} - \text{NCS\_RD\_SETUP} - \text{NCS\_RD\_PULSE}$$

## 23.8.1.4 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 23-9](#)).

**Figure 23-9.** No Setup, No Hold On NRD and NCS Read Signals



**23.8.1.5 Null Pulse**

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

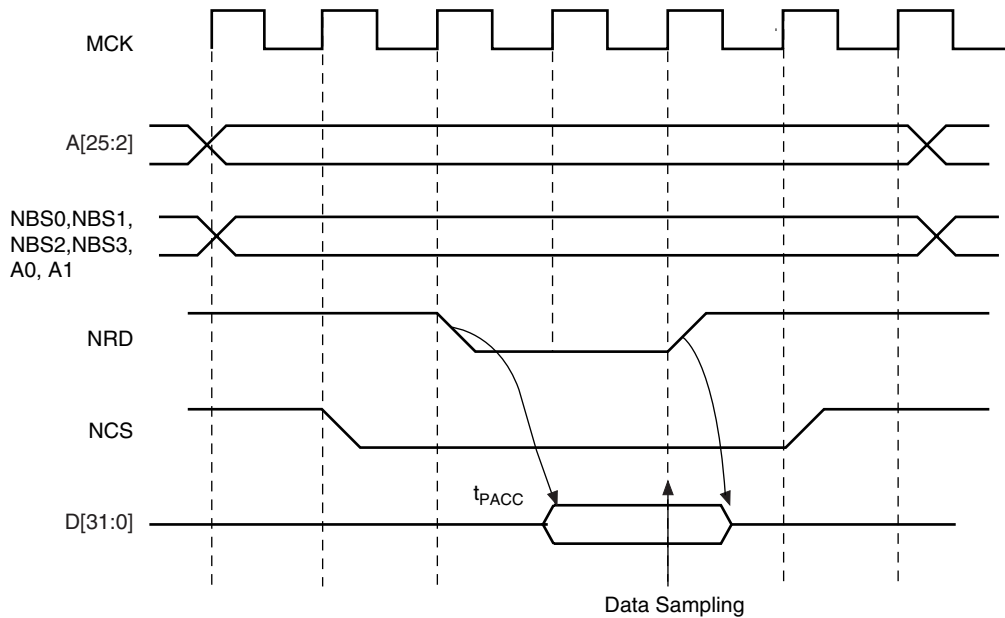
**23.8.2 Read Mode**

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.

**23.8.2.1 Read is Controlled by NRD (READ\_MODE = 1):**

Figure 23-10 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.

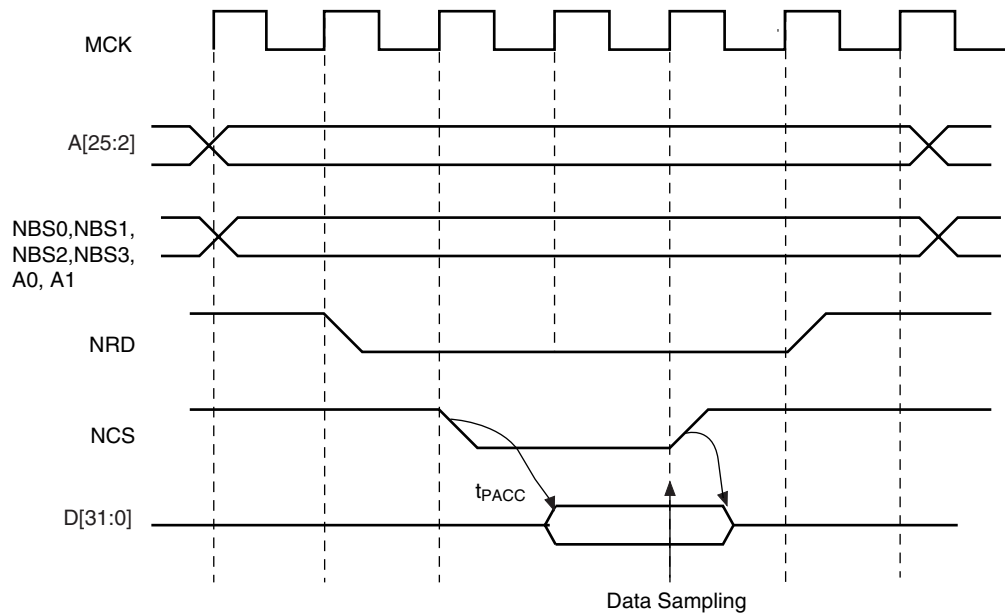
**Figure 23-10.** READ\_MODE = 1: Data is sampled by SMC before the rising edge of NRD



**23.8.2.2** Read is Controlled by NCS (READ\_MODE = 0)

Figure 23-11 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 23-11.** READ\_MODE = 0: Data is sampled by SMC before the rising edge of NCS



### 23.8.3 Write Waveforms

The write protocol is similar to the read protocol. It is depicted in [Figure 23-12](#). The write cycle starts with the address setting on the memory address bus.

#### 23.8.3.1 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.

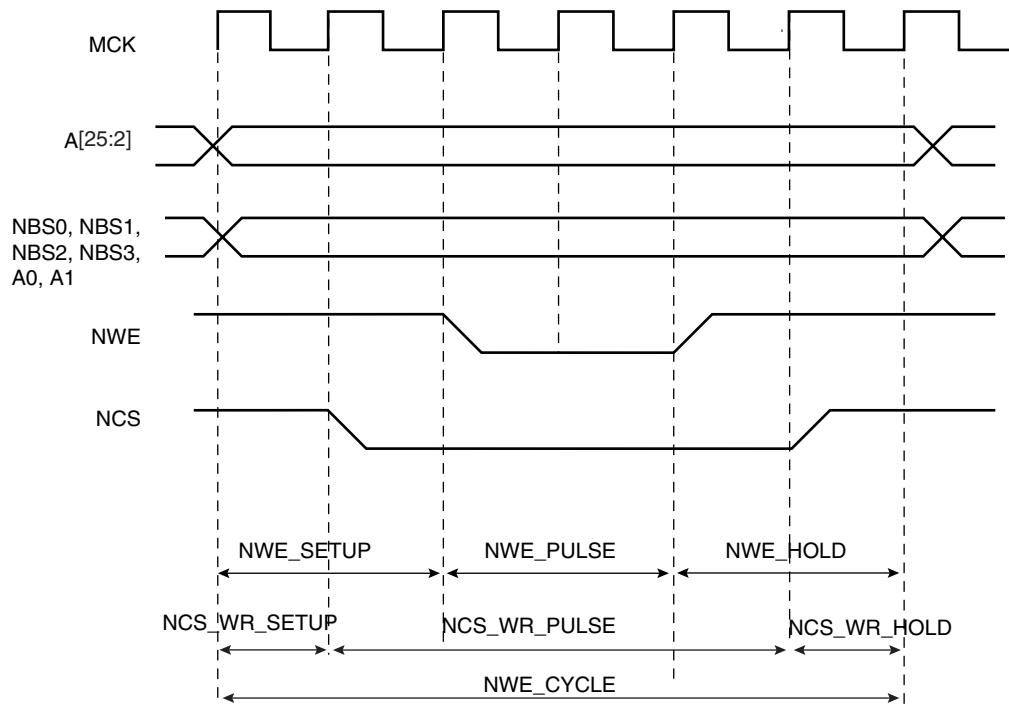
The NWE waveforms apply to all byte-write lines in Byte Write access mode: NWR0 to NWR3.

#### 23.8.3.2 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 23-12.** Write Cycle



### 23.8.3.3 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:

$$\begin{aligned} \text{NWE\_CYCLE} &= \text{NWE\_SETUP} + \text{NWE\_PULSE} + \text{NWE\_HOLD} \\ &= \text{NCS\_WR\_SETUP} + \text{NCS\_WR\_PULSE} + \text{NCS\_WR\_HOLD} \end{aligned}$$

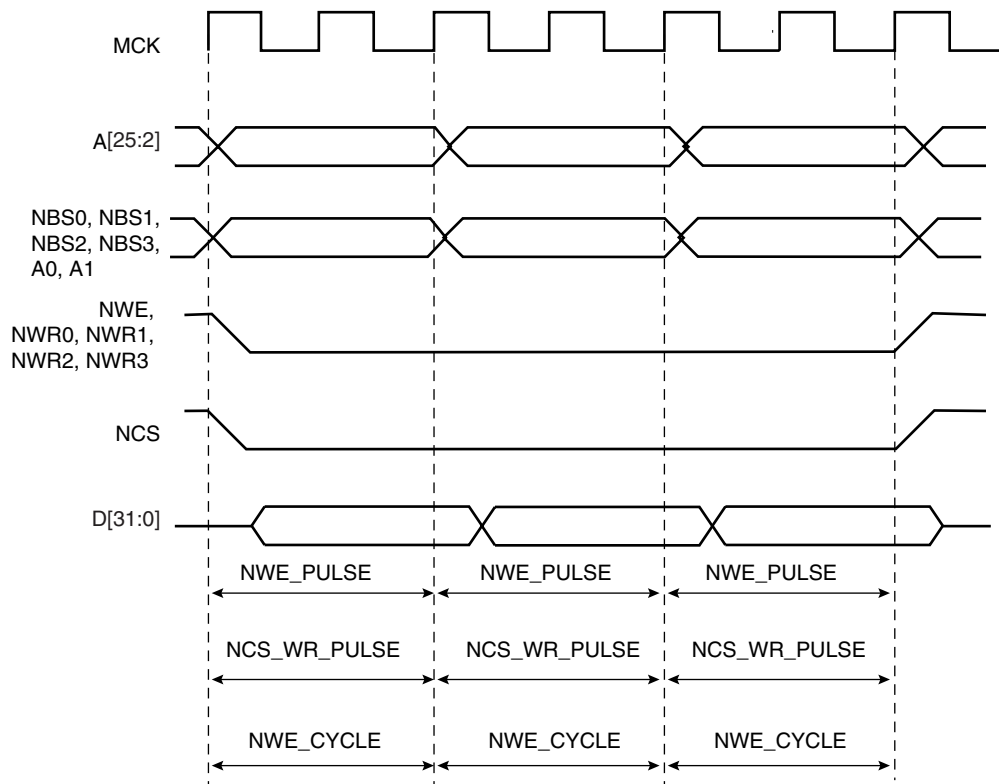
All NWE and NCS (write) timings are defined separately for each chip select as an integer number of Master Clock cycles. To ensure that the NWE and NCS timings are coherent, the user must define the total write cycle instead of the hold timing. This implicitly defines the NWE hold time and NCS (write)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

$$\begin{aligned} \text{NWE\_HOLD} &= \text{NWE\_CYCLE} - \text{NWE\_SETUP} - \text{NWE\_PULSE} \\ \text{NCS\_WR\_HOLD} &= \text{NWE\_CYCLE} - \text{NCS\_WR\_SETUP} - \text{NCS\_WR\_PULSE} \end{aligned}$$

### 23.8.3.4 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 23-13](#)). 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 23-13.** Null Setup and Hold Values of NCS and NWE in Write Cycle



### 23.8.3.5 Null Pulse

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

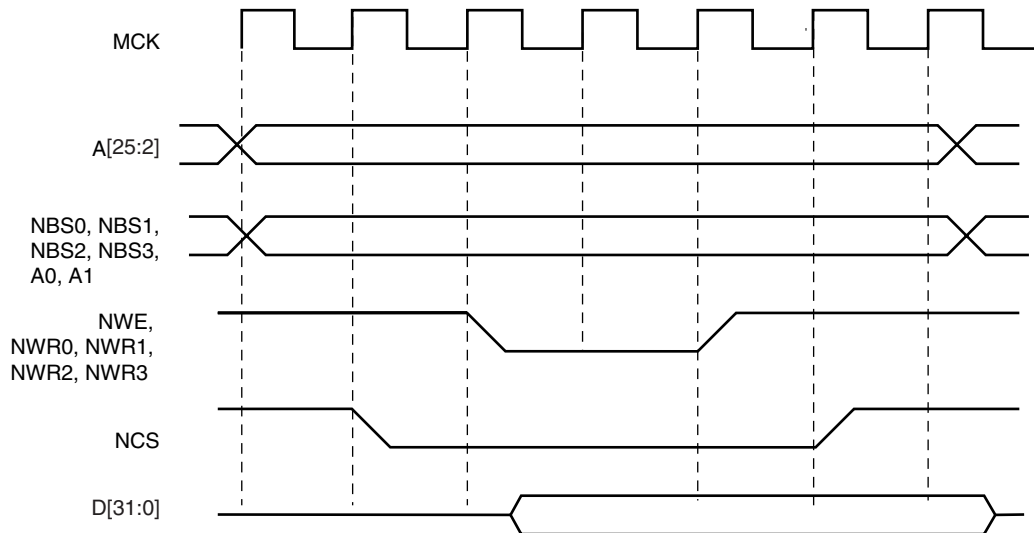
## 23.8.4 Write Mode

The WRITE\_MODE parameter in the SMC\_MODE register of the corresponding chip select indicates which signal controls the write operation.

### 23.8.4.1 Write is Controlled by NWE (WRITE\_MODE = 1):

Figure 23-14 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.

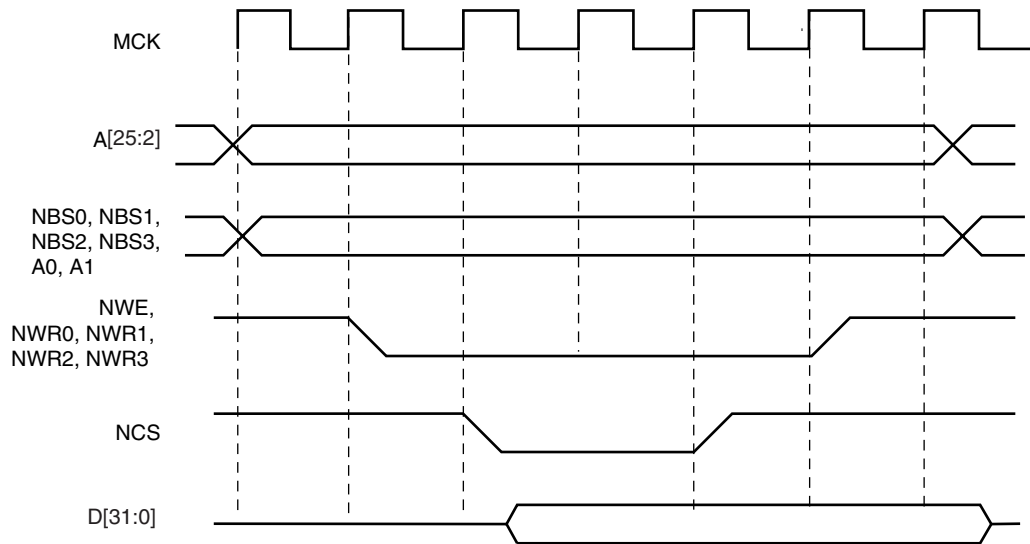
**Figure 23-14.** WRITE\_MODE = 1. The write operation is controlled by NWE



### 23.8.4.2 Write is Controlled by NCS (WRITE\_MODE = 0)

Figure 23-15 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 23-15.** WRITE\_MODE = 0. The write operation is controlled by NCS



### 23.8.5 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 23-4 shows how the timing parameters are coded and their permitted range.

**Table 23-4.** Coding and Range of Timing Parameters

Coded Value	Number of Bits	Effective Value	Permitted Range	
			Coded Value	Effective Value
setup [5:0]	6	$128 \times \text{setup}[5] + \text{setup}[4:0]$	$0 \leq \leq 31$	$128 \leq \leq 128+31$
pulse [6:0]	7	$256 \times \text{pulse}[6] + \text{pulse}[5:0]$	$0 \leq \leq 63$	$256 \leq \leq 256+63$
cycle [8:0]	9	$256 \times \text{cycle}[8:7] + \text{cycle}[6:0]$	$0 \leq \leq 127$	$256 \leq \leq 256+127$ $512 \leq \leq 512+127$ $768 \leq \leq 768+127$

### 23.8.6 Reset Values of Timing Parameters

Table 23-5 gives the default value of timing parameters at reset.

**Table 23-5.** Reset Values of Timing Parameters

Register	Reset Value	
SMC_SETUP	0x00000000	All setup timings are set to 1
SMC_PULSE	0x01010101	All pulse timings are set to 1
SMC_CYCLE	0x00010001	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

### 23.8.7 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 these 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 193.

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.

## 23.9 Automatic Wait States

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

### 23.9.1 Chip Select Wait States

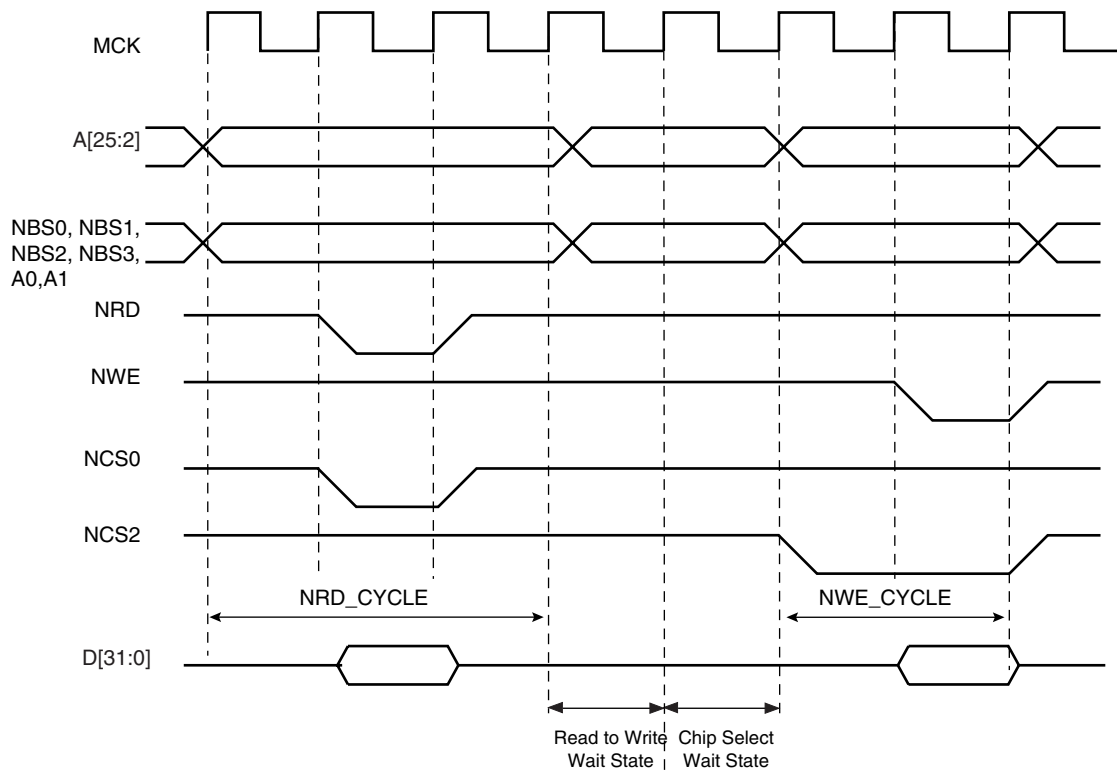
The SMC always inserts an idle cycle between 2 transfers on separate chip selects. This idle cycle ensures that there is no bus contention between the de-activation of one device and the activation of the next one.

During chip select wait state, all control lines are turned inactive: NBS0 to NBS3, NWR0 to NWR3, NCS[0..5], NRD lines are all set to 1.

[Figure 23-16](#) illustrates a chip select wait state between access on Chip Select 0 and Chip Select 2.



**Figure 23-16.** Chip Select Wait State between a Read Access on NCS0 and a Write Access on NCS2



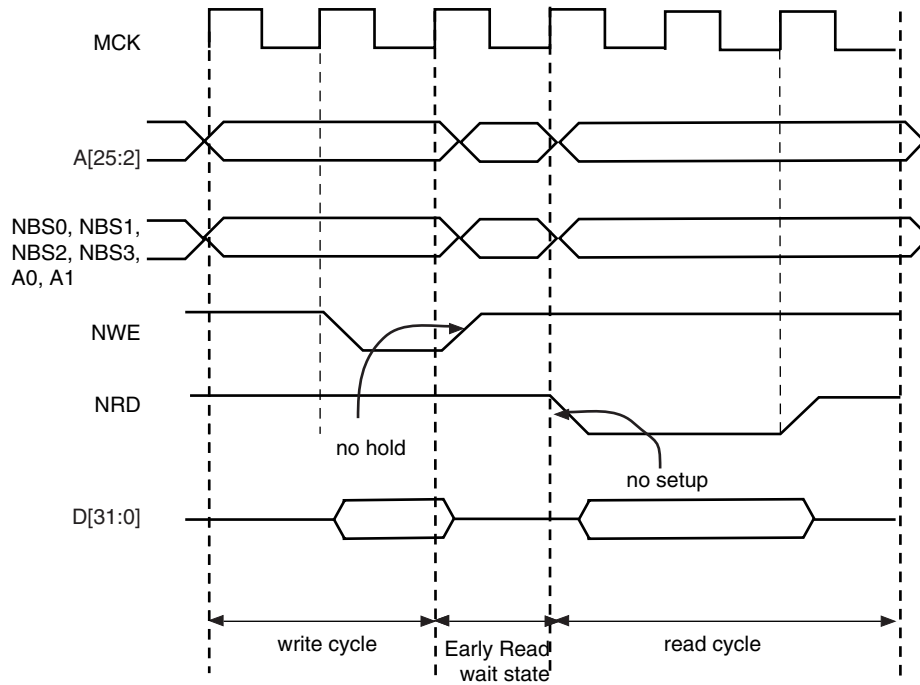
## 23.9.2 Early Read Wait State

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).

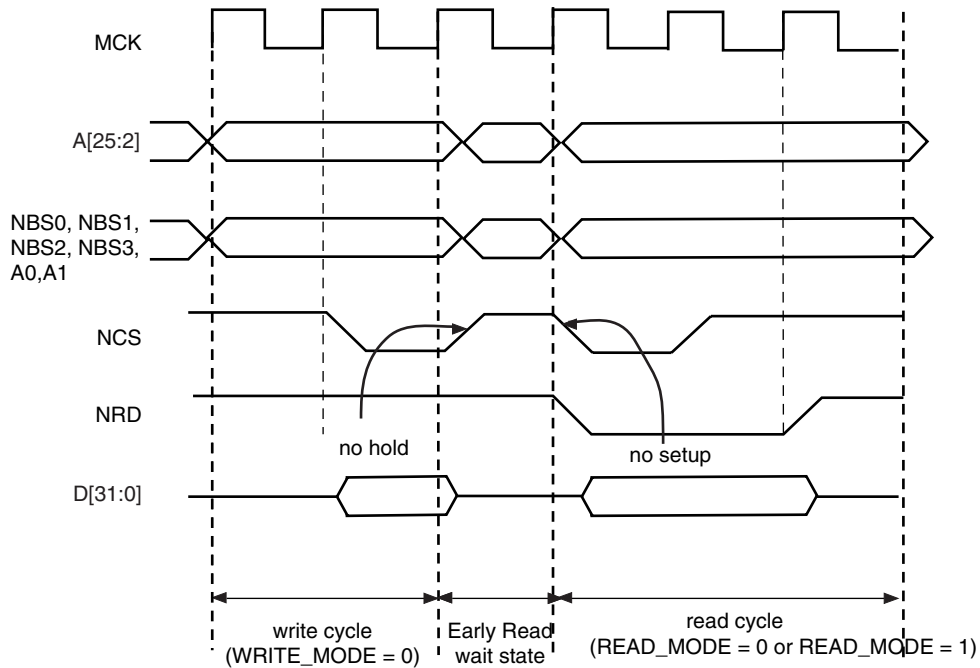
An early read wait state is automatically inserted if at least one of the following conditions is valid:

- if the write controlling signal has no hold time and the read controlling signal has no setup time (Figure 23-17).
- 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 23-18). 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 23-19.

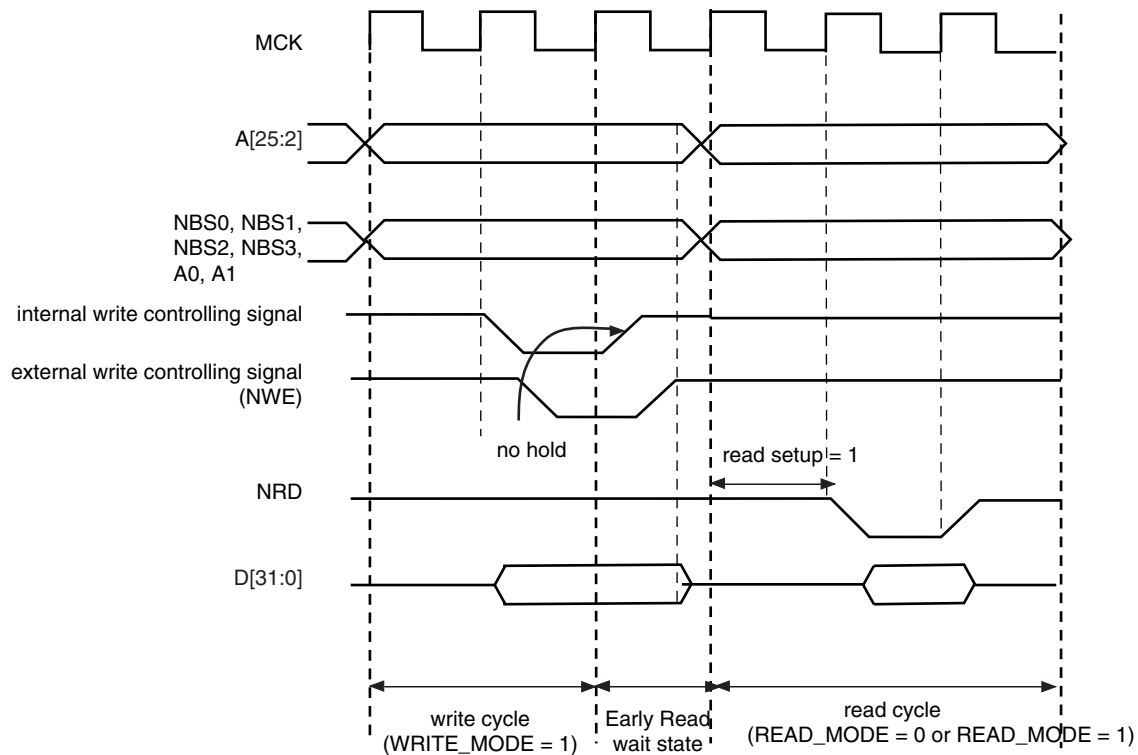
**Figure 23-17.** Early Read Wait State: Write with No Hold Followed by Read with No Setup



**Figure 23-18.** Early Read Wait State: NCS Controlled Write with No Hold Followed by a Read with No NCS Setup



**Figure 23-19.** Early Read Wait State: NWE-controlled Write with No Hold Followed by a Read with one Set-up Cycle



### 23.9.3 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.

#### 23.9.3.1 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.

#### 23.9.3.2 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 207).

#### 23.9.4 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 23-16 on page 193](#).

## 23.10 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.

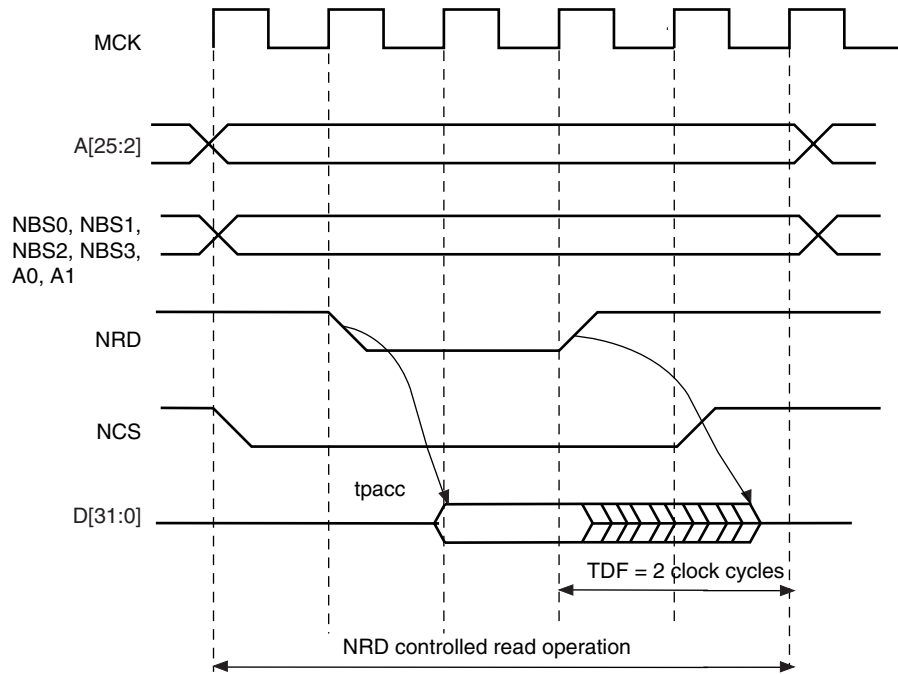
### 23.10.1 READ\_MODE

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.

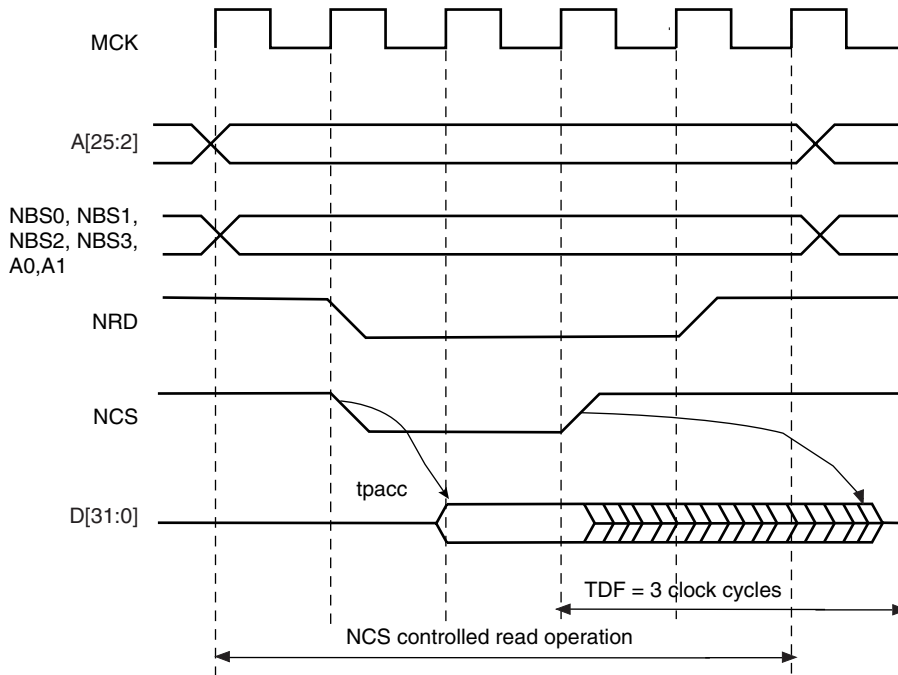
When the read operation is controlled by the NCS signal (READ\_MODE = 0), the TDF field gives the number of MCK cycles during which the data bus remains busy after the rising edge of NCS.

[Figure 23-20](#) illustrates the Data Float Period in NRD-controlled mode (READ\_MODE = 1), assuming a data float period of 2 cycles (TDF\_CYCLES = 2). [Figure 23-21](#) shows the read operation when controlled by NCS (READ\_MODE = 0) and the TDF\_CYCLES parameter equals 3.

**Figure 23-20. TDF Period in NRD Controlled Read Access (TDF = 2)**



**Figure 23-21. TDF Period in NCS Controlled Read Operation (TDF = 3)**



## 23.10.2 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.

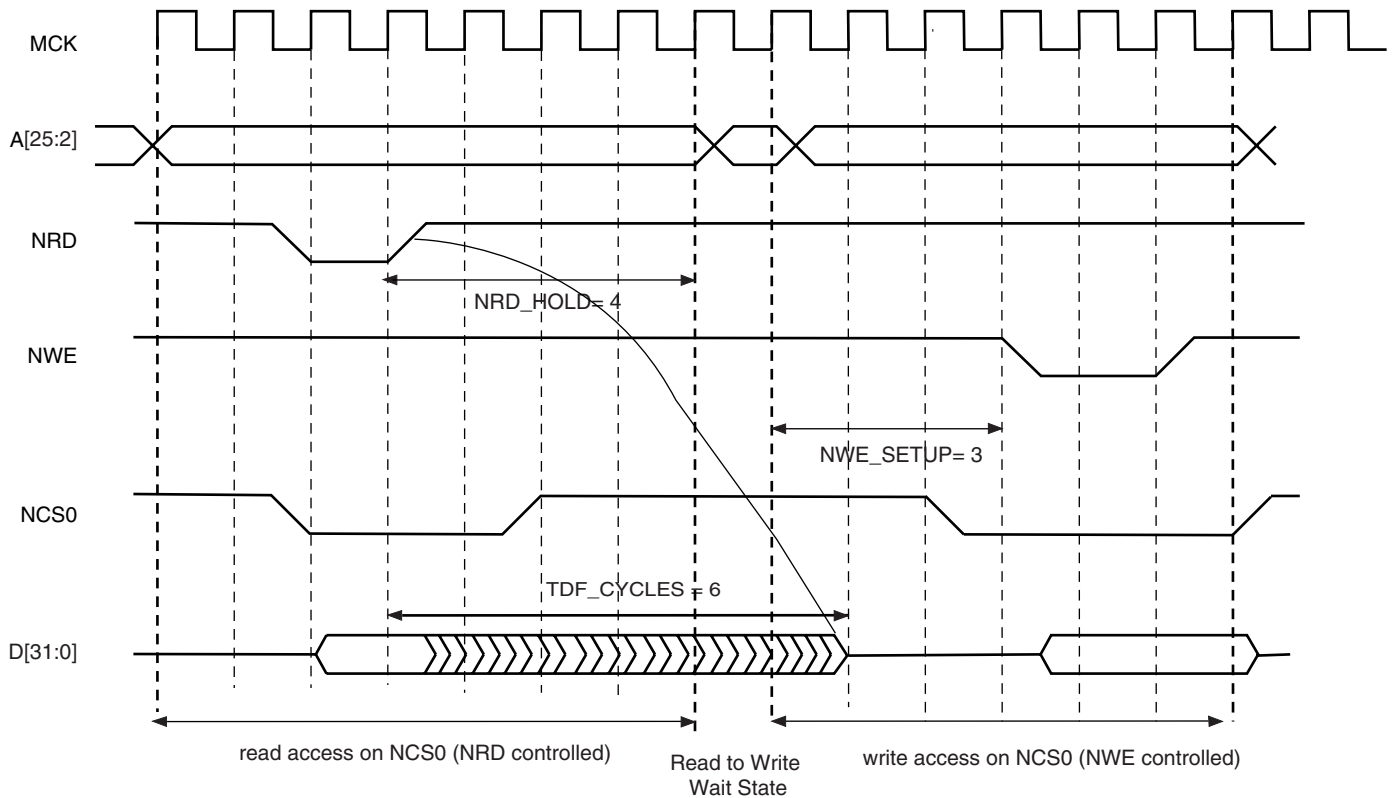
Figure 23-22 shows a read access controlled by NRD, followed by a write access controlled by NWE, on Chip Select 0. Chip Select 0 has been programmed with:

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 23-22.** TDF Optimization: No TDF wait states are inserted if the TDF period is over when the next access begins



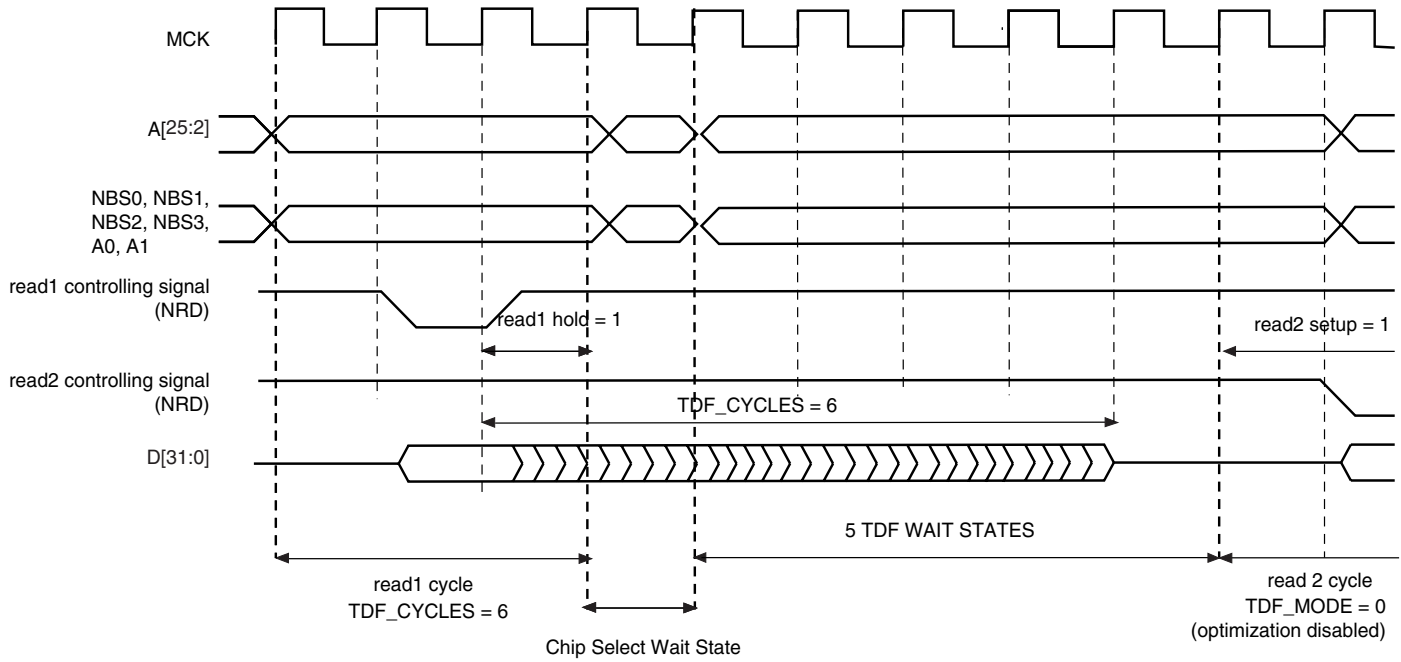
## 23.10.3 TDF Optimization Disabled (TDF\_MODE = 0)

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.

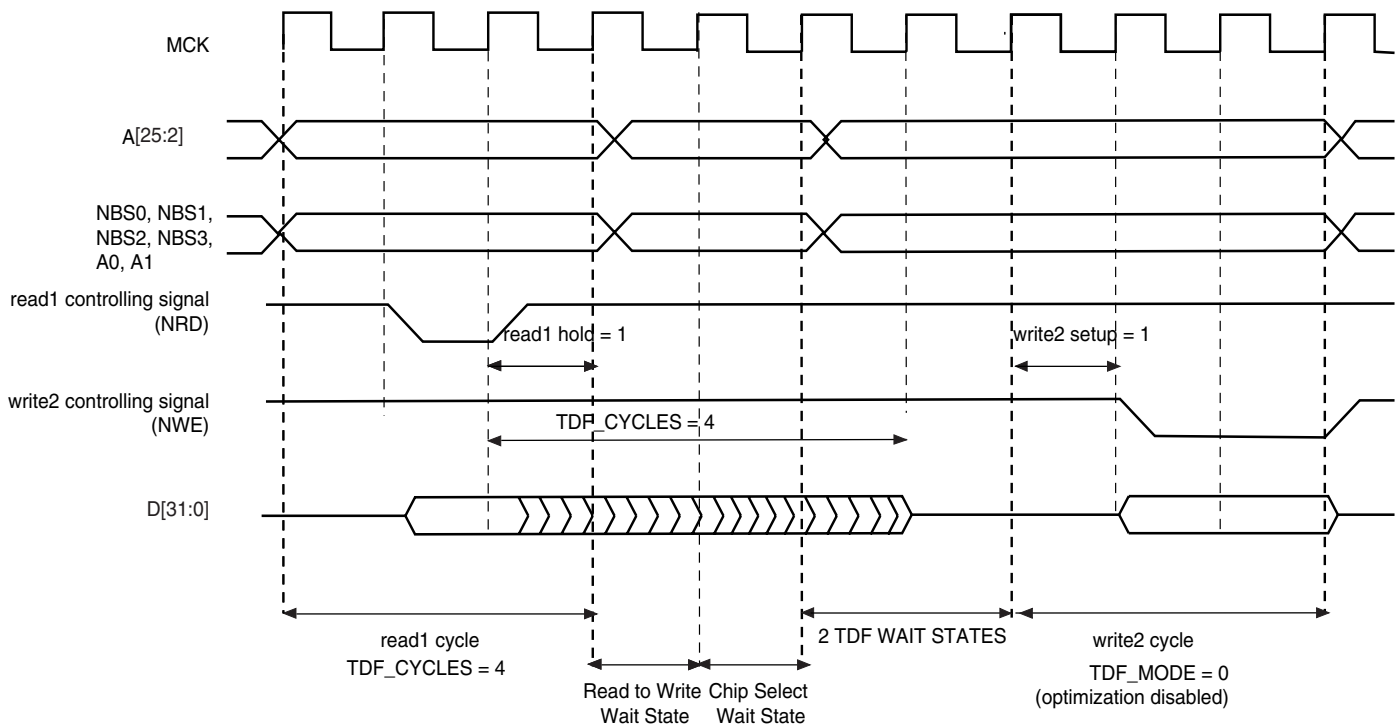
Figure 23-23, Figure 23-24 and Figure 23-25 illustrate the cases:

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

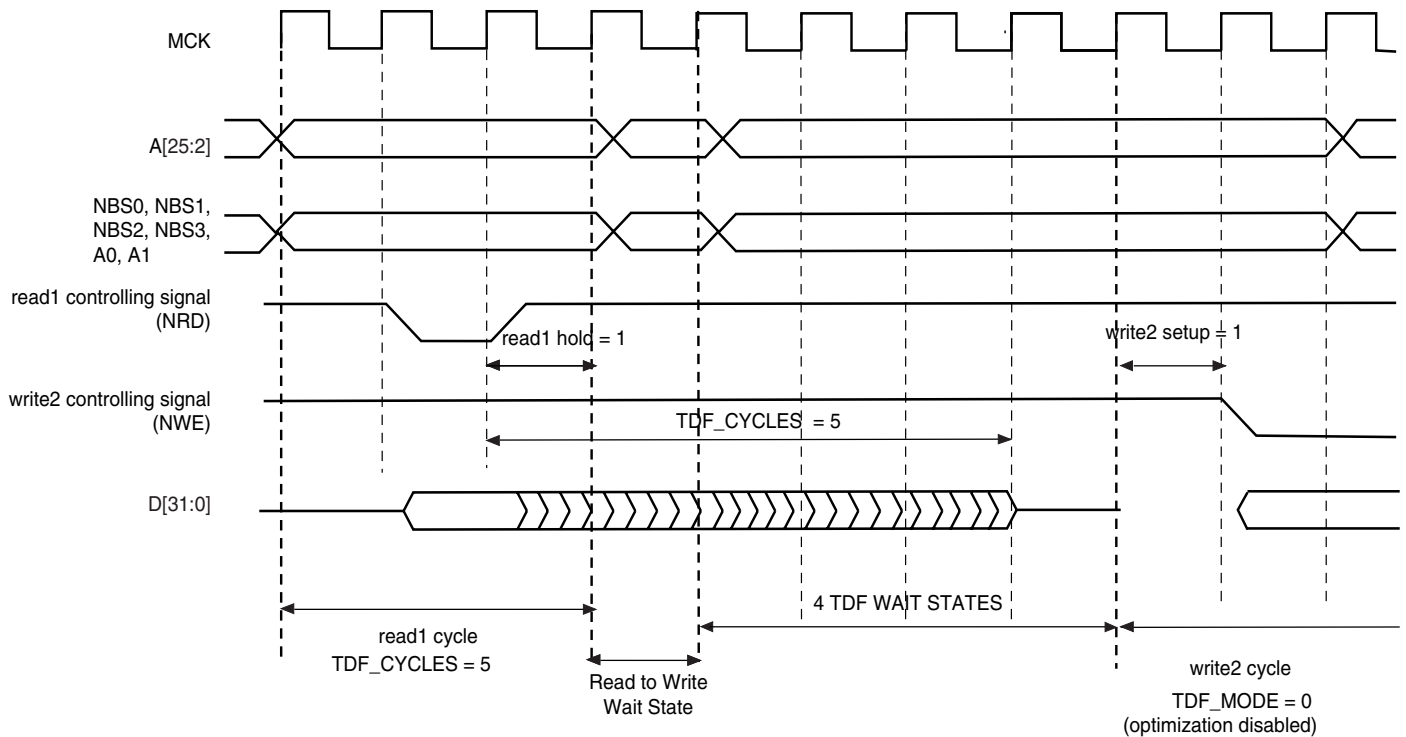


**Figure 23-24.** TDF Mode = 0: TDF wait states between a read and a write access on different chip selects





**Figure 23-25.** TDF Mode = 0: TDF wait states between read and write accesses on the same chip select



## 23.11 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.

### 23.11.1 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 210), or in Slow Clock Mode (“Slow Clock Mode” on page 207).**

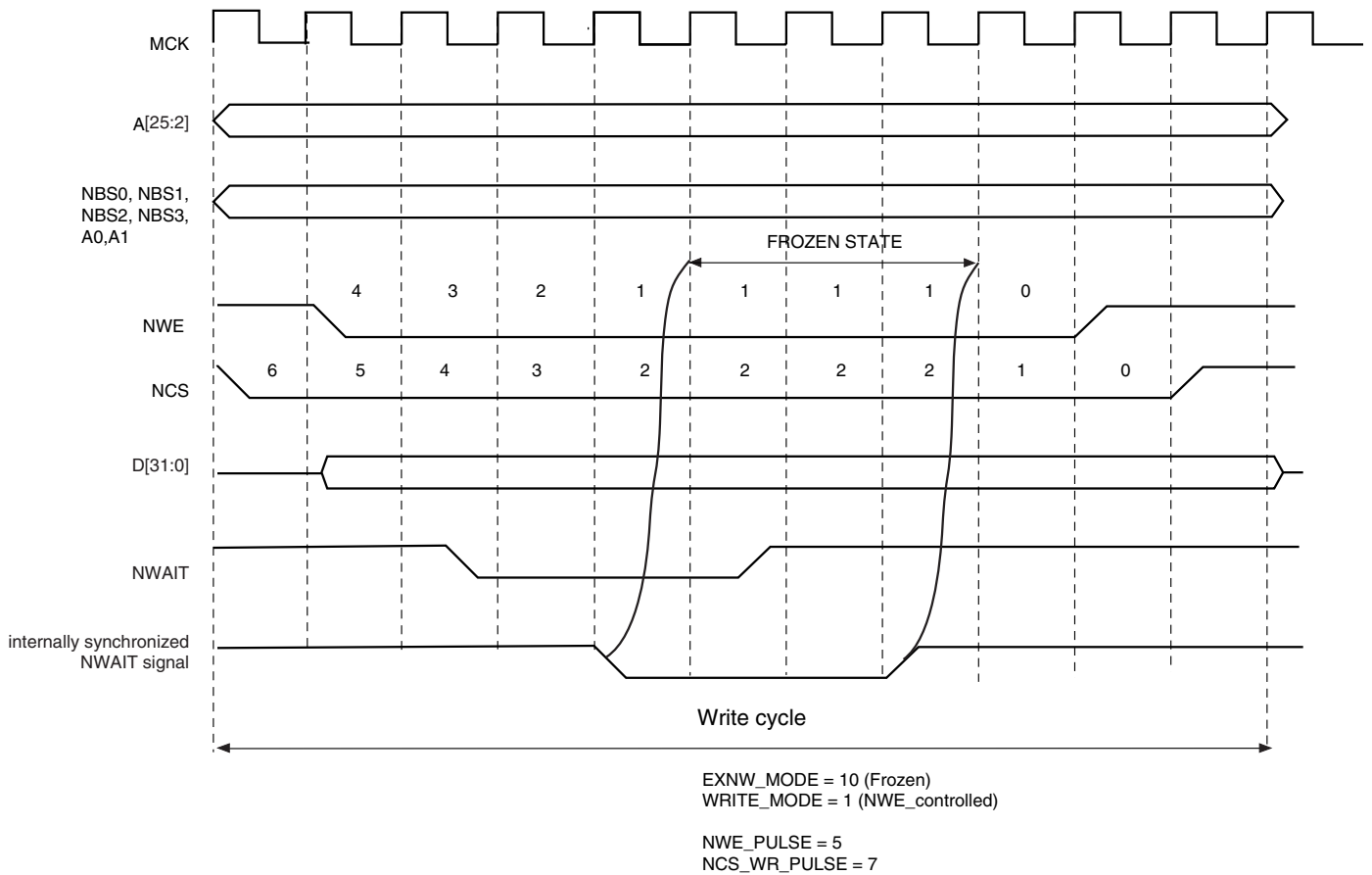
The NWAIT signal is assumed to be a response of the external device to the read/write request of the SMC. Then NWAIT is examined by the SMC only in the pulse state of the read or write controlling signal. The assertion of the NWAIT signal outside the expected period has no impact on SMC behavior.

### 23.11.2 Frozen Mode

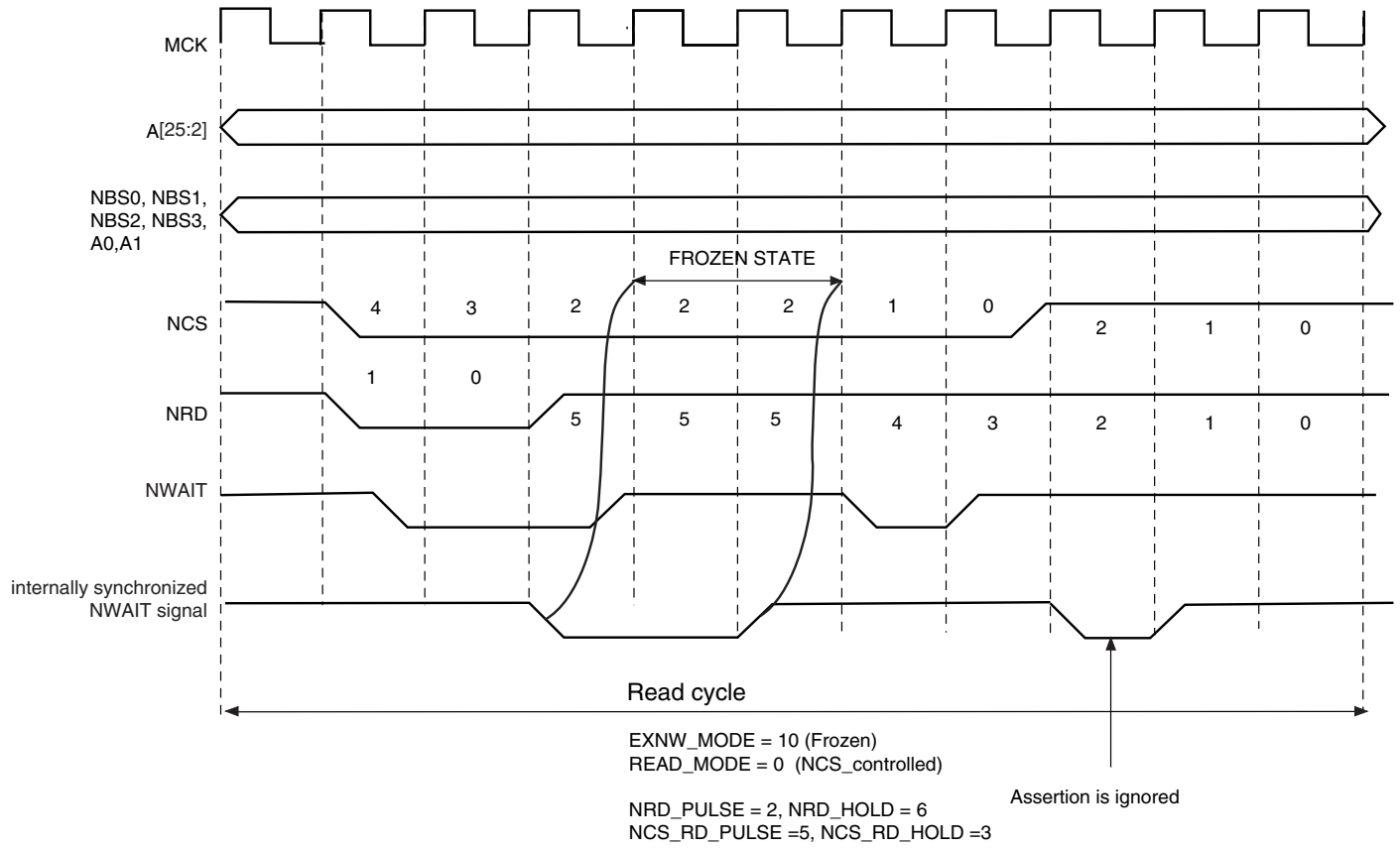
When the external device asserts the NWAIT signal (active low), and after internal synchronization of this signal, the SMC state is frozen, i.e., SMC internal counters are frozen, and all control signals remain unchanged. When the resynchronized NWAIT signal is deasserted, the SMC completes the access, resuming the access from the point where it was stopped. See [Figure 23-26](#). 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 23-27](#).

**Figure 23-26.** Write Access with NWAIT Assertion in Frozen Mode (EXNW\_MODE = 10)



**Figure 23-27. Read Access with NWAIT Assertion in Frozen Mode (EXNW\_MODE = 10)**



## 23.11.3 Ready Mode

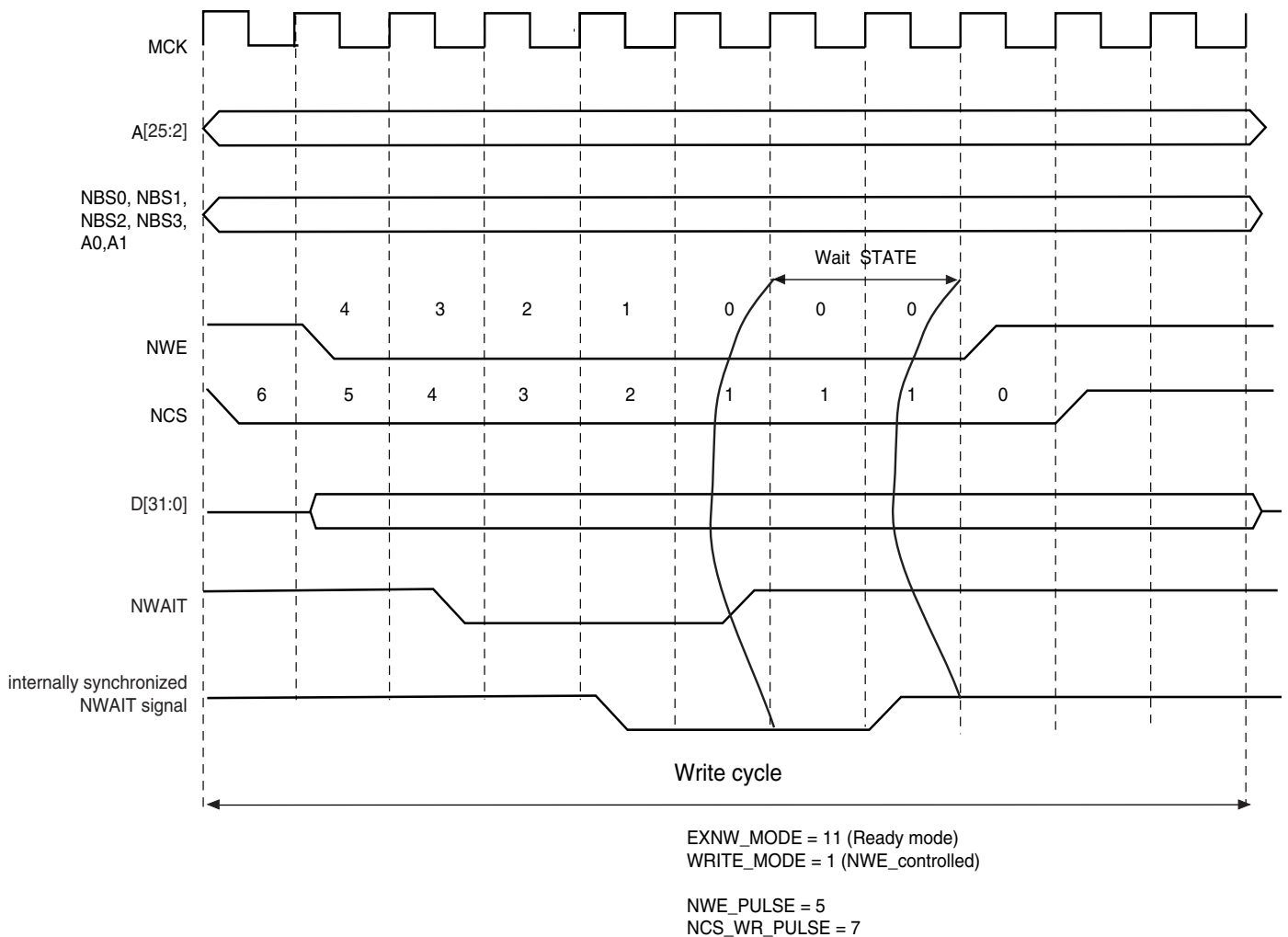
In Ready mode (`EXNW_MODE = 11`), the SMC behaves differently. Normally, the SMC begins the access by down counting the setup and pulse counters of the read/write controlling signal. In the last cycle of the pulse phase, the resynchronized `NWAIT` signal is examined.

If asserted, the SMC suspends the access as shown in [Figure 23-28](#) and [Figure 23-29](#). After deassertion, the access is completed: the hold step of the access is performed.

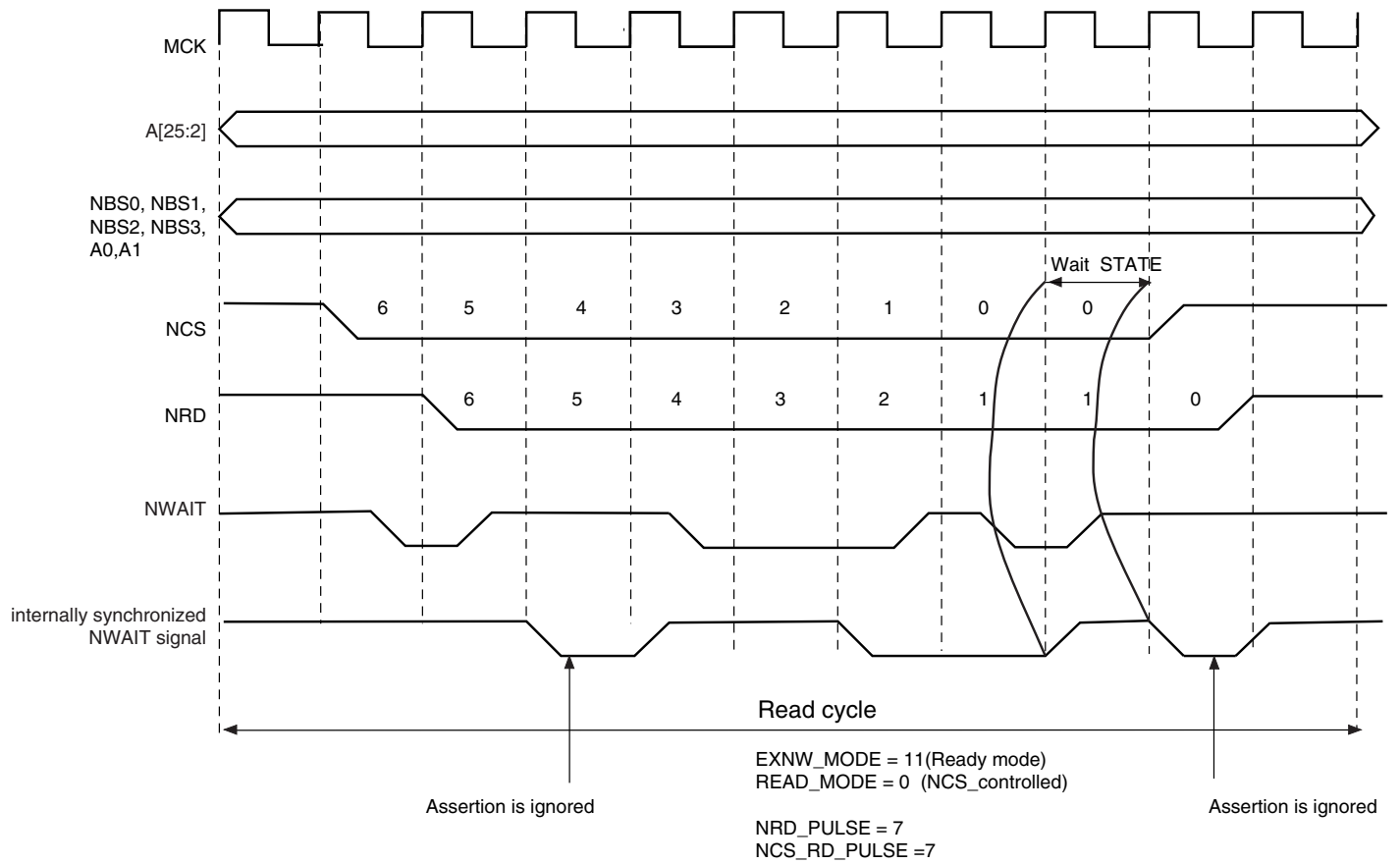
This mode must be selected when the external device uses deassertion of the `NWAIT` signal to indicate its ability to complete the read or write operation.

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 23-29](#).

**Figure 23-28.** `NWAIT` Assertion in Write Access: Ready Mode (`EXNW_MODE = 11`)



**Figure 23-29. NWAIT Assertion in Read Access: Ready Mode (EXNW\_MODE = 11)**



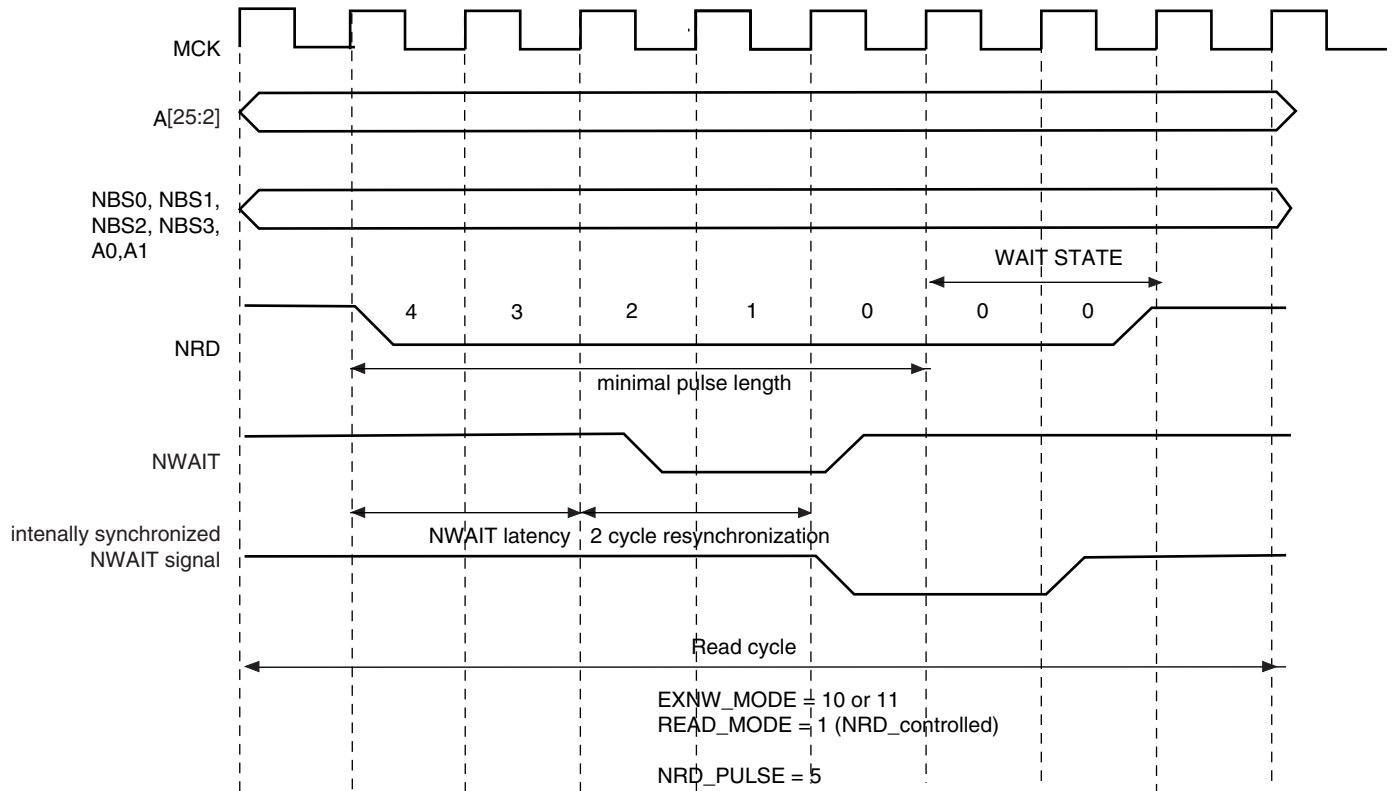
## 23.11.4 NWAIT Latency and Read/write Timings

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 23-30](#).

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:

$$\text{minimal pulse length} = \text{NWAIT latency} + 2 \text{ resynchronization cycles} + 1 \text{ cycle}$$

**Figure 23-30. NWAIT Latency**



## 23.12 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.

### 23.12.1 Slow Clock Mode Waveforms

Figure 23-31 illustrates the read and write operations in slow clock mode. They are valid on all chip selects. Table 23-6 indicates the value of read and write parameters in slow clock mode.

Figure 23-31. Read/write Cycles in Slow Clock Mode

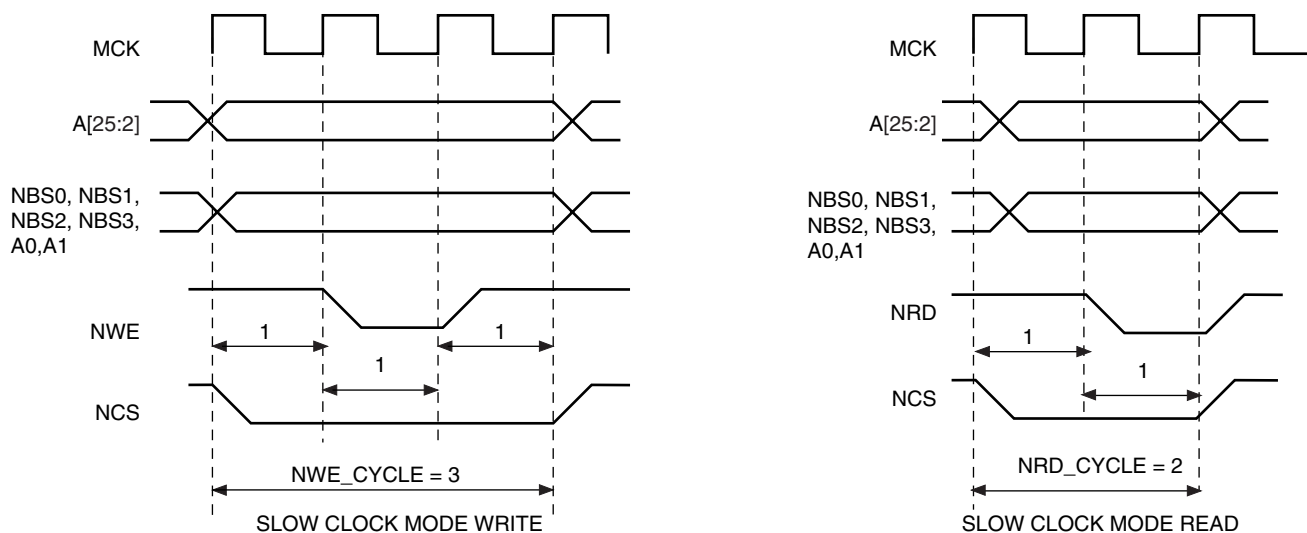


Table 23-6. Read and Write Timing Parameters in Slow Clock Mode

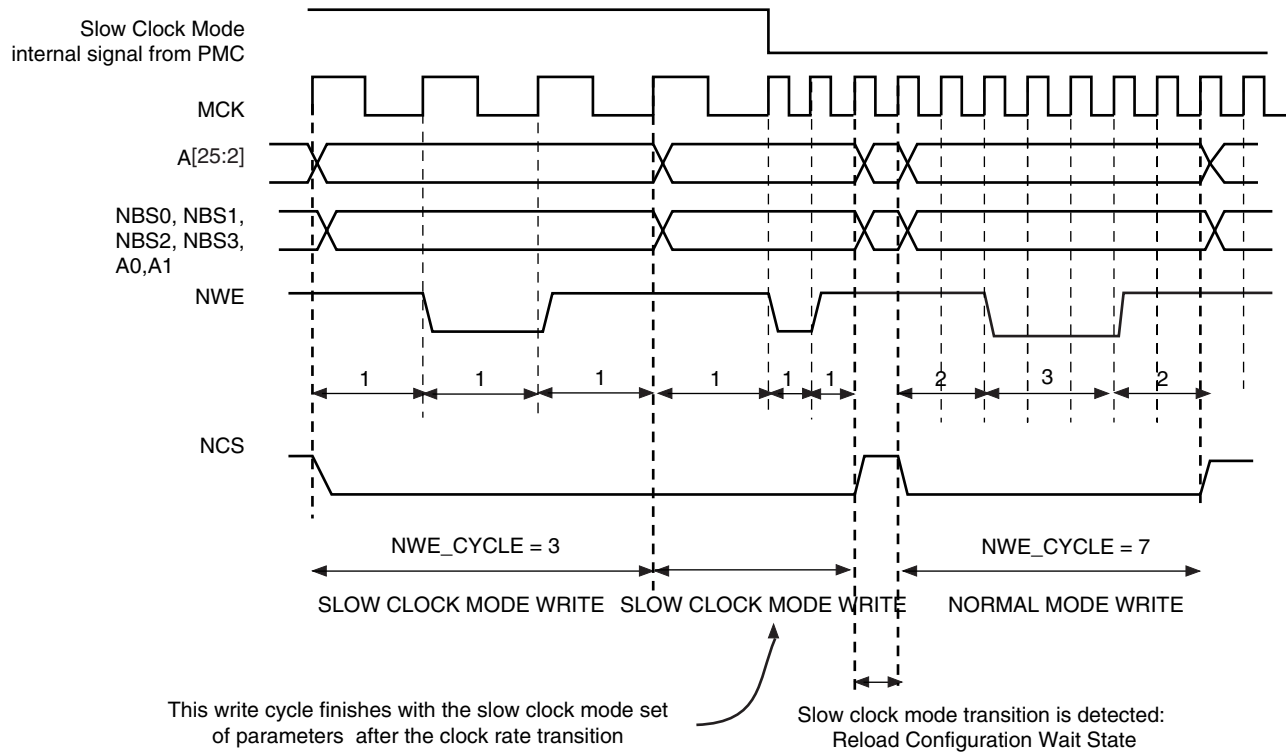
Read Parameters	Duration (cycles)	Write Parameters	Duration (cycles)
NRD_SETUP	1	NWE_SETUP	1
NRD_PULSE	1	NWE_PULSE	1
NCS_RD_SETUP	0	NCS_WR_SETUP	0
NCS_RD_PULSE	2	NCS_WR_PULSE	3
NRD_CYCLE	2	NWE_CYCLE	3

## 23.12.2 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 23-32 on page 208](#). The external device may not be fast enough to support such timings.

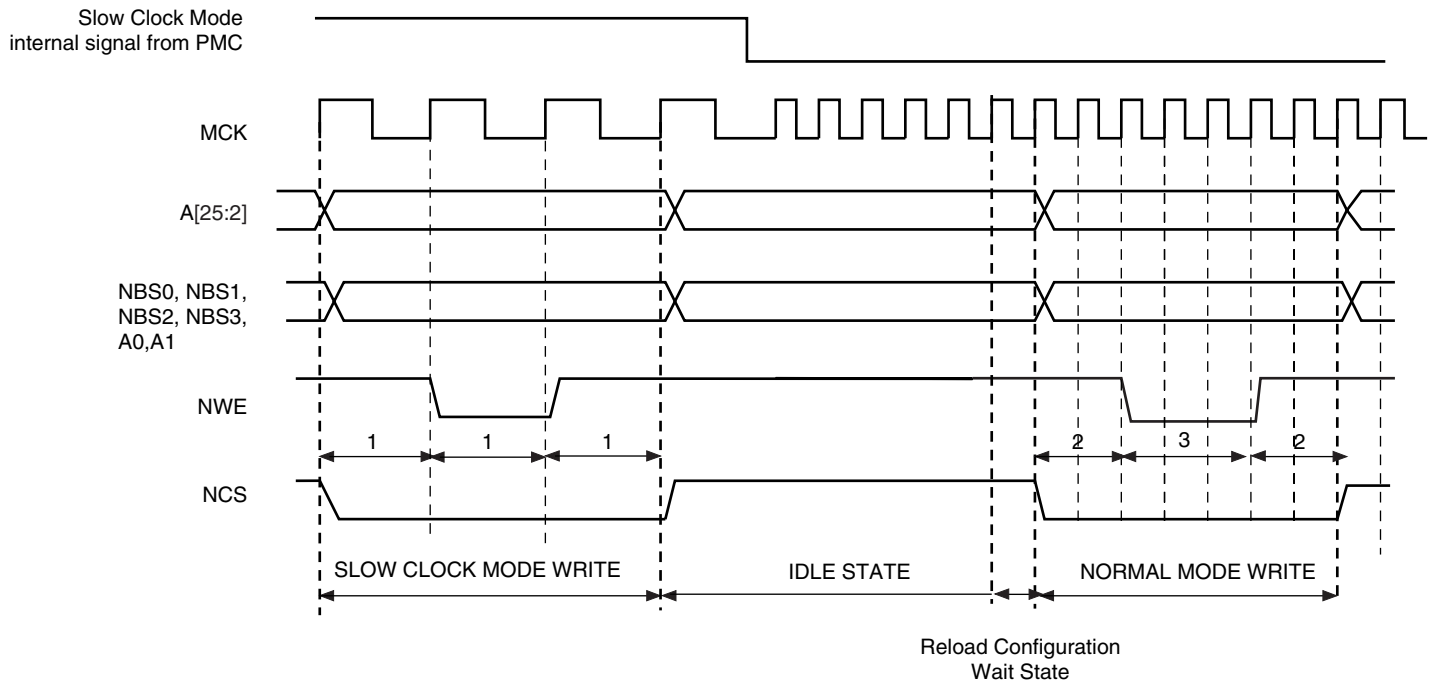
[Figure 23-33](#) illustrates the recommended procedure to properly switch from one mode to the other.

**Figure 23-32.** Clock Rate Transition Occurs while the SMC is Performing a Write Operation





**Figure 23-33.** Recommended Procedure to Switch from Slow Clock Mode to Normal Mode or from Normal Mode to Slow Clock Mode



## 23.13 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 23-7](#).

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 23-34](#). 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 23-7.** Page Address and Data Address within a Page

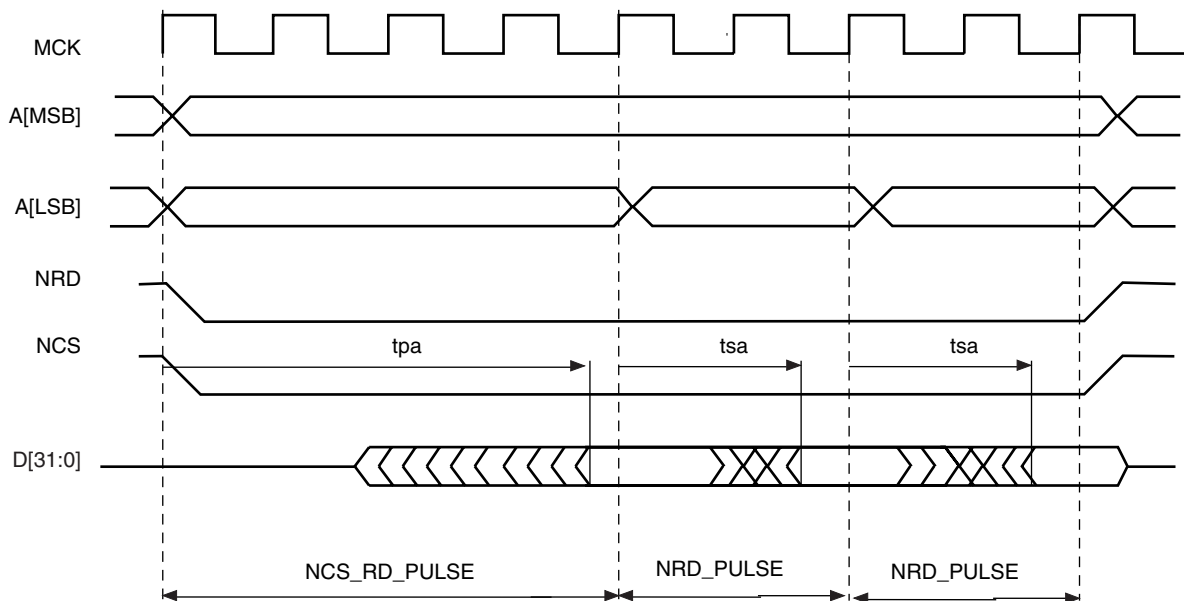
Page Size	Page Address <sup>(1)</sup>	Data Address in the Page <sup>(2)</sup>
4 bytes	A[25:2]	A[1:0]
8 bytes	A[25:3]	A[2:0]
16 bytes	A[25:4]	A[3:0]
32 bytes	A[25:5]	A[4:0]

- 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.

### 23.13.1 Protocol and Timings in Page Mode

[Figure 23-34](#) shows the NRD and NCS timings in page mode access.

**Figure 23-34.** Page Mode Read Protocol (Address MSB and LSB are defined in [Table 23-7](#))



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 23-8](#):

**Table 23-8.** 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  $t_{sa}$ .

### 23.13.2 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).

### 23.13.3 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.

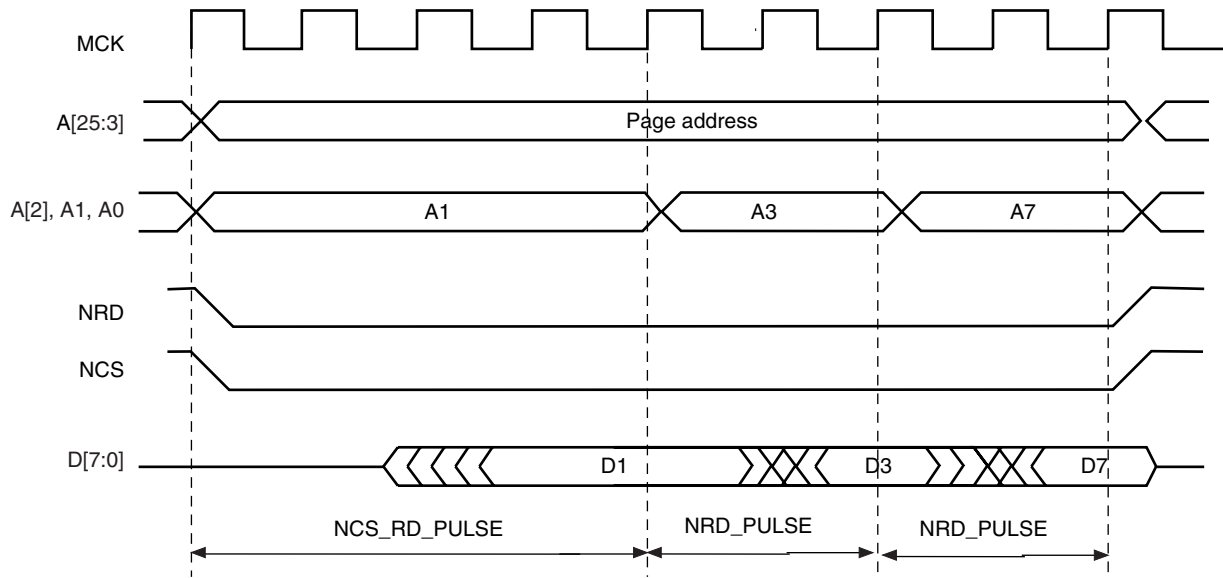
### 23.13.4 Sequential and Non-sequential Accesses

If the chip select and the MSB of addresses as defined in [Table 23-7](#) 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 23-35](#) 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 23-35.** Access to Non-sequential Data within the Same Page



## 23.14 Static Memory Controller (SMC) User Interface

The SMC is programmed using the registers listed in [Table 23-9](#). For each chip select, a set of 4 registers is used to program the parameters of the external device connected on it. In [Table 23-9](#), “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 23-9.** SMC Register Mapping

Offset	Register	Name	Access	Reset State
0x10 x CS_number + 0x00	SMC Setup Register	SMC_SETUP	Read/Write	0x00000000
0x10 x CS_number + 0x04	SMC Pulse Register	SMC_PULSE	Read/Write	0x01010101
0x10 x CS_number + 0x08	SMC Cycle Register	SMC_CYCLE	Read/Write	0x00010001
0x10 x CS_number + 0x0C	SMC Mode Register	SMC_MODE	Read/Write	0x10001000

## 23.14.1 SMC Setup Register

**Register Name:** SMC\_SETUP[0 ..5]

**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:

$$\text{NWE setup length} = (128 * \text{NWE\_SETUP}[5] + \text{NWE\_SETUP}[4:0]) \text{ clock cycles}$$

- **NCS\_WR\_SETUP: NCS Setup Length in WRITE Access**

In write access, the NCS signal setup length is defined as:

$$\text{NCS setup length} = (128 * \text{NCS\_WR\_SETUP}[5] + \text{NCS\_WR\_SETUP}[4:0]) \text{ clock cycles}$$

- **NRD\_SETUP: NRD Setup Length**

The NRD signal setup length is defined in clock cycles as:

$$\text{NRD setup length} = (128 * \text{NRD\_SETUP}[5] + \text{NRD\_SETUP}[4:0]) \text{ clock cycles}$$

- **NCS\_RD\_SETUP: NCS Setup Length in READ Access**

In read access, the NCS signal setup length is defined as:

$$\text{NCS setup length} = (128 * \text{NCS\_RD\_SETUP}[5] + \text{NCS\_RD\_SETUP}[4:0]) \text{ clock cycles}$$

## 23.14.2 SMC Pulse Register

**Register Name:** SMC\_PULSE[0..5]

**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:

$$\text{NWE pulse length} = (256 * \text{NWE\_PULSE}[6] + \text{NWE\_PULSE}[5:0]) \text{ clock cycles}$$

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:

$$\text{NCS pulse length} = (256 * \text{NCS\_WR\_PULSE}[6] + \text{NCS\_WR\_PULSE}[5:0]) \text{ clock cycles}$$

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:

$$\text{NRD pulse length} = (256 * \text{NRD\_PULSE}[6] + \text{NRD\_PULSE}[5:0]) \text{ clock cycles}$$

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:

$$\text{NCS pulse length} = (256 * \text{NCS\_RD\_PULSE}[6] + \text{NCS\_RD\_PULSE}[5:0]) \text{ clock cycles}$$

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.

## 23.14.3 SMC Cycle Register

**Register Name:** SMC\_CYCLE[0..5]

**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



## 23.14.4 SMC MODE Register

**Register Name:** SMC\_MODE[0..5]

**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

## 24. SDRAM Controller (SDRAMC)

### 24.1 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.

### 24.2 I/O Lines Description

**Table 24-1.** 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	

## 24.3 Application Example

### 24.3.1 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 24-2](#) to [Table 24-7](#) illustrate the SDRAM device memory mapping seen by the user in correlation with the device structure. Various configurations are illustrated.

### 24.3.2 32-bit Memory Data Bus Width

**Table 24-2.** 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 24-3.** 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 24-4.** 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.

## 24.3.3 16-bit Memory Data Bus Width

**Table 24-5.** 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 24-6.** 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 24-7.** 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.

## 24.4 Product Dependencies

### 24.4.1 SDRAM Device Initialization

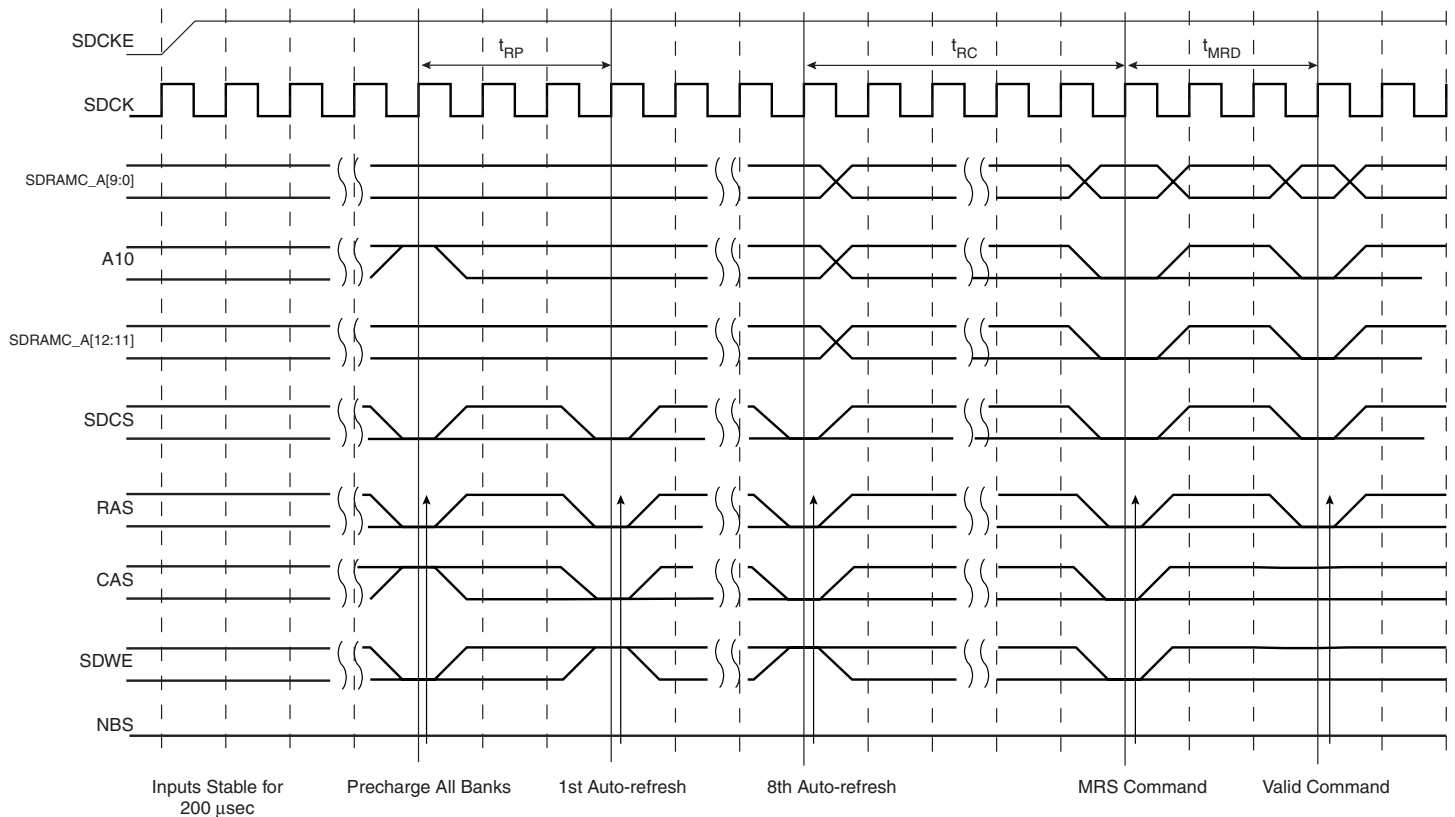
The initialization sequence is generated by software. The SDRAM devices are initialized by the following sequence:

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  $\mu$ s is provided to precede any signal toggle.
5. <sup>(1)</sup>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  $\mu$ s or 7.81  $\mu$ s. With a 100 MHz frequency, the Refresh Timer Counter Register must be set with the value 1562(15.652  $\mu$ s x 100 MHz) or 781(7.81  $\mu$ s 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 24-1.** SDRAM Device Initialization Sequence



## 24.4.2 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.

## 24.4.3 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).

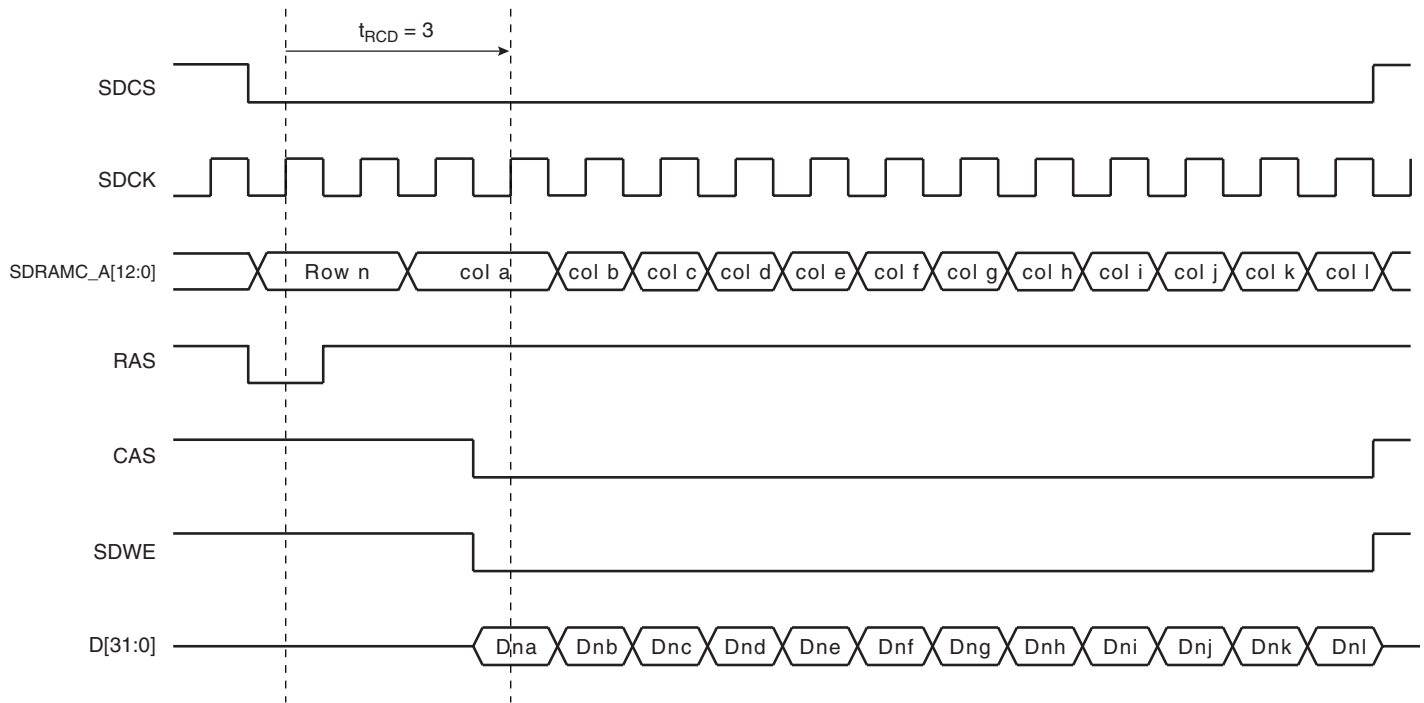
Using the SDRAM Controller interrupt requires the AIC to be programmed first.

## 24.5 Functional Description

### 24.5.1 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 235. This is described in [Figure 24-2](#) below.

**Figure 24-2.** Write Burst, 32-bit SDRAM Access





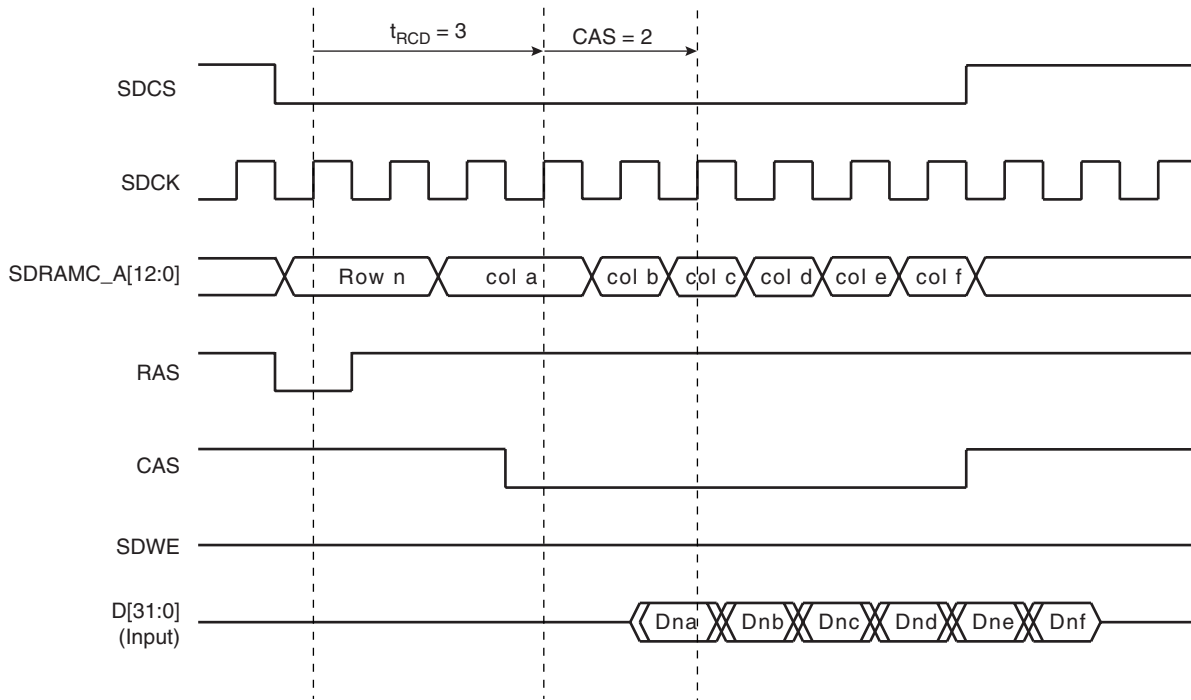
## 24.5.2 SDRAM Controller Read Cycle

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.

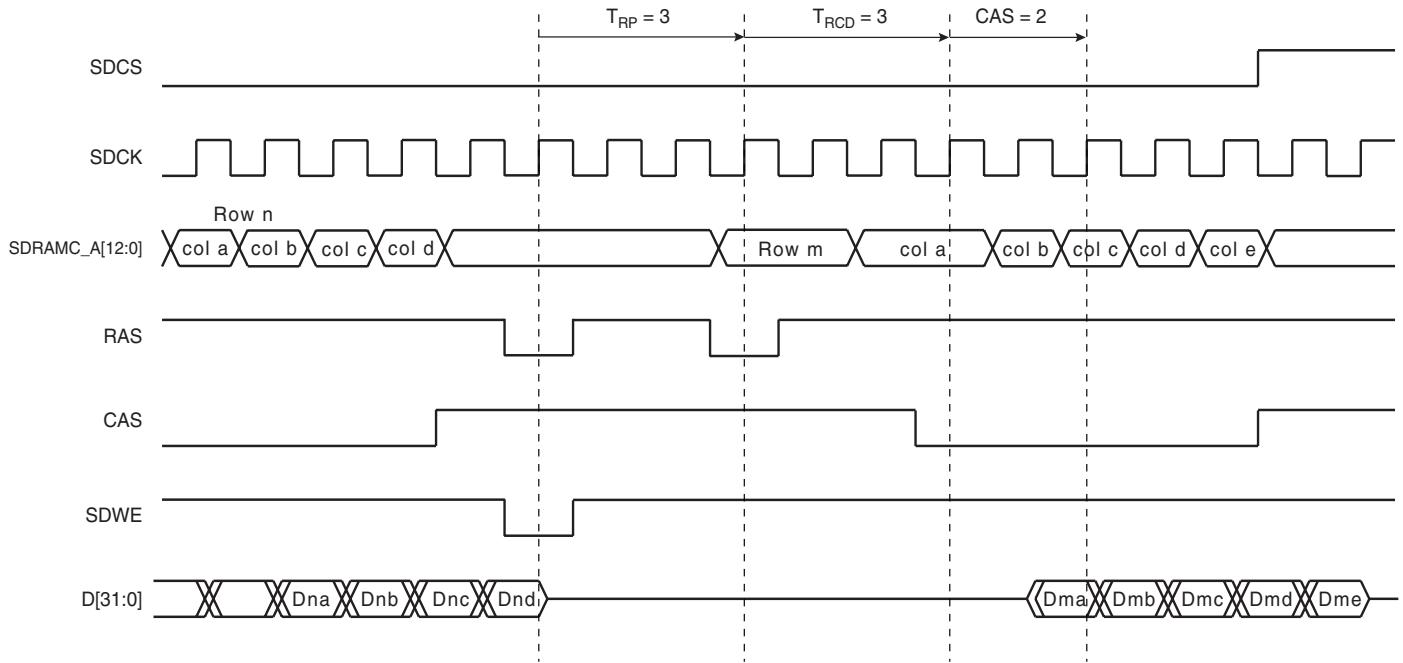
**Figure 24-3.** Read Burst, 32-bit SDRAM Access



### 24.5.3 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 24-4 below.

**Figure 24-4.** Read Burst with Boundary Row Access



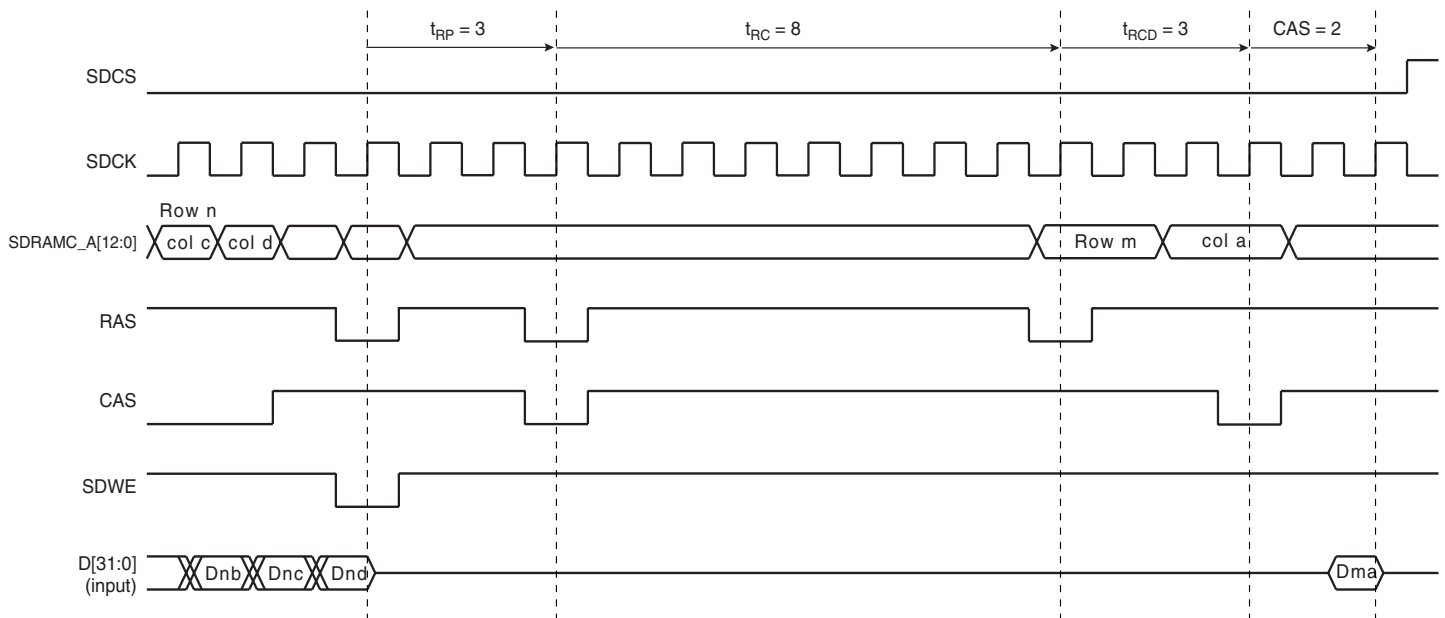
## 24.5.4 SDRAM Controller Refresh Cycles

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 24-5](#).

**Figure 24-5.** Refresh Cycle Followed by a Read Access



### 24.5.5 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.

## 24.5.6 Self-refresh Mode

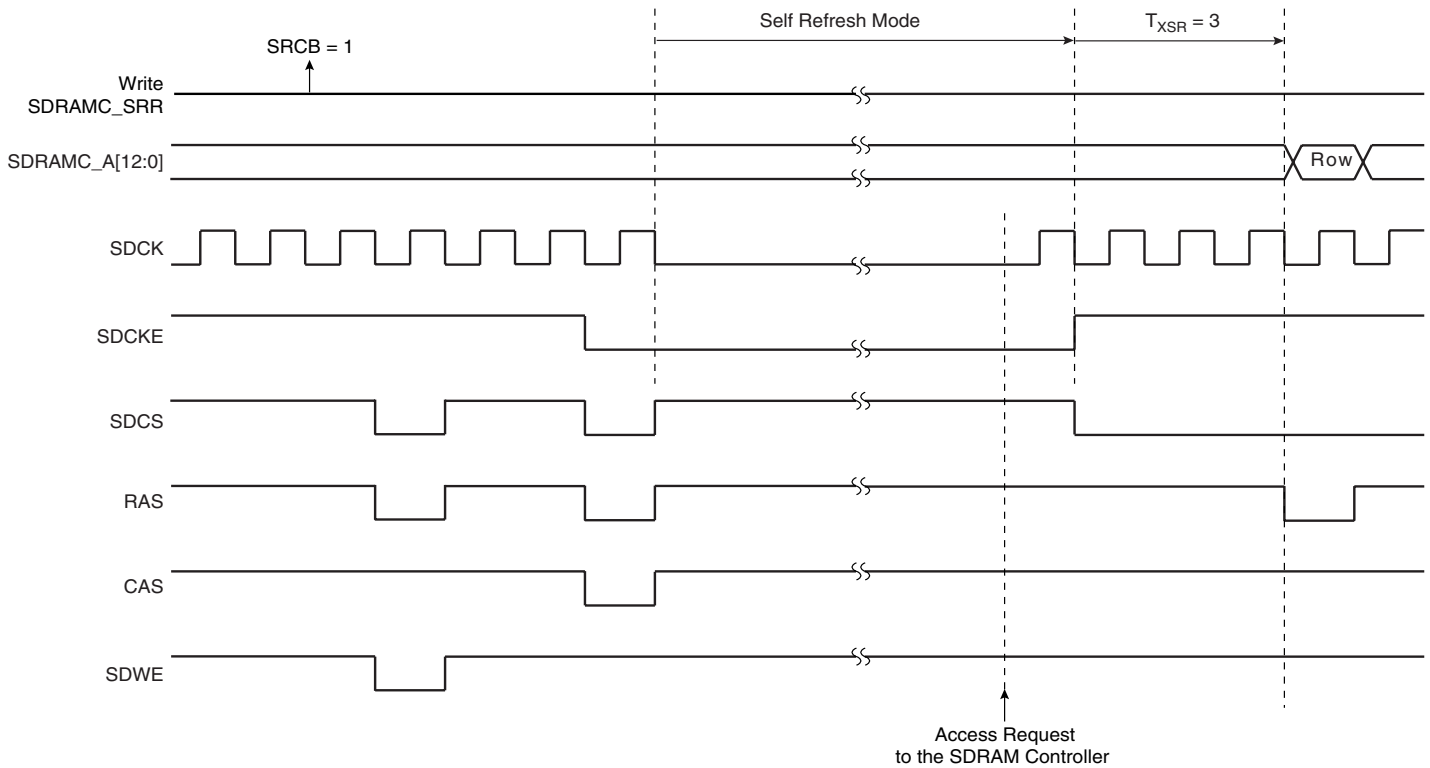
This mode is selected by programming the LPCB field to 1 in the SDRAMC Low Power Register. In self-refresh mode, the SDRAM device retains data without external clocking and provides its own internal clocking, thus performing its own auto-refresh cycles. All the inputs to the SDRAM device become “don’t 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.

After initialization, as soon as PASR/DS/TCSR fields are modified and self-refresh mode is activated, the Extended Mode Register is accessed automatically and PASR/DS/TCSR bits are updated before entry into self-refresh mode.

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 24-6](#).

**Figure 24-6.** Self-refresh Mode Behavior

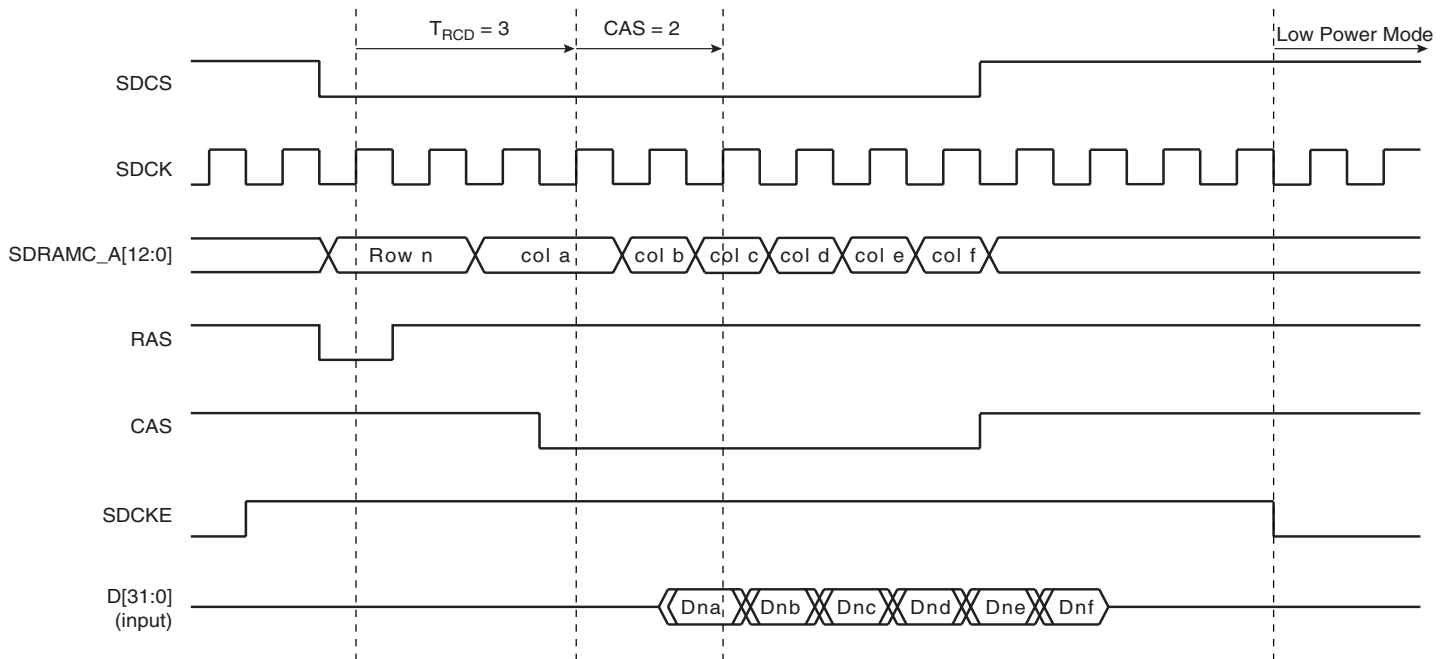


### 24.5.7 Low-power Mode

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 24-7](#).

**Figure 24-7.** Low-power Mode Behavior



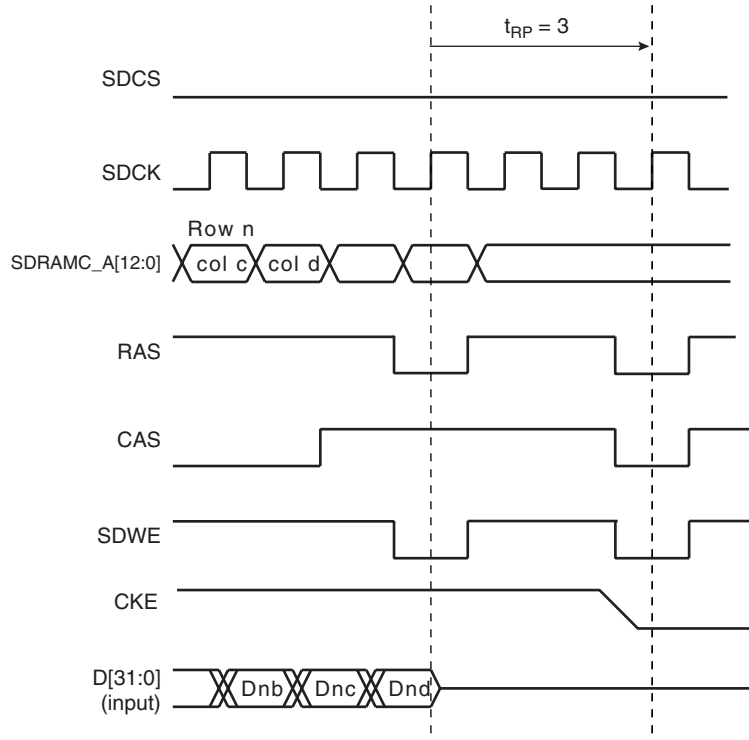
## 24.5.8 Deep Power-down Mode

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 222).

This is described in [Figure 24-8](#).

**Figure 24-8.** Deep Power-down Mode Behavior



## 24.6 SDRAM Controller User Interface

**Table 24-8.** 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
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	–	–	–



## 24.6.1 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.

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.



### 24.6.2 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.



## 24.6.3 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.

## 24.6.4 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.

After initialization, as soon as PASR field is modified and self-refresh mode is activated, the Extended Mode Register is accessed automatically and PASR bits are updated before entry in self-refresh mode.

### • 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.

After initialization, as soon as TCSR field is modified and self-refresh mode is activated, the Extended Mode Register is accessed automatically and TCSR bits are updated before entry in self-refresh mode.

### • 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.

After initialization, as soon as DS field is modified and self-refresh mode is activated, the Extended Mode Register is accessed automatically and DS bits are updated before entry in self-refresh mode.

- **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.

## 24.6.5 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.

## 24.6.6 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.

### 24.6.7 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.

### 24.6.8 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.



## 24.6.9 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.



## 25. Error Corrected Code (ECC) Controller

### 25.1 Description

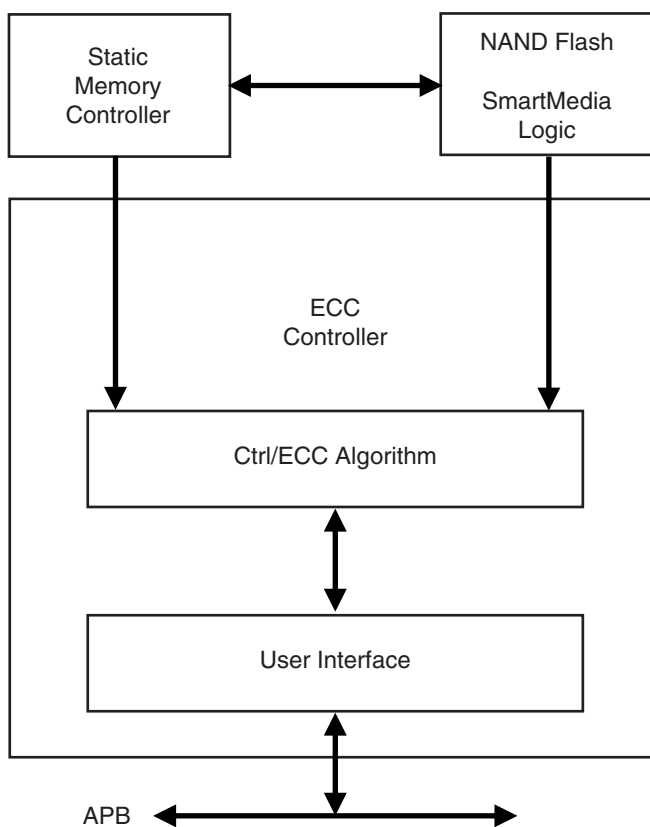
NAND Flash/SmartMedia devices contain by default invalid blocks which have one or more invalid bits. Over the NAND Flash/SmartMedia lifetime, additional invalid blocks may occur which can be detected/corrected by ECC code.

The ECC Controller is a mechanism that encodes data in a manner that makes possible the identification and correction of certain errors in data. The ECC controller is capable of single bit error correction and 2-bit random detection. When NAND Flash/SmartMedia have more than 2 bits of errors, the data cannot be corrected.

The ECC user interface is compliant with the ARM Advanced Peripheral Bus (APB rev2).

### 25.2 Block Diagram

Figure 25-1. Block Diagram



### 25.3 Functional Description

A page in NAND Flash and SmartMedia memories contains an area for main data and an additional area used for redundancy (ECC). The page is organized in 8-bit or 16-bit words. The page size corresponds to the number of words in the main area plus the number of words in the extra area used for redundancy.

The only configuration required for ECC is the NAND Flash or the SmartMedia page size (528/1056/2112/4224). Page size is configured setting the PAGESIZE field in the ECC Mode Register (ECC\_MR).

ECC is automatically computed as soon as a read (00h)/write (80h) command to the NAND Flash or the SmartMedia is detected. Read and write access must start at a page boundary.

ECC results are available as soon as the counter reaches the end of the main area. Values in the ECC Parity Register (ECC\_PR) and ECC NParity Register (ECC\_NPR) are then valid and locked until a new start condition occurs (read/write command followed by address cycles).

### 25.3.1 Write Access

Once the flash memory page is written, the computed ECC code is available in the ECC Parity Error (ECC\_PR) and ECC NParity Error (ECC\_NPR) registers. The ECC code value must be written by the software application in the extra area used for redundancy.

### 25.3.2 Read Access

After reading the whole data in the main area, the application must perform read accesses to the extra area where ECC code has been previously stored. Error detection is automatically performed by the ECC controller. Please note that it is mandatory to read consecutively the entire main area and the locations where Parity and NParity values have been previously stored to let the ECC controller perform error detection.

The application can check the ECC Status Register (ECC\_SR) for any detected errors.

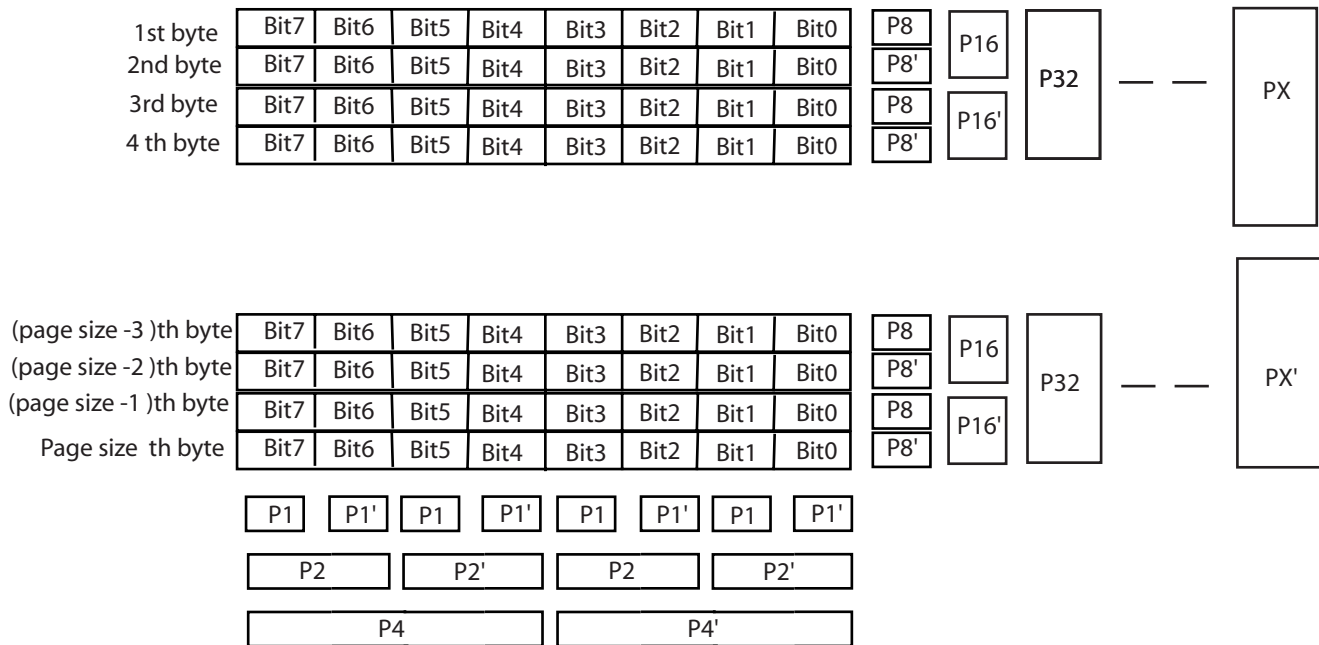
It is up to the application to correct any detected error. ECC computation can detect four different circumstances:

- No error: XOR between the ECC computation and the ECC code stored at the end of the NAND Flash or SmartMedia page is equal to 0. No error flags in the ECC Status Register (ECC\_SR).
- Recoverable error: Only the RECERR flag in the ECC Status register (ECC\_SR) is set. The corrupted word offset in the read page is defined by the WORDADDR field in the ECC Parity Register (ECC\_PR). The corrupted bit position in the concerned word is defined in the BITADDR field in the ECC Parity Register (ECC\_PR).
- ECC error: The ECCERR flag in the ECC Status Register is set. An error has been detected in the ECC code stored in the Flash memory. The position of the corrupted bit can be found by the application performing an XOR between the Parity and the NParity contained in the ECC code stored in the flash memory.
- Non correctable error: The MULERR flag in the ECC Status Register is set. Several unrecoverable errors have been detected in the flash memory page.

ECC Status Register, ECC Parity Register and ECC NParity Register are cleared when a read/write command is detected or a software reset is performed.

For Single-bit Error Correction and Double-bit Error Detection (SEC-DED) hsiao code is used. 32-bit ECC is generated in order to perform one bit correction per 512/1024/2048/4096 8- or 16-bit words. Of the 32 ECC bits, 26 bits are for line parity and 6 bits are for column parity. They are generated according to the schemes shown in [Figure 25-2](#) and [Figure 25-3](#).

**Figure 25-2. Parity Generation for 512/1024/2048/4096 8-bit Words1**



Page size = 512 Px = 2048  
 Page size = 1024 Px = 4096  
 Page size = 2048 Px = 8192  
 Page size = 4096 Px = 16384

P1=bit7(+)+bit5(+)+bit3(+)+bit1(+)+P1  
 P2=bit7(+)+bit6(+)+bit3(+)+bit2(+)+P2  
 P4=bit7(+)+bit6(+)+bit5(+)+bit4(+)+P4  
 P1'=bit6(+)+bit4(+)+bit2(+)+bit0(+)+P1'  
 P2'=bit5(+)+bit4(+)+bit1(+)+bit0(+)+P2'  
 P4'=bit7(+)+bit6(+)+bit5(+)+bit4(+)+P4'

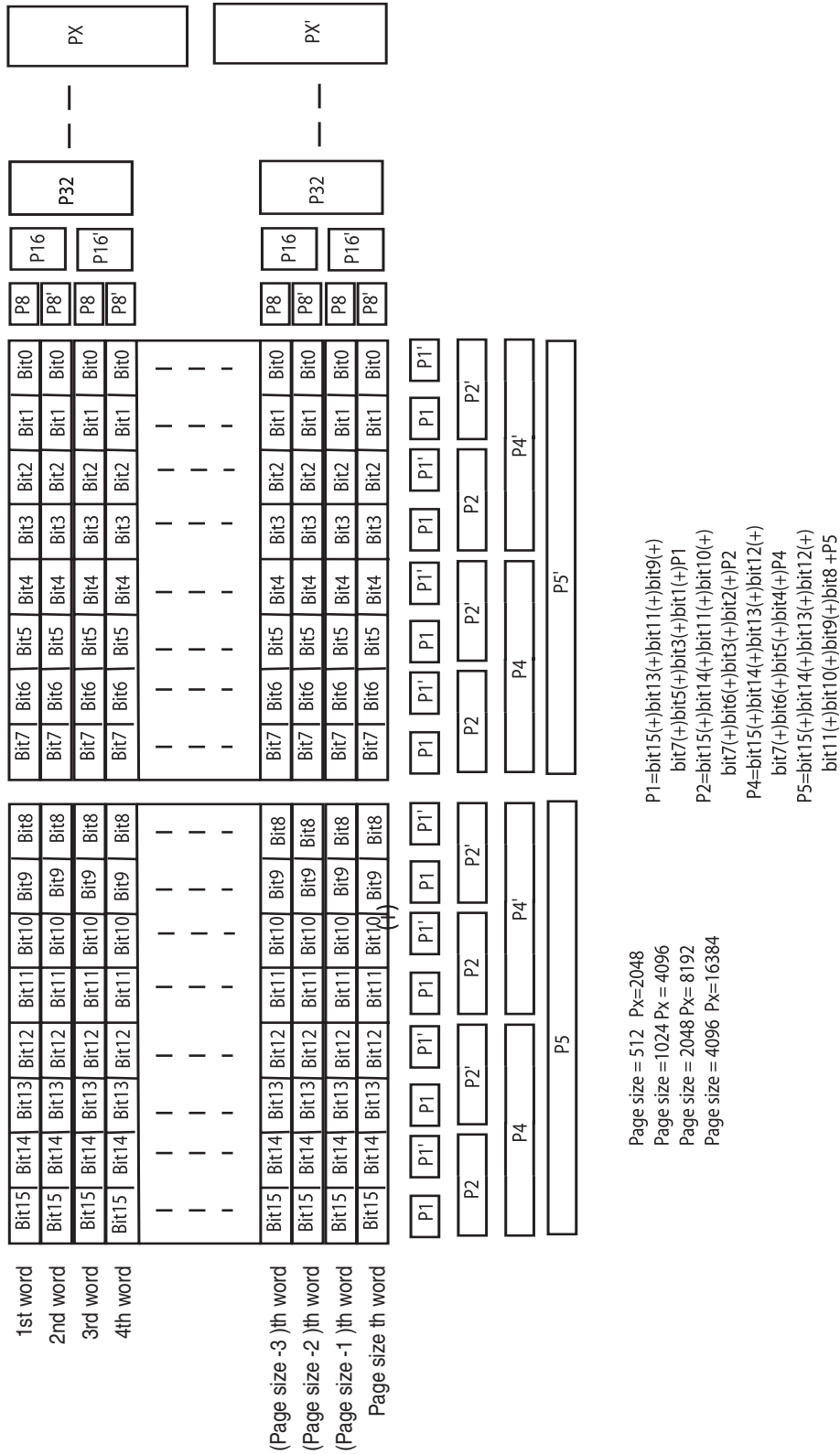
To calculate P8' to PX' and P8 to PX, apply the algorithm that follows.

```

Page size = 2^n

for i =0 to n
begin
for (j = 0 to page_size_byte)
begin
if (j[i] ==1)
P[2i+3]=bit7(+)+bit6(+)+bit5(+)+bit4(+)+bit3(+)+
bit2(+)+bit1(+)+bit0(+)+P[2i+3]
else
P[2i+3]'=bit7(+)+bit6(+)+bit5(+)+bit4(+)+bit3(+)+
bit2(+)+bit1(+)+bit0(+)+P[2i+3]'
end
end
end
    
```

**Figure 25-3. Parity Generation for 512/1024/2048/4096 16-bit Words**



Page size = 512 Px=2048  
 Page size = 1024 Px = 4096  
 Page size = 2048 Px= 8192  
 Page size = 4096 Px=16384

To calculate P8' to PX' and P8 to PX, apply the algorithm that follows.

Page size =  $2^n$

```
for i =0 to n
begin
for (j = 0 to page_size_word)
begin
if(j[i] ==1)
P[2i+3] = bit15(+)bit14(+)bit13(+)bit12(+)
bit11(+)bit10(+)bit9(+)bit8(+)
bit7(+)bit6(+)bit5(+)bit4(+)bit3(+)
bit2(+)bit1(+)bit0(+)P[2n+3]
else
P[2i+3]' =bit15(+)bit14(+)bit13(+)bit12(+)
bit11(+)bit10(+)bit9(+)bit8(+)
bit7(+)bit6(+)bit5(+)bit4(+)bit3(+)
bit2(+)bit1(+)bit0(+)P[2i+3]'
end
end
end
```

## 25.4 Error Corrected Code (ECC) Controller User Interface

**Table 25-1.** ECC Register Mapping

Offset	Register	Register Name	Access	Reset
0x00	ECC Control Register	ECC_CR	Write-only	0x0
0x04	ECC Mode Register	ECC_MR	Read/Write	0x0
0x08	ECC Status Register	ECC_SR	Read-only	0x0
0x0C	ECC Parity Register	ECC_PR	Read-only	0x0
0x10	ECC NParity Register	ECC_NPR	Read-only	0x0



## 25.4.1 ECC Control Register

**Name:** ECC\_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
–	–	–	–	–	–	–	RST

- **RST: RESET Parity**

Provides reset to current ECC by software.

1 = Resets ECC Parity and ECC NParity register.

0 = No effect.

## 25.4.2 ECC Mode Register

**Register Name:** ECC\_MR

**Access Type:** Read/Write

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

- **PAGESIZE: Page Size**

This field defines the page size of the NAND Flash device.

Page Size	Description
00	528 words
01	1056 words
10	2112 words
11	4224 words

A word has a value of 8 bits or 16 bits, depending on the NAND Flash or SmartMedia memory organization.

### 25.4.3 ECC Status Register

**Register Name:** ECC\_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
–	–	–	–	–	MULERR	ECCERR	RECERR

- **RECERR: Recoverable Error**

0 = No Errors Detected.

1 = Errors Detected. If MUL\_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected.

- **ECCERR: ECC Error**

0 = No Errors Detected.

1 = A single bit error occurred in the ECC bytes.

Read both ECC Parity and ECC NParity register, the error occurred at the location which contains a 1 in the least significant 16 bits.

- **MULERR: Multiple Error**

0 = No Multiple Errors Detected.

1 = Multiple Errors Detected.

## 25.4.4 ECC Parity Register

**Register Name:** ECC\_PR

**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
WORDADDR							
7	6	5	4	3	2	1	0
WORDADDR				BITADDR			

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

- **BITADDR**

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **WORDADDR**

During a page read, this value contains the word address (8-bit or 16-bit word depending on the memory plane organization) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

## 25.4.5 ECC NParity Register

**Register Name:** ECC\_NPR

**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
NPARITY							
7	6	5	4	3	2	1	0
NPARITY							

- **NPARITY:**

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.



## 26. Peripheral DMA Controller (PDC)

### 26.1 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 twenty-two channels. The full-duplex peripherals feature twenty-one mono-directional channels used in pairs (transmit only or receive only). The half-duplex peripherals feature one bi-directional channel.

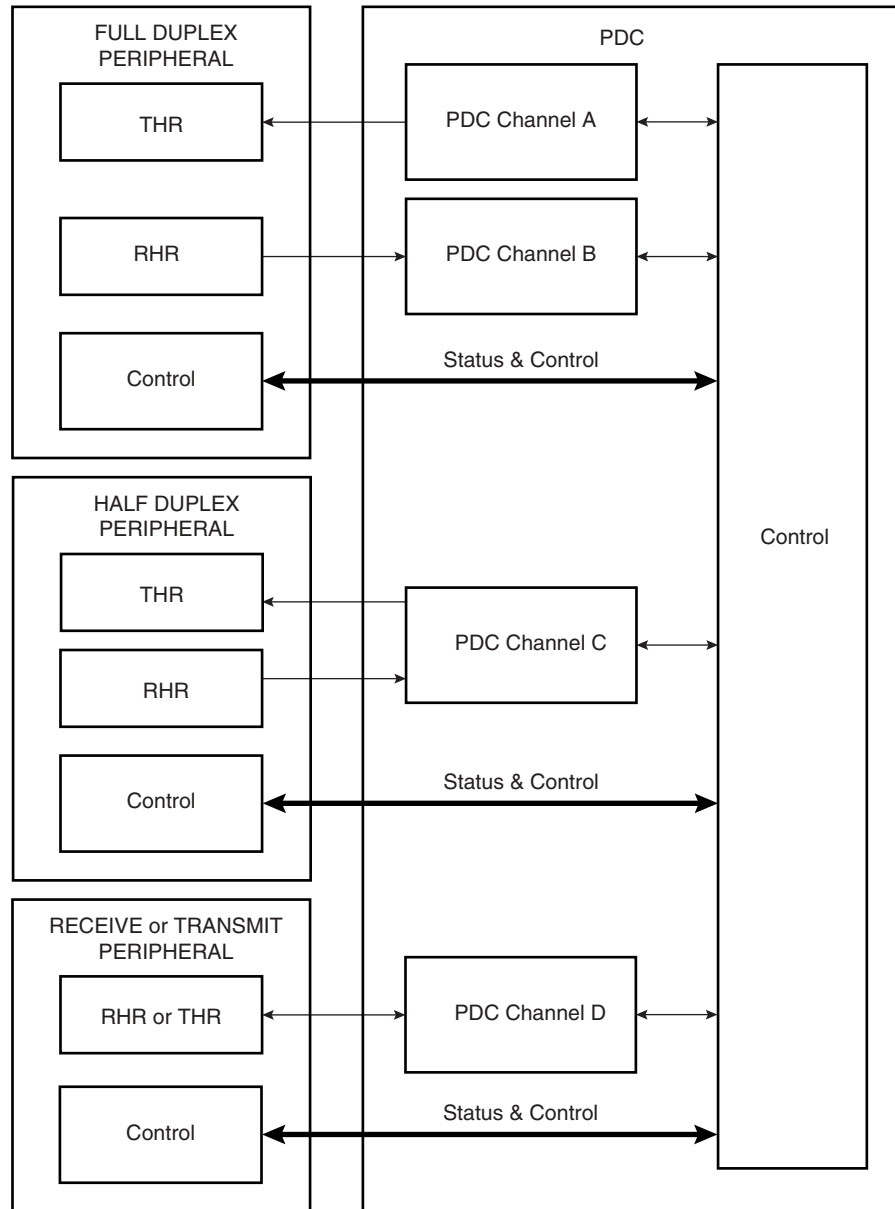
The user interface of each PDC channel is integrated into the user interface of the peripheral it serves. The user interface of mono directional channels (receive only or transmit only), contains two 32-bit memory pointers and two 16-bit counters, one set (pointer, counter) for current transfer and one 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.

## 26.2 Block Diagram

Figure 26-1. Block Diagram



## 26.3 Functional Description

### 26.3.1 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 26.3.3](#) and to the associated peripheral user interface.

### 26.3.2 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.

### 26.3.3 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 than 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.

### 26.3.4 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.

### 26.3.5 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.

#### 26.3.5.1 *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.

#### 26.3.5.2 *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.

#### 26.3.5.3 *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.



## 26.3.5.4 *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.

## 26.4 Peripheral DMA Controller (PDC) User Interface

**Table 26-1.** Memory Map

Offset	Register	Name	Access	Reset State
0x100	Receive Pointer Register	PERIPH <sup>(1)</sup> _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, SSC, SPI, MCI, etc.)

## 26.4.1 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.

## 26.4.2 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

## 26.4.3 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.

### 26.4.4 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

## 26.4.5 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.

### 26.4.6 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.



## 26.4.7 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.

### 26.4.8 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.

## 26.4.9 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.

### 26.4.10 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.

## 27. Clock Generator

### 27.1 Description

The Clock Generator is made up of one PLL, a 12 MHz Main Oscillator, as well as an RC Oscillator and a 32,768 Hz low-power Oscillator.

It provides the following clocks:

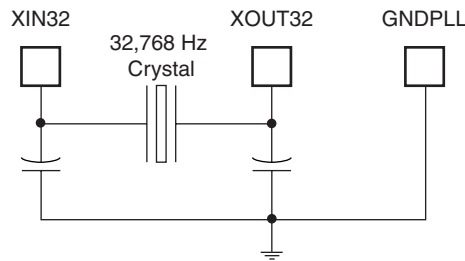
- SLCK, the Slow Clock, which is the only permanent clock within the system
- MAINCK is the output of the 12 MHz Main Oscillator
- PLLCK is the output of the Divider and PLL block

The Clock Generator User Interface is embedded within the Power Management Controller one and is described in [Section 28.8](#). However, the Clock Generator registers are named CKGR\_.

### 27.2 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 27-1](#).

**Figure 27-1.** Typical Slow Clock Crystal Oscillator Connection



### 27.3 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.

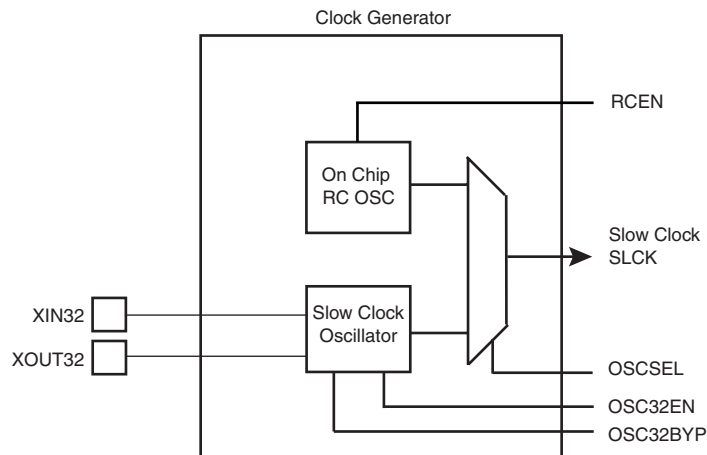
### 27.4 Slow Clock Selection

The AT91SAM9R64/RL64 slow clock can be generated either by an external 32,768 Hz crystal or by the on-chip RC oscillator. The 32,768 Hz crystal oscillator can be bypassed by setting the bit OSC32BYP to accept an external slow clock on XIN32.

The internal RC oscillator and the 32,768 Hz oscillator can be enabled by setting to 1, respectively, RCEN bit and OSC32EN bit in the System Controller user interface. The OSCSEL command selects the slow clock source.

By default the AT91SAM9R64/RL64 slow clock source is the internal slow clock oscillator. System startup time is 4 slow clock periods, typically 125  $\mu$ s.

**Figure 27-2.** Slow Clock Selection



RCEN, OSC32EN, OSCSEL and OSC32BYP bits are located in the Slow Clock Control Register (SCKCR) located at address 0xFFFFFD50 in the backed up part of the System Controller and so are preserved while VDDBU is present.

After a VDDBU power on reset, the default configuration is RCEN=1, OSC32EN=0 and OSCSEL=0, allowing the system to start on the internal RC oscillator.

The programmer controls the slow clock switching by software and so must take precautions during the switching phase.

#### 27.4.1 Switching from Internal RC Oscillator to the 32,768 Hz Crystal

To switch from internal RC oscillator to the 32,768 Hz crystal, the programmer must execute the following sequence:

- Switch the master clock to a source different from slow clock (PLL or Main Oscillator) through the Power Management Controller.
- Enable the 32,768 Hz oscillator by setting the bit OSC32EN to 1.
- Wait 32,768 Hz Startup Time for clock stabilization (software loop).
- Switch from internal RC to 32,768 Hz oscillator by setting the bit OSCSEL to 1.
- Wait 5 slow clock cycles for internal resynchronization.
- Disable the RC oscillator by setting the bit RCEN to 0.

#### 27.4.2 Bypassing the 32,768 Hz Oscillator

Following steps must be added to bypass the 32,768 Hz oscillator:

- An external clock must be connected on XIN32.
- Enable the bypass path OSC32BYP bit set to 1.
- Disable the 32,768 Hz oscillator by setting the bit OSC32EN to 0.

#### 27.4.3 Switching from 32768 Hz Crystal to the Internal RC Oscillator

The same procedure must be followed to switch from a 32,768 Hz crystal to the internal RC oscillator.

- Switch the master clock to a source different from slow clock (PLL or Main Oscillator).
- Enable the internal RC oscillator by setting the bit RCEN to 1.

- Wait internal RC Startup Time for clock stabilization (software loop).
- Switch from 32768 Hz oscillator to internal RC by setting the bit OSCSEL to 0.
- Wait 5 slow clock cycles for internal resynchronization.
- Disable the 32768 Hz oscillator by setting the bit OSC32EN to 0.

## 27.4.4 Slow Clock Configuration Register

**Register Name:** SCKCR

**Address:** 0xFFFFFD50

**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
–	–	–	–	OSCSEL	OSC32BYP	OSC32EN	RCEN

- **RCEN: Internal RC**

0: RC is disabled

1: RC is enabled

- **OSC32EN: 32768 Hz oscillator**

0: 32768Hz oscillator is disabled

1: 32768Hz oscillator is enabled

- **OSC32BYP: 32768Hz oscillator bypass**

0: 32768Hz oscillator is not bypassed

1: 32768Hz oscillator is bypassed, accept an external slow clock on XIN32

- **OSCSEL: Slow clock selector**

0: Slow clock is internal RC

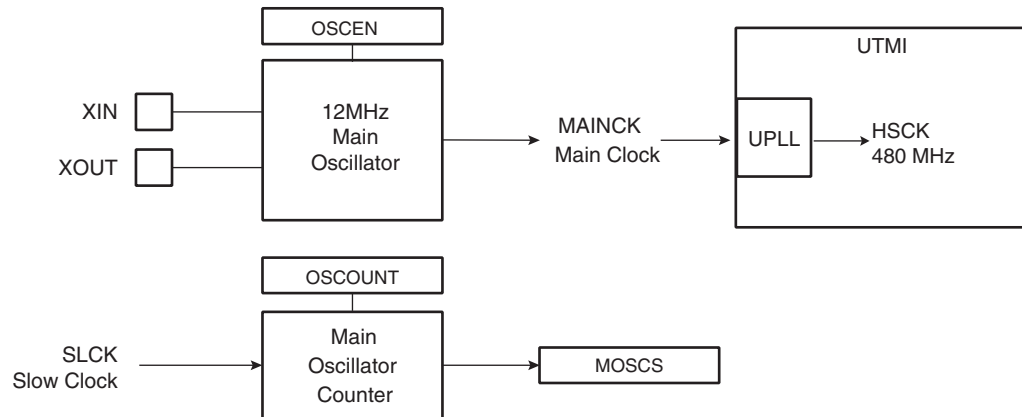
1: Slow clock is 32768 Hz oscillator

## 27.5 Main Oscillator

The Main Oscillator is designed for a 12 MHz fundamental crystal. The 12 MHz is also used to generate the 480 MHz USB High Speed Clock (HSCK) thanks to the UTMI PLL (UPLL).

Figure 27-3 shows the Main Oscillator block diagram.

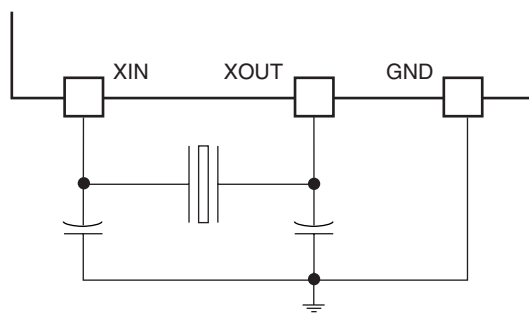
**Figure 27-3.** Main Oscillator Block Diagram



### 27.5.1 Main Oscillator Connections

The typical crystal connection is illustrated in Figure 27-4. For further details on the electrical characteristics of the Main Oscillator, see the section “DC Characteristics” of the product datasheet.

**Figure 27-4.** Typical Crystal Connection



### 27.5.2 Main Oscillator Startup Time

The startup time of the 12 MHz Main Oscillator is given in the section “DC Characteristics” of the product datasheet.

### 27.5.3 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.

## 27.5.4 Main Oscillator Bypass

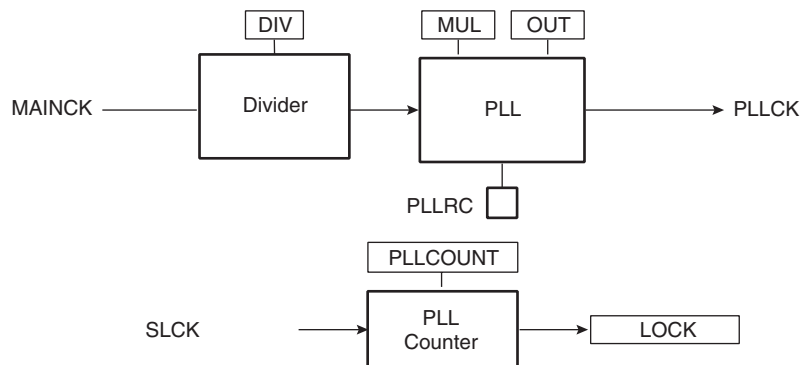
The user can input a clock on the device instead of connecting a crystal. In this case, the user has to provide the external clock signal on the XIN pin. The input characteristics of the XIN pin under these conditions are given in the product electrical characteristics section. The programmer has to 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.

## 27.6 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 27-5 shows the block diagram of the divider and PLL block.

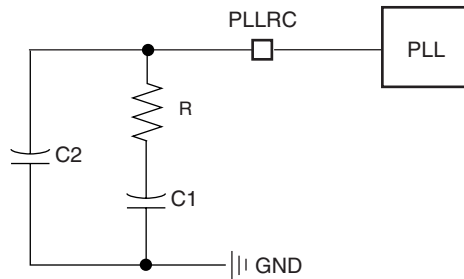
**Figure 27-5.** Divider and PLL Block Diagram



### 27.6.1 PLL Filter

The PLL requires connection to an external second-order filter through the PLLRC pin. Figure 27-6 shows a schematic of these filters.

**Figure 27-6.** PLL Capacitors and Resistors



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.

### 27.6.2 Divider and Phase Lock Loop Programming

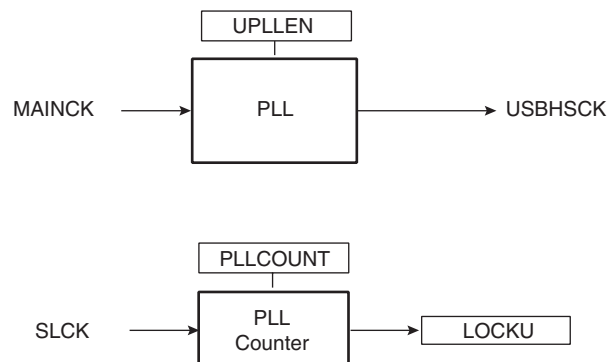
The divider can be set between 1 and 255 in steps of 1. When a divider field (DIV) is set to 0, the output of the corresponding divider and the PLL output is a continuous signal at level 0. On reset, each DIV field is set to 0, thus the corresponding PLL input clock is set to 0.

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 in PMC\_SR is automatically cleared. The values written in the PLLCOUNT field in CKGR\_PLLR 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.

### 27.6.3 UTMI Bias and Phase Lock Loop Programming

The multiplier is hard-wired to 40 to obtain the USB High Speed 480 MHz.



Whenever the PLL is enabled by writing UPLEN in CKGR\_UCKR, the LOCKU bit in PMC\_SR is automatically cleared, the BIAS is enabled by writing BIASEN in CKGR\_UCKR in the same time. The values written in the PLLCOUNT field in CKGR\_UCKR are loaded in the PLL counter. The PLL counter then decrements at the speed of the Slow Clock divided by 8 until it reaches 0. At this time, the LOCKU 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.

## 28. Power Management Controller (PMC)

### 28.1 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), must be 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.
- Programmable Clock Outputs can be selected from the clocks provided by the clock generator and driven on the PCKx pins.

### 28.2 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.

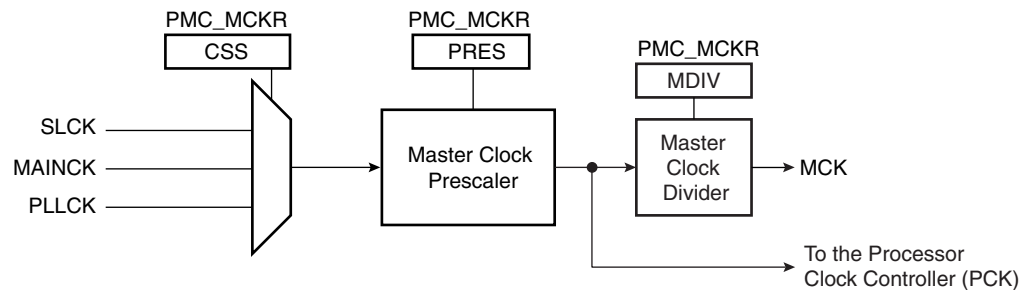
The Master Clock is selected from one of the clocks provided by the Clock Generator. Selecting the Slow Clock provides a Slow Clock signal to the whole device. Selecting the Main Clock saves power consumption of the PLL.

The Master Clock Controller is made up of a clock selector and a prescaler. It also contains a Master Clock divider which allows the processor clock to be faster than the Master Clock.

The Master Clock selection is made by writing the CSS field (Clock Source Selection) in PMC\_MCKR (Master Clock Register). The prescaler supports the division by a power of 2 of the selected clock between 1 and 64. The PRES field in PMC\_MCKR programs the prescaler. The Master Clock divider can be programmed through the MDIV field in PMC\_MCKR.

Each time PMC\_MCKR is written to define a new Master Clock, the MCKRDY bit is cleared in PMC\_SR. It reads 0 until the Master Clock is established. Then, the MCKRDY bit is set and can trigger an interrupt to the processor. This feature is useful when switching from a high-speed clock to a lower one to inform the software when the change is actually done.

**Figure 28-1.** Master Clock Controller



## 28.3 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 purpose) 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 and entering in Wait for Interrupt Mode. The Processor Clock 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.

- Notes:
1. The Processor clock is disabled in PMC\_SCDR register.
  2. The ARM Wait For Interrupt mode is entered with CP15 coprocessor operation.
  3. Refer to the Atmel application note “[Optimizing Power Consumption of AT91SAM9261-based Systems](#)”, lit. no. 6217 for details.

## 28.4 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.

## 28.5 Programmable Clock Output Controller

The PMC controls 2 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 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 bit in PMC\_SR indicates that the Programmable Clock is actually what has been programmed in the Programmable Clock registers.

As the Programmable Clock Controller does not manage with glitch prevention when switching clocks, it is strongly recommended to disable the Programmable Clock before any configuration change and to re-enable it after the change is actually performed.

## 28.6 Programming Sequence

### 1. Enabling the 12MHz 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.

### 2. Setting PLL and divider:

All parameters needed to configure PLL and the divider are located in the CKGR\_PLLR register.

The DIV field is used to control divider itself. A value between 0 and 255 can be programmed. Divider output is divider input divided by DIV parameter. By default DIV parameter is set to 0 which means that divider is turned off.

The OUT field is used to select the PLL output frequency range.

The MUL field is the PLL multiplier factor. This parameter can be programmed between 0 and 2047. If MUL is set to 0, PLL will be turned off, otherwise the PLL output frequency is PLL input frequency multiplied by (MUL + 1).

The PLLCOUNT field specifies the number of slow clock cycles before LOCK bit is set in the PMC\_SR register after CKGR\_PLLR register has been written.

Once the PMC\_PLL register has been written, the user must wait for the LOCK 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 LOCK has been enabled in the PMC\_IER register. All parameters in CKGR\_PLLR can be programmed in a single write operation. If at some stage one of the following parameters, MUL, DIV is modified, LOCK bit will go low to indicate that PLL is not ready yet. When PLL is locked, LOCK will be set again. The user is constrained to wait for LOCK bit to be set before using the PLL output clock.

Code Example:

```
write_register(CKGR_PLLR, 0x00040805)
```

If PLL and divider are enabled, the PLL input clock is the main clock. PLL output clock is PLL input clock multiplied by 5. Once CKGR\_PLLR has been written, LOCK bit will be set after eight slow clock cycles.

### 3. Setting Bias and High Speed PLL (UPLL) for UTMI

The UTMI PLL is enabled by setting the UPLEN field in the CKGR\_UCKR register. The UTMI Bias must be enabled by setting the BIASEN field in the CKGR\_UCKR register in the same time. In some cases it may be advantageous to define a start-up time. This can be achieved by writing a value in the PLLCOUNT field in the CKGR\_UCKR register.

Note: If UTMI Bias is not enabled, the USB Device works only in Full Speed Mode.

Once this register has been correctly configured, the user must wait for LOCKU 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 LOCKU has been enabled in the PMC\_IER register.

### 4. 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.

The MDIV field is used to control the Master Clock prescaler. It is possible to choose between different values (0, 1, 2). The Master Clock output is Processor Clock divided by 1, 2 or 4, depending on the value programmed in MDIV. By default, MDIV is set to 0, which indicates that the Processor Clock is equal to the Master 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, the MCKRDY flag will go low while PLL is unlocked. Once PLL is locked again, LOCK goes high and MCKRDY is set. While PLL is unlocked, the Master Clock selection is automatically changed to Main Clock. For further information, see [Section 28.7.2. “Clock Switching Waveforms” on page 282.](#)

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.

#### 5. 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, 2 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, PLLCK. 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.

## 6. 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, 19 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.

## 28.7 Clock Switching Details

### 28.7.1 Master Clock Switching Timings

[Table 28-1](#) gives 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 28-1.** Clock Switching Timings (Worst Case)

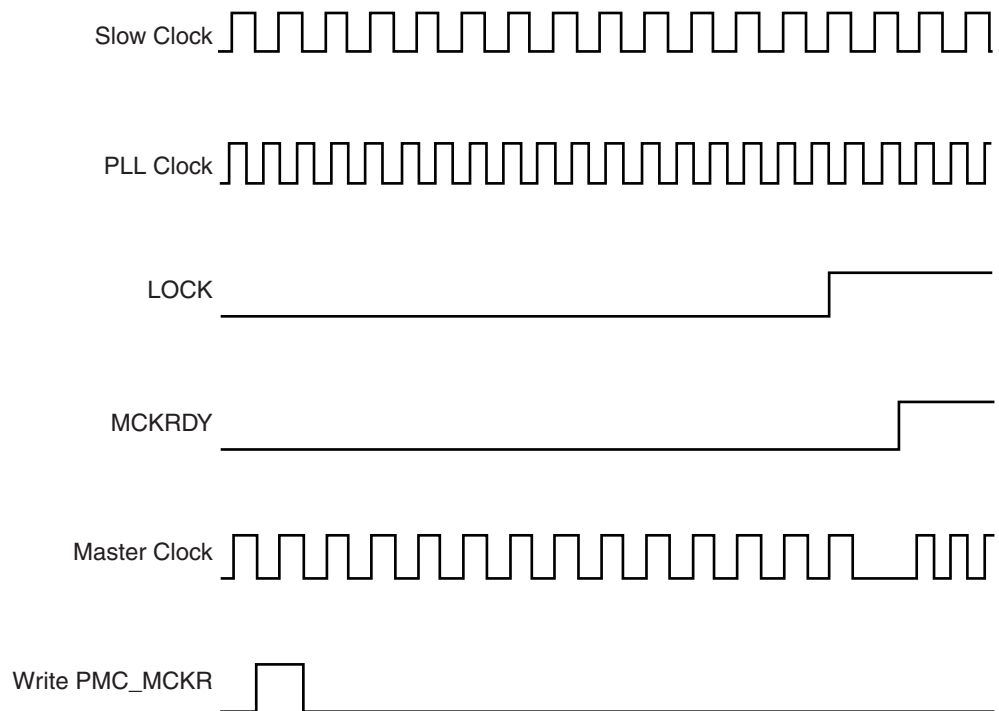
	From	Main Clock	SLCK	PLL Clock
To				

**Table 28-1.** Clock Switching Timings (Worst Case)

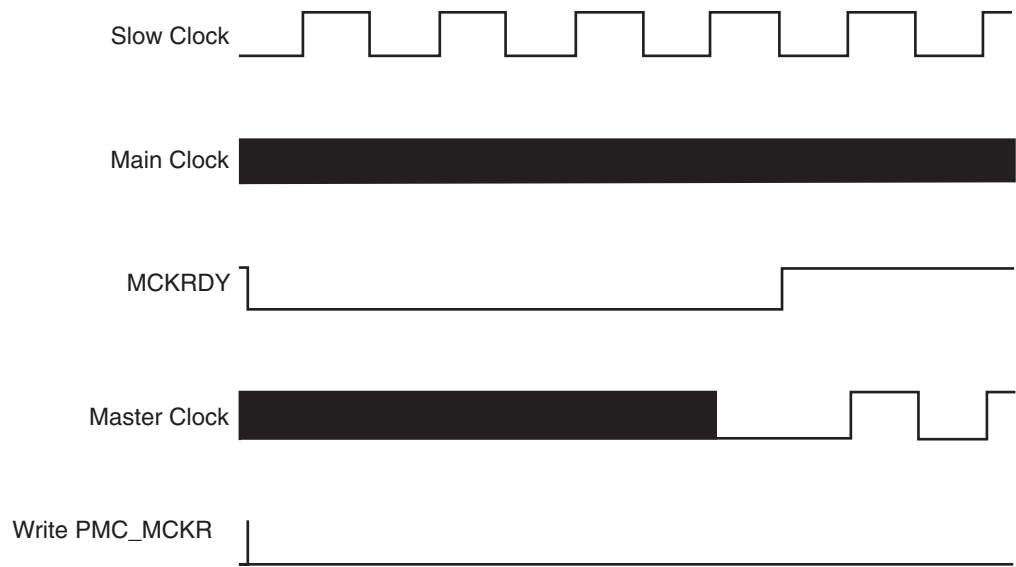
From	Main Clock	SLCK	PLL Clock
Main Clock	–	4 x SLCK + 2.5 x Main Clock	3 x PLL Clock + 4 x SLCK + 1 x Main Clock
SLCK	0.5 x Main Clock + 4.5 x SLCK	–	3 x PLL Clock + 5 x SLCK
PLL Clock	0.5 x Main Clock + 4 x SLCK + PLLCOUNT x SLCK + 2.5 x PLLx Clock	2.5 x PLL Clock + 5 x SLCK + PLLCOUNT x SLCK	2.5 x PLL Clock + 4 x SLCK + PLLCOUNT x SLCK

### 28.7.2 Clock Switching Waveforms

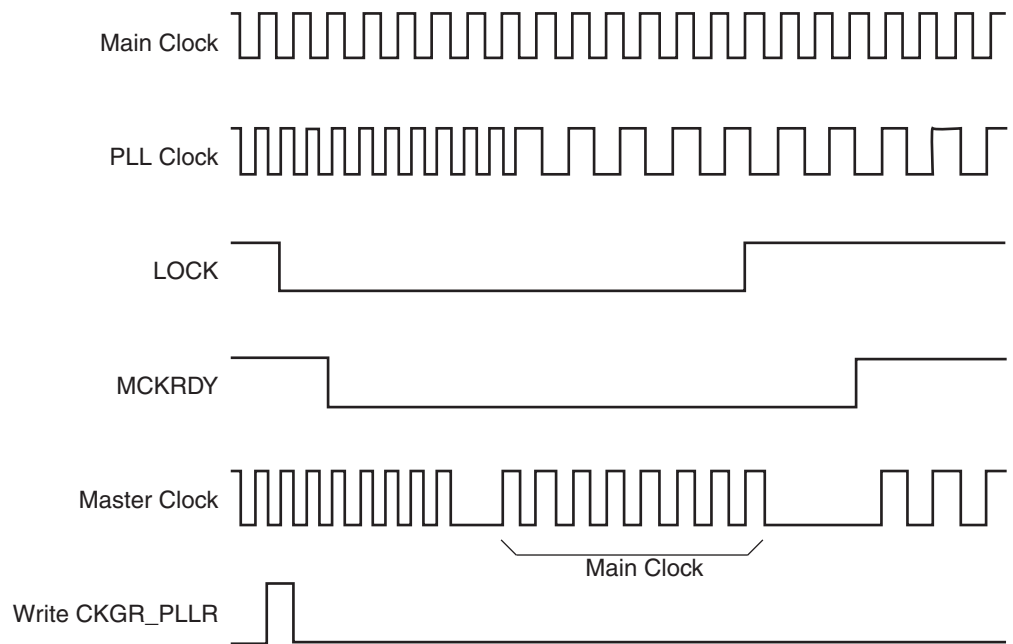
**Figure 28-2.** Switch Master Clock from Slow Clock to PLL Clock



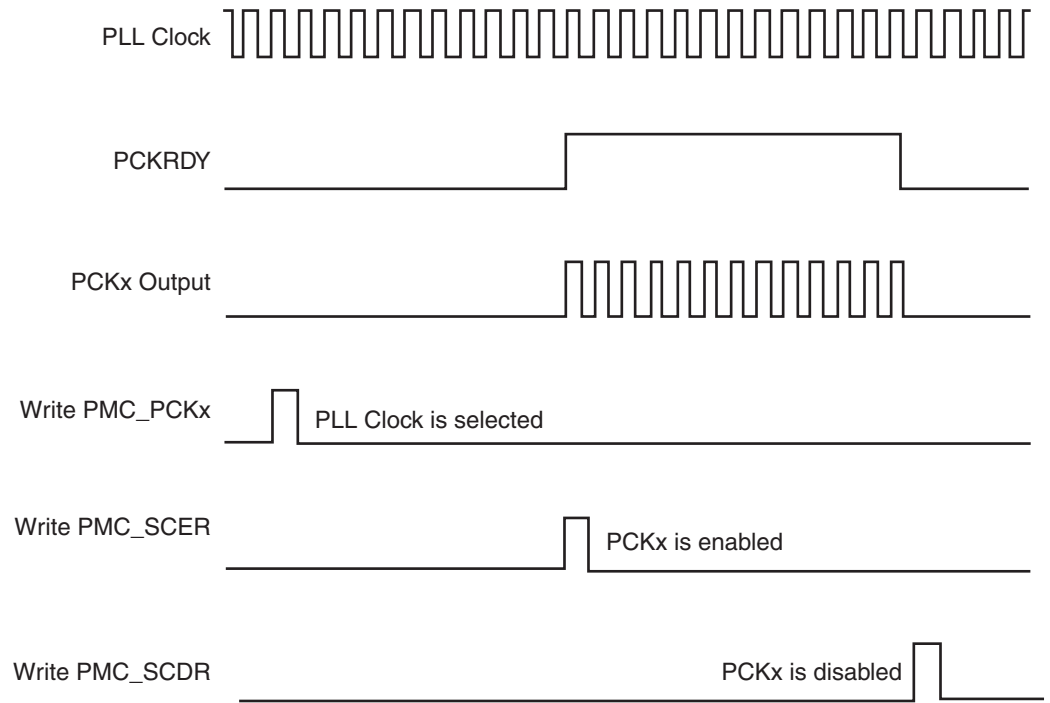
**Figure 28-3.** Switch Master Clock from Main Clock to Slow Clock



**Figure 28-4.** Change PLL Programming



**Figure 28-5.** Programmable Clock Output Programming



## 28.8 Power Management Controller (PMC) User Interface

**Table 28-2.** 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	UTMI Clock Register	CKGR_UCKR	Read-write	0x0
0x0020	Main Oscillator Register	CKGR_MOR	Read-write	0x0
0x0024	Main Clock Frequency Register	CKGR_MCFR	Read-only	0x0
0x0028	PLL Register	CKGR_PLLR	Read-write	0x3F00
0x002C	Reserved	–	–	–
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
0x0048 - 0x005C	Reserved	–	–	–
0x0060	Interrupt Enable Register	PMC_IER	Write-only	--
0x0064	Interrupt Disable Register	PMC_IDR	Write-only	--
0x0068	Status Register	PMC_SR	Read-only	0x08
0x006C	Interrupt Mask Register	PMC_IMR	Read-only	0x0
0x0070 - 0x007C	Reserved	–	–	–

### 28.8.1 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
–	–	–	–	–	–	PCK1	PCK0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **PCKx: Programmable Clock x Output Enable**

0 = No effect.

1 = Enables the corresponding Programmable Clock output.

## 28.8.2 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
–	–	–	–	–	–	PCK1	PCK0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	PCK

- **PCK: Processor Clock Disable**

0 = No effect.

1 = Disables the Processor clock. This is used to enter the processor in Idle Mode.

- **PCKx: Programmable Clock x Output Disable**

0 = No effect.

1 = Disables the corresponding Programmable Clock output.

### 28.8.3 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
–	–	–	–	–	–	PCK1	PCK0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	PCK

- **PCK: Processor Clock Status**

0 = The Processor clock is disabled.

1 = The Processor clock is enabled.

- **PCKx: Programmable Clock x Output Status**

0 = The corresponding Programmable Clock output is disabled.

1 = The corresponding Programmable Clock output is enabled.



## 28.8.4 PMC Peripheral Clock Enable Register

**Register Name:** PMC\_PCER

**Access Type:** Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	-	-

- **PIDx: Peripheral Clock x Enable**

0 = No effect.

1 = Enables the corresponding peripheral clock.

Note: PID2 to PID31 refer to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet.

Note: Programming the control bits of the Peripheral ID that are not implemented has no effect on the behavior of the PMC.

## 28.8.5 PMC Peripheral Clock Disable Register

**Register Name:** PMC\_PCDR

**Access Type:** Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	-	-

- **PIDx: Peripheral Clock x Disable**

0 = No effect.

1 = Disables the corresponding peripheral clock.

Note: PID2 to PID31 refer to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet.



## 28.8.6 PMC Peripheral Clock Status Register

**Register Name:** PMC\_PCSR

**Access Type:** Read-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	–	–

- **PIDx: Peripheral Clock x Status**

0 = The corresponding peripheral clock is disabled.

1 = The corresponding peripheral clock is enabled.

Note: PID2 to PID31 refer to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet.

## 28.8.7 PMC UTMI Clock Configuration Register

**Register Name:** CKGR\_UCKR

**Access Type:** Read-write

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

- **UPLLEN: UTMI PLL Enable**

0 = The UTMI PLL is disabled.

1 = The UTMI PLL is enabled.

When UPLLEN is set, the LOCKU flag is set once the UTMI PLL startup time is achieved.

- **PLLCOUNT: UTMI PLL Start-up Time**

Specifies the number of Slow Clock cycles multiplied by 8 for the UTMI PLL start-up time.

- **BIASEN: UTMI BIAS Enable**

0 = The UTMI BIAS is disabled. The USB Device works only in FS Mode.

1 = The UTMI BIAS is enabled. The USB Device works in HS Mode.

- **BIASCOUNT: UTMI BIAS Start-up Time**

Specifies the number of Slow Clock cycles for the UTMI BIAS start-up time.

### 28.8.8 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.

## 28.8.9 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.

### 28.8.10 PMC Clock Generator PLL Register

**Register Name:** CKGR\_PLLR

**Access Type:** Read-write

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

Possible limitations on PLL input frequencies and multiplier factors should be checked before using the PMC.

- **DIV: Divider**

DIV	Divider Selected
0	Divider output is 0
1	Divider is bypassed
2 - 255	Divider output is the selected clock divided by DIV.

- **PLLCOUNT: PLL Counter**

Specifies the number of slow clock cycles before the LOCK bit is set in PMC\_SR after CKGR\_PLLR is written.

- **OUT: PLL 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.

- **MUL: PLL Multiplier**

0 = The PLL is deactivated.

1 up to 2047 = The PLL Clock frequency is the PLL input frequency multiplied by MUL+ 1.

## 28.8.11 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
–	–	–	–	–	–	MDIV	
7	6	5	4	3	2	1	0
–	–	–	PRES			CSS	

- **CSS: Master Clock Selection**

CSS		Clock Source Selection	
0	0	0	Slow Clock is selected
0	1	1	Main Clock is selected
1	0	0	PLL Clock is selected.
1	1	1	Reserved

- **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

- **MDIV: Master Clock Division**

MDIV		Master Clock Division
0	0	Master Clock is Processor Clock.
0	1	Master Clock is Processor Clock divided by 2.
1	0	Master Clock is Processor Clock divided by 4.
1	1	Reserved.



### 28.8.12 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	0	Slow Clock is selected
0	1	1	Main Clock is selected
1	0	0	PLL Clock is selected.
1	1	1	Reserved

• **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



## 28.8.13 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
–	–	–	–	–	–	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	LOCKU	–	–	MCKRDY	–	LOCK	MOSCS

- **MOSCS: Main Oscillator Status Interrupt Enable**
- **LOCK: PLL Lock Interrupt Enable**
- **MCKRDY: Master Clock Ready Interrupt Enable**
- **LOCKU: UTMI PLL Lock Interrupt Enable**
- **PCKRDYx: Programmable Clock Ready x Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

### 28.8.14 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
–	–	–	–	–	–	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	LOCKU	–	–	MCKRDY	–	LOCK	MOSCS

- **MOSCS: Main Oscillator Status Interrupt Disable**
- **LOCK: PLL Lock Interrupt Disable**
- **MCKRDY: Master Clock Ready Interrupt Disable**
- **LOCKU: UTMI PLL Lock Interrupt Disable**
- **PCKRDYx: Programmable Clock Ready x Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

## 28.8.15 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
–	–	–	–	–	–	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
OSC_SEL	LOCKU	–	–	MCKRDY	–	LOCK	MOSCS

- **MOSCS: MOSCS Flag Status**

0 = Main oscillator is not stabilized.

1 = Main oscillator is stabilized.

- **LOCK: PLL Lock Status**

0 = PLL is not locked

1 = PLL is locked.

- **MCKRDY: Master Clock Status**

0 = Master Clock is not ready.

1 = Master Clock is ready.

- **LOCKU: UPLL Lock Status**

0 = UPLL is not locked

1 = UPLL is locked.

- **OSC\_SEL: Slow Clock Oscillator**

0 = Internal slow clock RC oscillator is selected.

1 = External slow clock 32 kHz oscillator is selected.

- **PCKRDYx: Programmable Clock Ready Status**

0 = Programmable Clock x is not ready.

1 = Programmable Clock x is ready.

### 28.8.16 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
–	–	–	–	–	–	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	LOCKU	–	–	MCKRDY	–	LOCK	MOSCS

- **MOSCS: Main Oscillator Status Interrupt Mask**
- **LOCK: PLL Lock Interrupt Mask**
- **MCKRDY: Master Clock Ready Interrupt Mask**
- **LOCKU: UTMI PLL Lock Interrupt Mask**
- **PCKRDYx: Programmable Clock Ready x Interrupt Mask**

0 = The corresponding interrupt is enabled.

1 = The corresponding interrupt is disabled.

## 29. Advanced Interrupt Controller (AIC)

### 29.1 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.

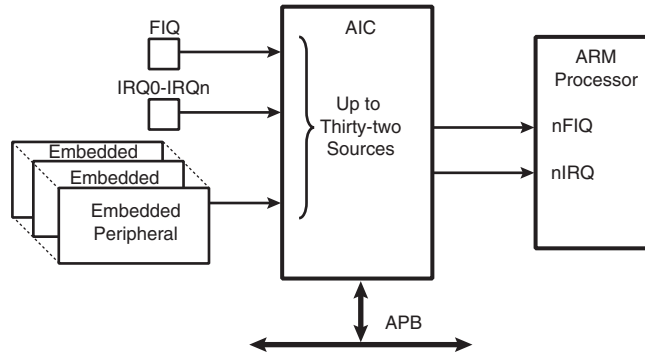
The 8-level Priority Controller allows the user to define the priority for each interrupt source, thus permitting higher priority interrupts to be serviced even if a lower priority interrupt is being treated.

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.

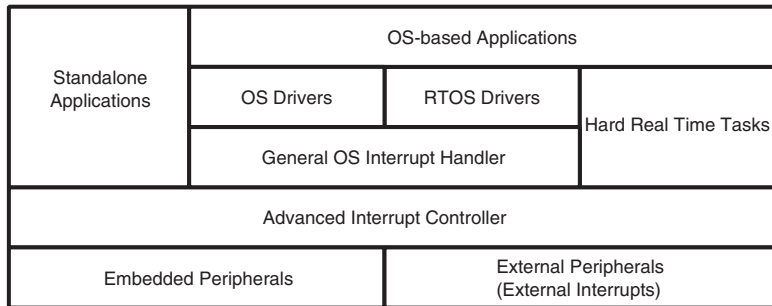
## 29.2 Block Diagram

Figure 29-1. Block Diagram



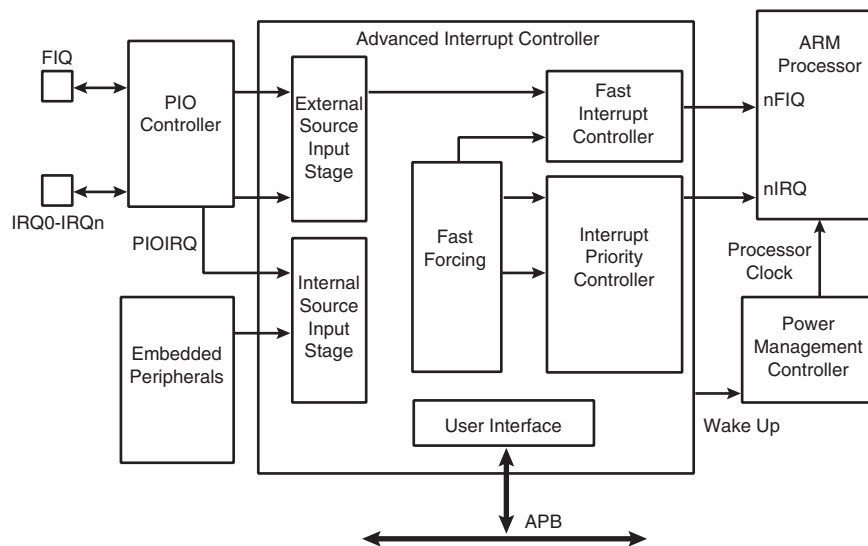
## 29.3 Application Block Diagram

Figure 29-2. Description of the Application Block



## 29.4 AIC Detailed Block Diagram

Figure 29-3. AIC Detailed Block Diagram



## 29.5 I/O Line Description

**Table 29-1.** I/O Line Description

Pin Name	Pin Description	Type
FIQ	Fast Interrupt	Input
IRQ0 - IRQn	Interrupt 0 - Interrupt n	Input

## 29.6 Product Dependencies

### 29.6.1 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.

### 29.6.2 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.

### 29.6.3 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.

## 29.7 Functional Description

### 29.7.1 Interrupt Source Control

#### 29.7.1.1 *Interrupt Source Mode*

The Advanced Interrupt Controller independently programs each interrupt source. The SRC\_TYPE field of the corresponding AIC\_SMR (Source Mode Register) selects the interrupt condition of each source.

The internal interrupt sources wired on the interrupt outputs of the embedded peripherals can be programmed either in level-sensitive mode or in edge-triggered mode. The active level of the internal interrupts is not important for the user.

The external interrupt sources can be programmed either in high level-sensitive or low level-sensitive modes, or in positive edge-triggered or negative edge-triggered modes.

#### 29.7.1.2 *Interrupt Source Enabling*

Each interrupt source, including the FIQ in source 0, can be enabled or disabled by using the command registers; AIC\_I ECR (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.

#### 29.7.1.3 *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 307.) 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 311.)

The automatic clear of the interrupt source 0 is performed when AIC\_FVR is read.

#### 29.7.1.4 *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.

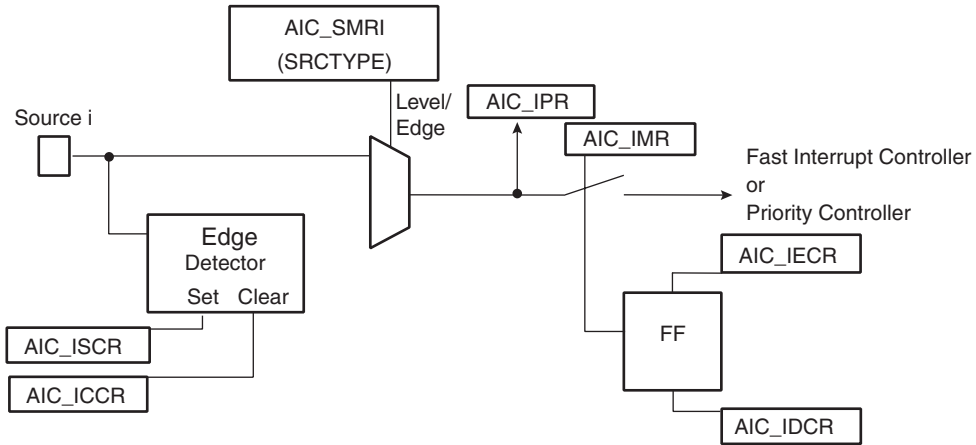
The AIC\_ISR register reads the number of the current interrupt (see “Priority Controller” on page 307) and the register AIC\_CISR gives an image of the signals nIRQ and nFIQ driven on the processor.

Each status referred to above can be used to optimize the interrupt handling of the systems.



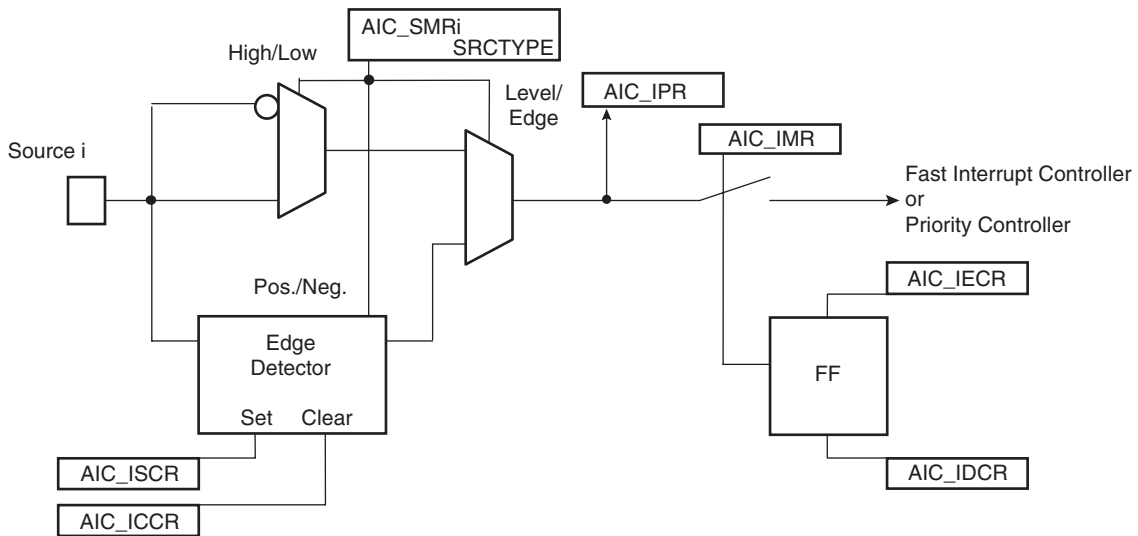
## 29.7.1.5 Internal Interrupt Source Input Stage

**Figure 29-4.** Internal Interrupt Source Input Stage



## 29.7.1.6 External Interrupt Source Input Stage

**Figure 29-5.** External Interrupt Source Input Stage



## 29.7.2 Interrupt Latencies

Global interrupt latencies depend on several parameters, including:

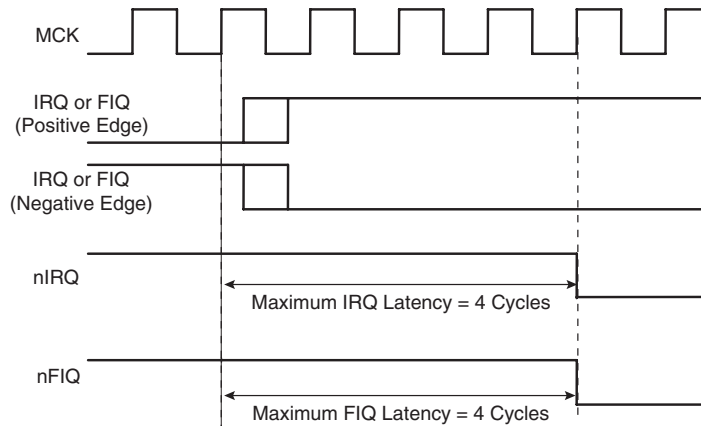
- The time the software masks the interrupts.
- Occurrence, either at the processor level or at the AIC level.
- The execution time of the instruction in progress when the interrupt occurs.
- The treatment of higher priority interrupts and the resynchronization of the hardware signals.

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.

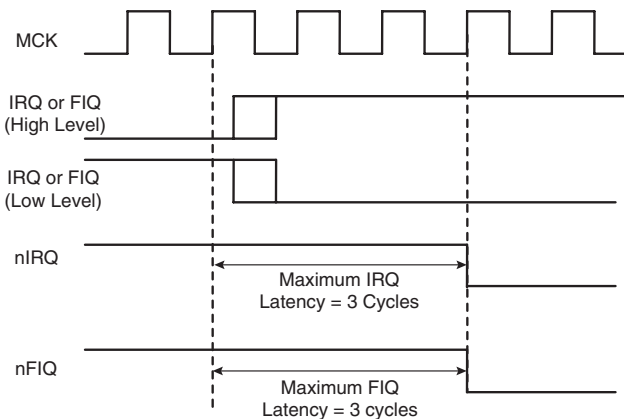
### 29.7.2.1 External Interrupt Edge Triggered Source

**Figure 29-6.** External Interrupt Edge Triggered Source



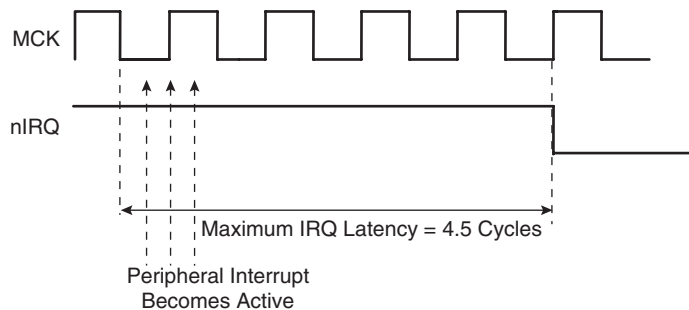
### 29.7.2.2 External Interrupt Level Sensitive Source

**Figure 29-7.** External Interrupt Level Sensitive Source



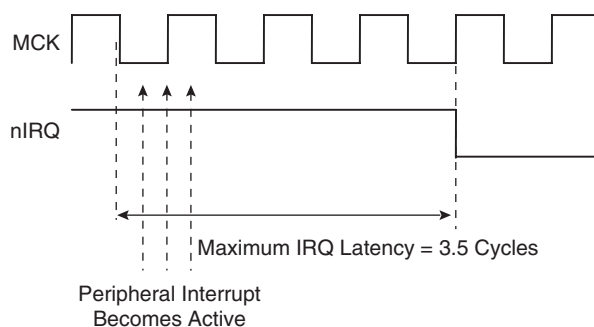
## 29.7.2.3 Internal Interrupt Edge Triggered Source

**Figure 29-8.** Internal Interrupt Edge Triggered Source



## 29.7.2.4 Internal Interrupt Level Sensitive Source

**Figure 29-9.** Internal Interrupt Level Sensitive Source



## 29.7.3 Normal Interrupt

### 29.7.3.1 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.**

### 29.7.3.2 *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.

### 29.7.3.3 *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.

### 29.7.3.4 *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 are used and restoring them at the end. During this phase, an interrupt of higher priority than the current level restarts 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 exe-

cutted 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).

## 29.7.4 Fast Interrupt

### 29.7.4.1 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.

### 29.7.4.2 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.

### 29.7.4.3 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.

### 29.7.4.4 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.

#### 29.7.4.5 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).

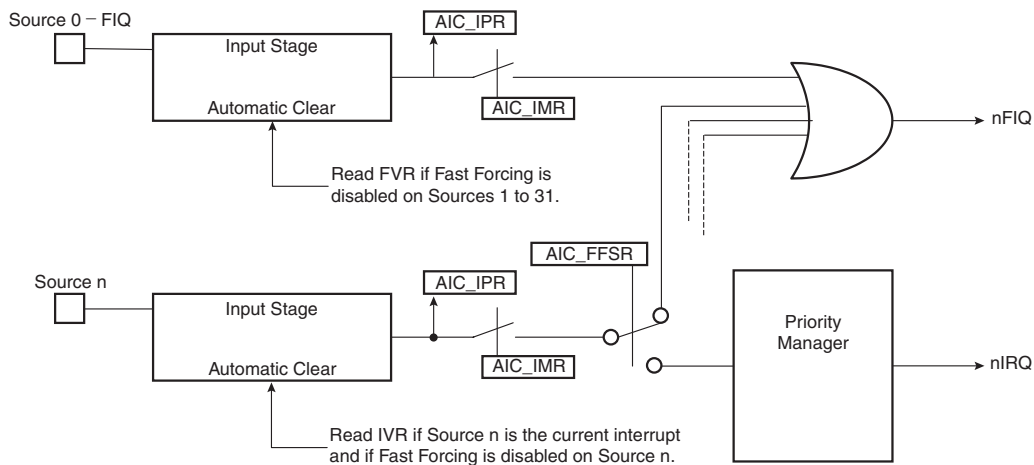
The FIQ Vector Register (AIC\_FVR) reads the contents of the Source Vector Register 0 (AIC\_SVR0), whatever the source of the fast interrupt may be. The read of the FVR does not clear the Source 0 when the fast forcing feature is used and the interrupt source should be cleared by writing to the Interrupt 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 29-10. Fast Forcing**



### 29.7.5 Protect Mode

The Protect Mode permits reading the Interrupt Vector Register without performing the associated automatic operations. This is necessary when working with a debug system. When a debugger, working either with a Debug Monitor or the ARM processor's ICE, stops the applications and updates the opened windows, it might read the AIC User Interface and thus the IVR. This has undesirable consequences:

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

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

This is avoided by using the Protect Mode. Writing DBGM in AIC\_DCR (Debug Control Register) at 0x1 enables the Protect Mode.

When the Protect Mode is enabled, the AIC performs interrupt stacking only when a write access is performed on the AIC\_IVR. Therefore, the Interrupt Service Routines must write (arbitrary data) to the AIC\_IVR just after reading it. The new context of the AIC, including the value of the



Interrupt Status Register (AIC\_ISR), is updated with the current interrupt only when AIC\_IVR is written.

An AIC\_IVR read on its own (e.g., by a debugger), modifies neither the AIC context nor the AIC\_ISR. Extra AIC\_IVR reads perform the same operations. However, it is recommended to not stop the processor between the read and the write of AIC\_IVR of the interrupt service routine to make sure the debugger does not modify the AIC context.

To summarize, in normal operating mode, the read of AIC\_IVR performs the following operations within the AIC:

1. Calculates active interrupt (higher than current or spurious).
2. Determines and returns the vector of the active interrupt.
3. Memorizes the interrupt.
4. Pushes the current priority level onto the internal stack.
5. Acknowledges the interrupt.

However, while the Protect Mode is activated, only operations 1 to 3 are performed when AIC\_IVR is read. Operations 4 and 5 are only performed by the AIC when AIC\_IVR is written.

Software that has been written and debugged using the Protect Mode runs correctly in Normal Mode without modification. However, in Normal Mode the AIC\_IVR write has no effect and can be removed to optimize the code.

## 29.7.6 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.

## 29.7.7 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.

## 29.8 Advanced Interrupt Controller (AIC) User Interface

### 29.8.1 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  $\pm$  4-Kbyte offset.

### 29.8.2 Register Mapping

**Table 29-2.** 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 <sup>(2)</sup>	AIC_IPR	Read-only	0x0 <sup>(1)</sup>
0x110	Interrupt Mask Register <sup>(2)</sup>	AIC_IMR	Read-only	0x0
0x114	Core Interrupt Status Register	AIC_CISR	Read-only	0x0
0x118	Reserved	---	---	---
0x11C	Reserved	---	---	---
0x120	Interrupt Enable Command Register <sup>(2)</sup>	AIC_IECR	Write-only	---
0x124	Interrupt Disable Command Register <sup>(2)</sup>	AIC_IDCR	Write-only	---
0x128	Interrupt Clear Command Register <sup>(2)</sup>	AIC_ICCR	Write-only	---
0x12C	Interrupt Set Command Register <sup>(2)</sup>	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 <sup>(2)</sup>	AIC_FFER	Write-only	---
0x144	Fast Forcing Disable Register <sup>(2)</sup>	AIC_FFDR	Write-only	---
0x148	Fast Forcing Status Register <sup>(2)</sup>	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 section "Peripheral Identifiers" of the product datasheet.

## 29.8.3 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.

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

### 29.8.4 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.

### 29.8.5 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.

## 29.8.6 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.

## 29.8.7 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.



### 29.8.8 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.

### 29.8.9 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.

## 29.8.10 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.

## 29.8.11 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.

### 29.8.12 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.

### 29.8.13 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.



## 29.8.14 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.

## 29.8.15 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.

### 29.8.16 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.

### 29.8.17 AIC Debug Control Register

**Register Name:** AIC\_DEBUG

**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.

## 29.8.18 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.

## 29.8.19 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.

## 29.8.20 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.

## 30. Debug Unit (DBGU)

### 30.1 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.

## 30.2 Block Diagram

Figure 30-1. Debug Unit Functional Block Diagram

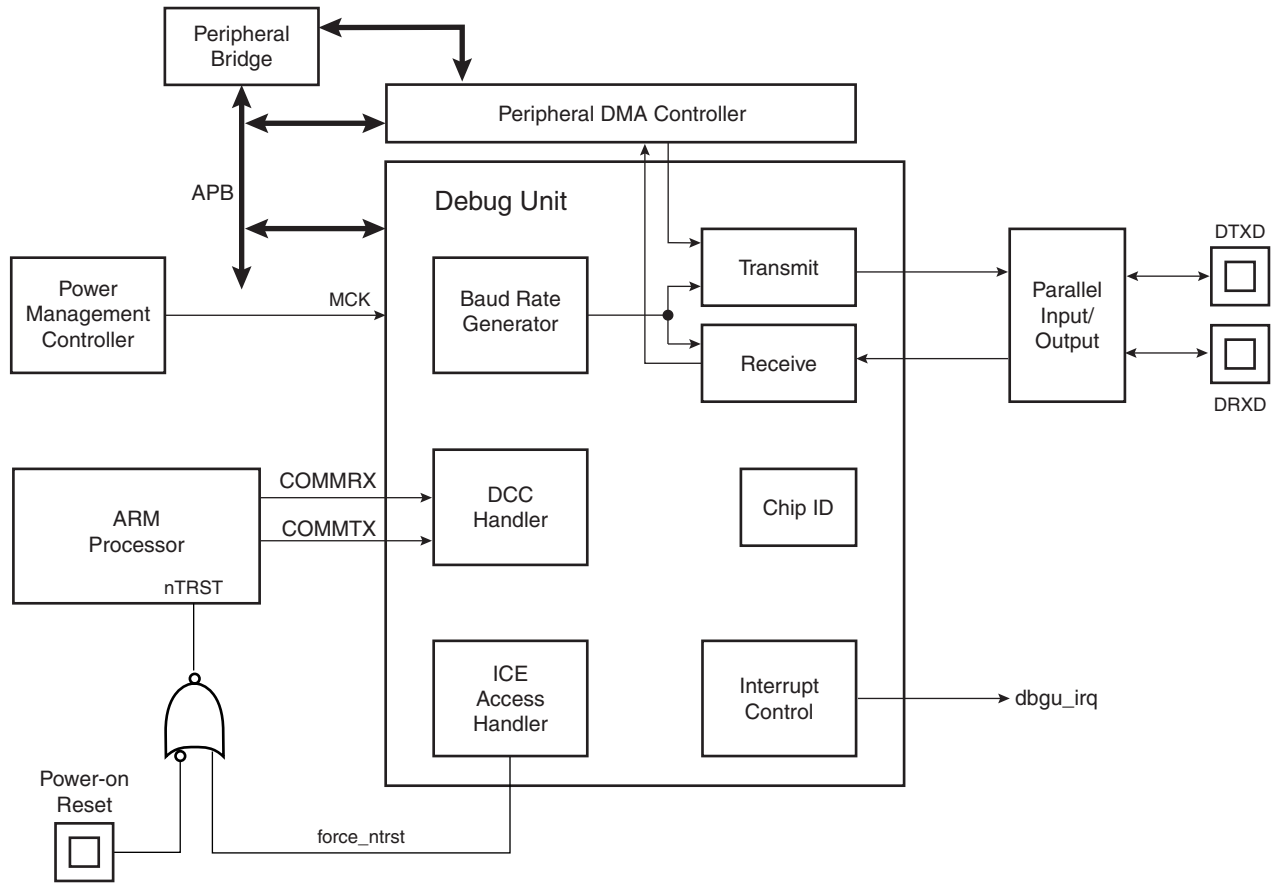
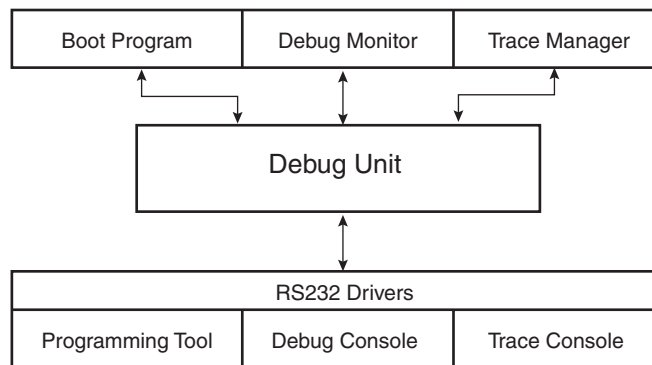


Table 30-1. Debug Unit Pin Description

Pin Name	Description	Type
DRXD	Debug Receive Data	Input
DTXD	Debug Transmit Data	Output

Figure 30-2. Debug Unit Application Example



## 30.3 Product Dependencies

### 30.3.1 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.

### 30.3.2 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.

### 30.3.3 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 30-1](#). This sharing requires the programmer to determine the source of the interrupt when the source 1 is triggered.

## 30.4 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.

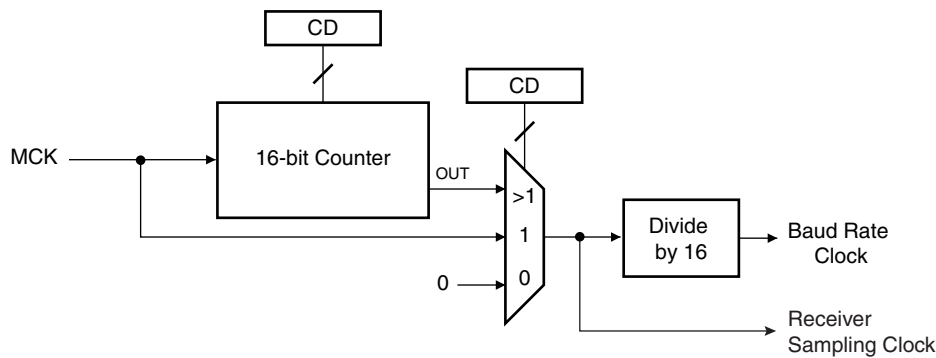
### 30.4.1 Baud Rate Generator

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 30-3.** Baud Rate Generator



## 30.4.2 Receiver

### 30.4.2.1 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.

### 30.4.2.2 Start Detection and Data Sampling

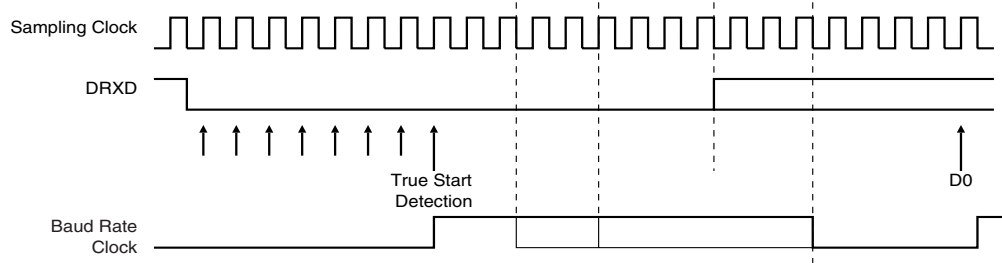
The Debug Unit only supports asynchronous operations, and this affects only its receiver. The Debug Unit receiver detects the start of a received character by sampling the `DRXD` signal until it detects a valid start bit. A low level (space) on `DRXD` is interpreted as a valid start bit if it is detected 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.

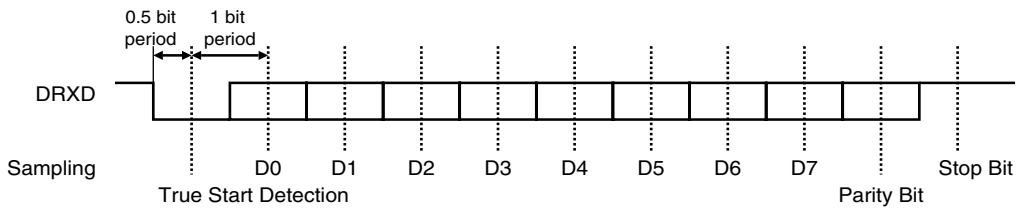


**Figure 30-4.** Start Bit Detection



**Figure 30-5.** Character Reception

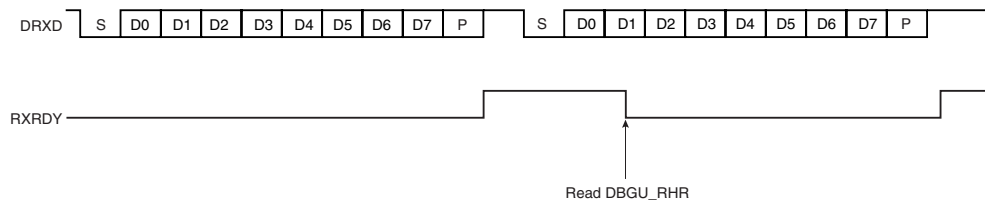
Example: 8-bit, parity enabled 1 stop



### 30.4.2.3 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.

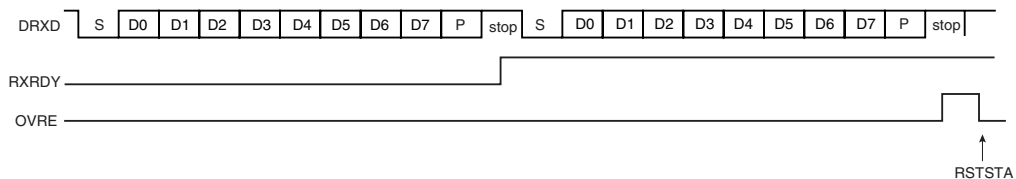
**Figure 30-6.** Receiver Ready



### 30.4.2.4 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 30-7.** Receiver Overrun

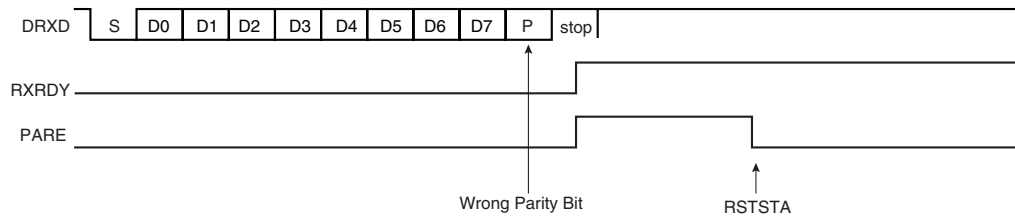


### 30.4.2.5 Parity Error

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.

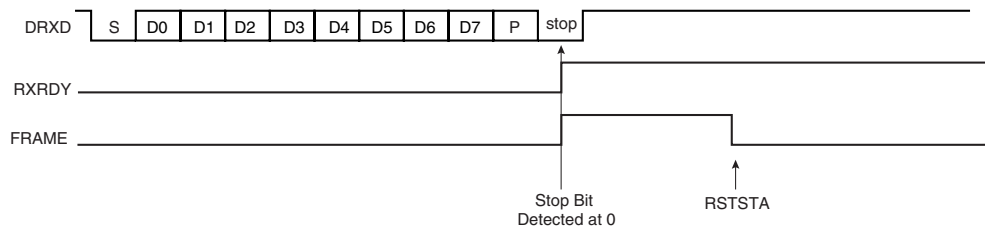
**Figure 30-8.** Parity Error



### 30.4.2.6 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 30-9.** Receiver Framing Error



## 30.4.3 Transmitter

### 30.4.3.1 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.

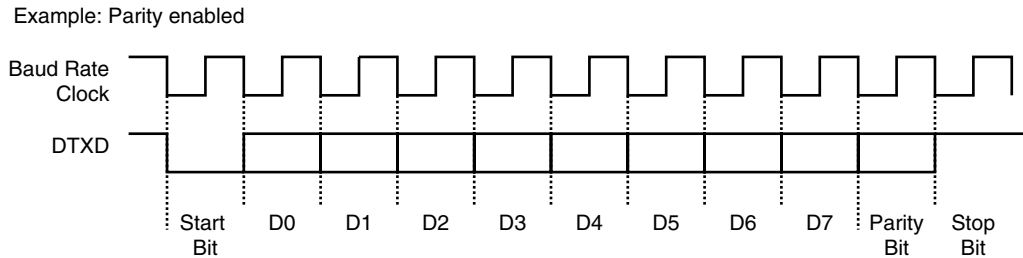
The programmer can also put the transmitter in its reset state by writing the DBGU\_CR with the bit RSTTX at 1. This immediately stops the transmitter, whether or not it is processing characters.

### 30.4.3.2 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 30-10.** Character Transmission

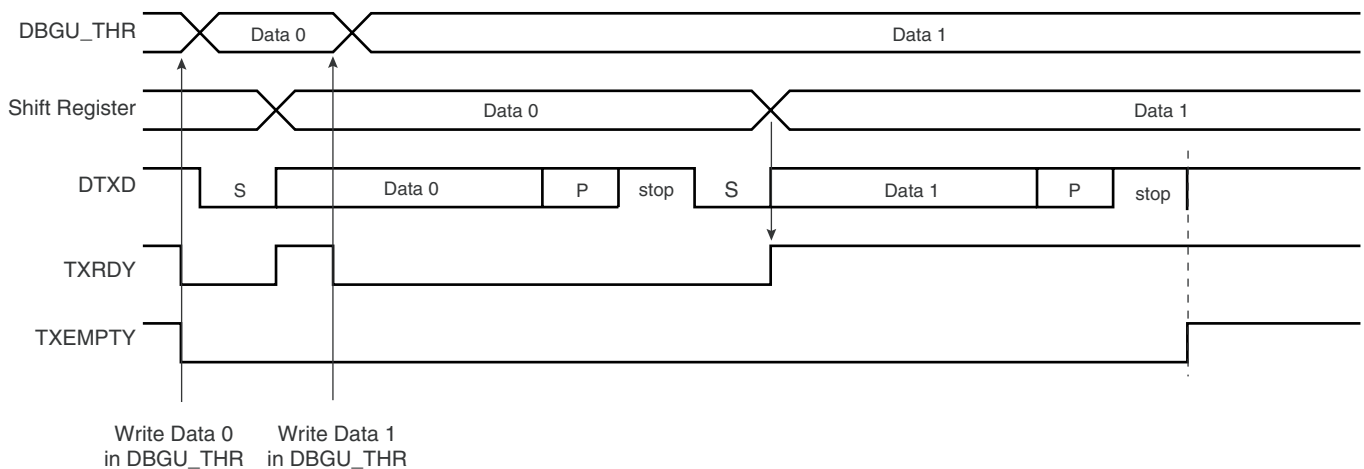


### 30.4.3.3 Transmitter Control

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 30-11.** Transmitter Control



### 30.4.4 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.

### 30.4.5 Test Modes

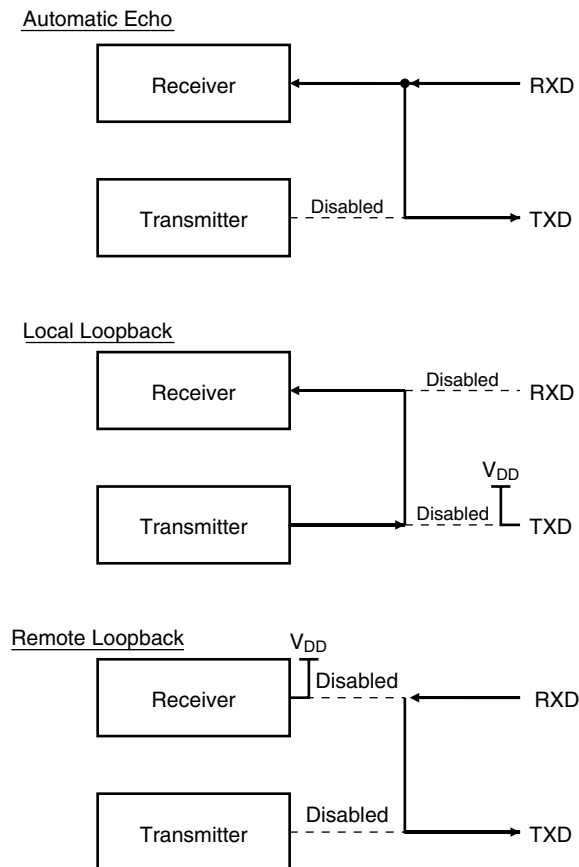
The Debug Unit supports three tests modes. These modes of operation are programmed by using the field CHMODE (Channel Mode) in the mode register DBGU\_MR.

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 30-12.** Test Modes



### 30.4.6 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.

## 30.4.7 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.

## 30.4.8 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.

## 30.5 Debug Unit User Interface

**Table 30-2.** 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	–	–	–

## 30.5.1 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.

## 30.5.2 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



## 30.5.3 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.

## 30.5.4 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.

## 30.5.5 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.

## 30.5.6 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.

## 30.5.7 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.

## 30.5.8 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.

## 30.5.9 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	$MCK / (CD \times 16)$

## 30.5.10 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	ARM946E-S™
0	1	0	ARM7TDMI®
1	0	0	ARM920T™
1	0	1	ARM926EJ-S

- **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		Architecture
Hex	Bin	
0x19	0001 1001	AT91SAM9xx Series
0x29	0010 1001	AT91SAM9XExx Series
0x34	0011 0100	AT91x34 Series
0x37	0011 0111	CAP7 Series
0x39	0011 1001	CAP9 Series
0x3B	0011 1011	CAP11 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.

## 30.5.11 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.

## 30.5.12 Debug Unit Force NTRST Register

**Name:** DBGU\_FNR

**Access Type:** Read/Write

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

- **FNTRST: Force NTRST**

0 = NTRST of the ARM processor's TAP controller is driven by the power\_on\_reset signal.

1 = NTRST of the ARM processor's TAP controller is held low.



## 31. Parallel Input/Output Controller (PIO)

### 31.1 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 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.

## 31.2 Block Diagram

Figure 31-1. Block Diagram

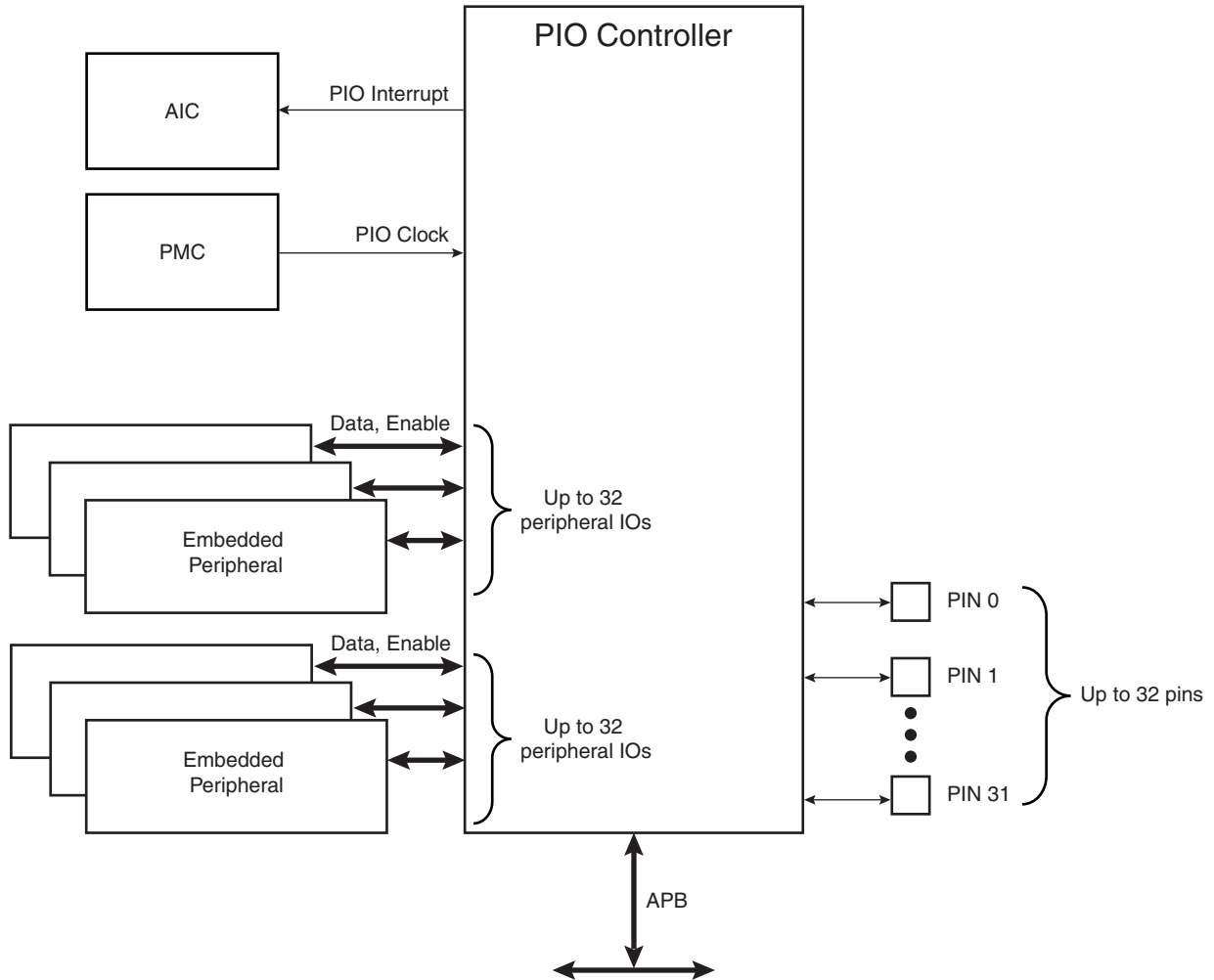
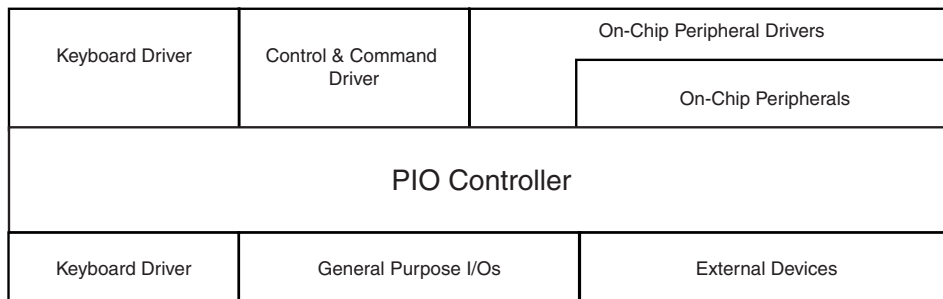


Figure 31-2. Application Block Diagram



## 31.3 Product Dependencies

### 31.3.1 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.

### 31.3.2 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.

### 31.3.3 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.

### 31.3.4 Interrupt Generation

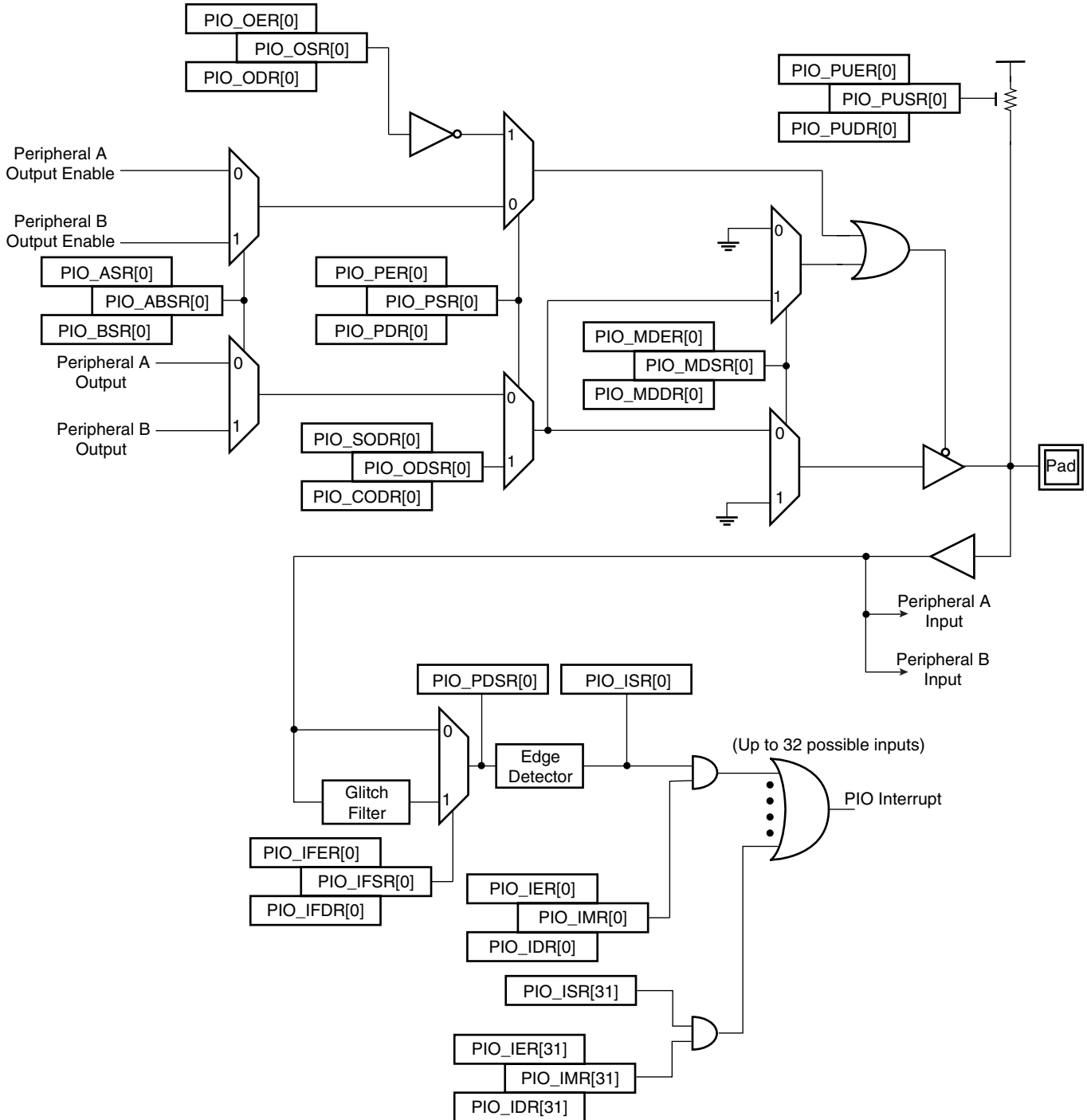
For interrupt handling, the PIO Controllers are considered as user peripherals. This means that the PIO Controller interrupt lines are connected among the interrupt sources 2 to 31. Refer to the PIO Controller peripheral identifier in the product description to identify the interrupt sources dedicated to the PIO Controllers.

The PIO Controller interrupt can be generated only if the PIO Controller clock is enabled.

### 31.4 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 31-3](#). In this description each signal shown represents but one of up to 32 possible indexes.

**Figure 31-3.** I/O Line Control Logic





## 31.4.1 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 Register). Writing in these registers results in setting or clearing the corresponding bit in PIO\_PUSR (Pull-up Status Register). Reading a 1 in PIO\_PUSR means the pull-up is disabled and reading a 0 means the pull-up is enabled.

Control of the pull-up resistor is possible regardless of the configuration of the I/O line.

After reset, all of the pull-ups are enabled, i.e. PIO\_PUSR resets at the value 0x0.

## 31.4.2 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.

## 31.4.3 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.

## 31.4.4 Output Control

When the I/O line is assigned to a peripheral function, i.e. the corresponding bit in PIO\_PSR is at 0, the drive of the I/O line is controlled by the peripheral. Peripheral A or B, depending on the value in PIO\_ABSR, determines whether the pin is driven or not.

When the I/O line is controlled by the PIO controller, the pin can be configured to be driven. This is done by writing PIO\_OER (Output Enable Register) and PIO\_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.

#### 31.4.5 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.

#### 31.4.6 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.

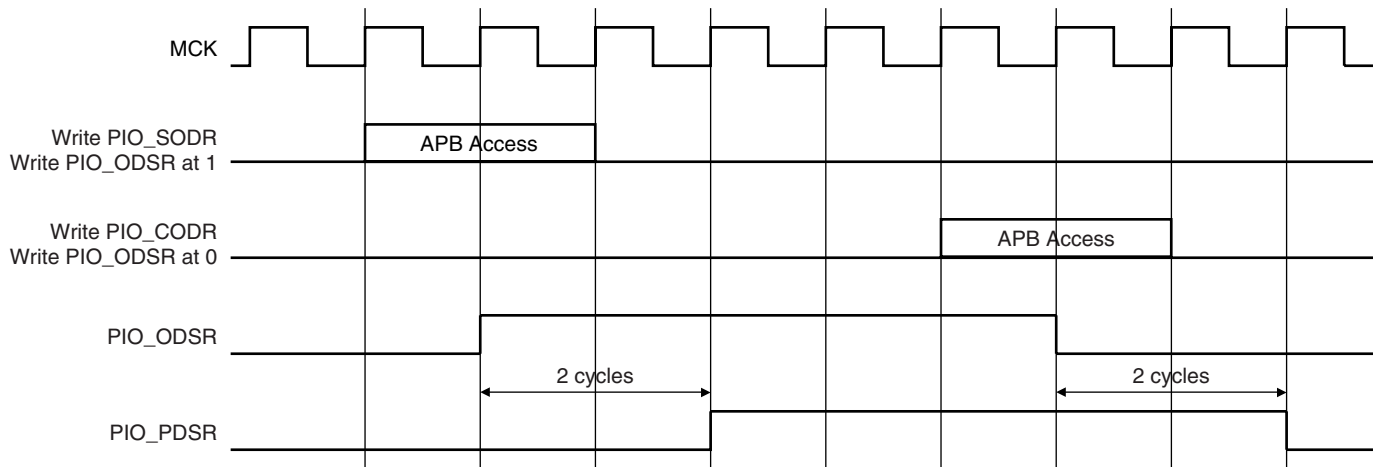
The Multi Drive feature is controlled by PIO\_MDER (Multi-driver Enable Register) and PIO\_MDDR (Multi-driver Disable Register). The Multi Drive can be selected whether the I/O line is controlled by the PIO controller or assigned to a peripheral function. PIO\_MDSR (Multi-driver Status Register) indicates 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.

#### 31.4.7 Output Line Timings

Figure 31-4 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 31-4 also shows when the feedback in PIO\_PDSR is available.

**Figure 31-4. Output Line Timings**



### 31.4.8 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.

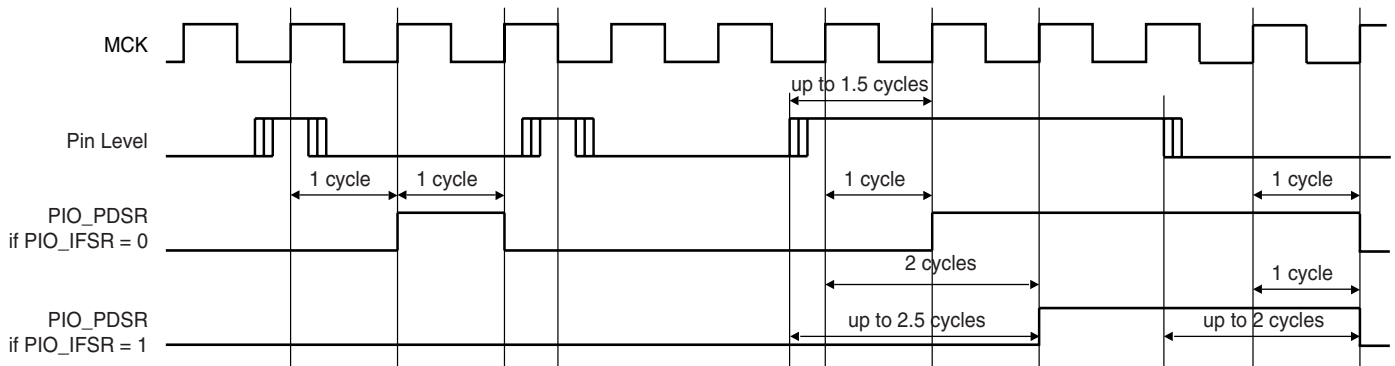
### 31.4.9 Input Glitch Filtering

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 31-5](#).

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 31-5.** Input Glitch Filter Timing



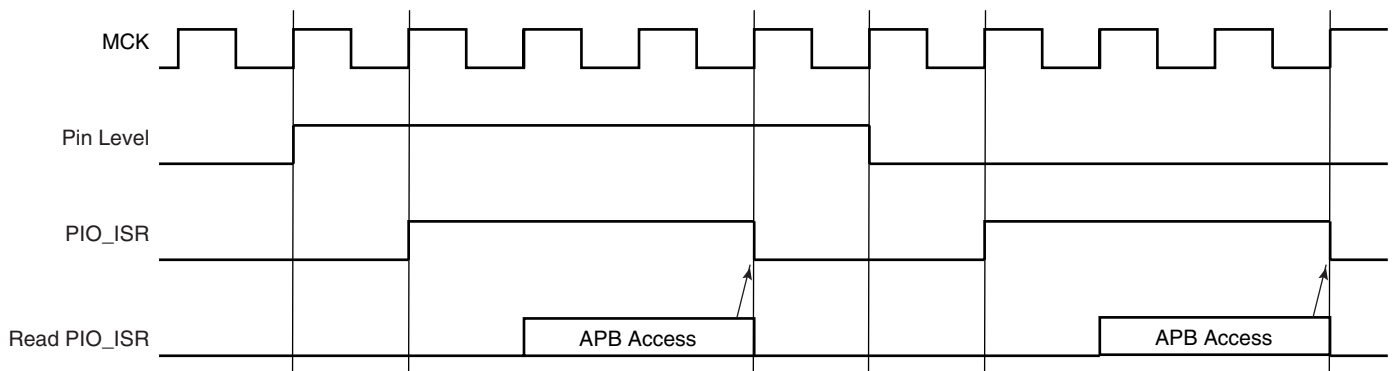
### 31.4.10 Input Change Interrupt

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 31-6.** Input Change Interrupt Timings



## 31.5 I/O Lines Programming Example

The programming example as shown in [Table 31-1](#) 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 31-1.** 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

## 31.6 Parallel Input/Output (PIO) Controller 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 31-2.** 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 <sup>(2)</sup> 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 <sup>(4)</sup>	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 31-2.** Register Mapping (Continued)

Offset	Register	Name	Access	Reset Value
0x0070	Peripheral A Select Register <sup>(5)</sup>	PIO_ASR	Write-only	–
0x0074	Peripheral B Select Register <sup>(5)</sup>	PIO_BSR	Write-only	–
0x0078	AB Status Register <sup>(5)</sup>	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.



### 31.6.1 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).

### 31.6.2 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).



### 31.6.3 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).

### 31.6.4 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.

### 31.6.5 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.

### 31.6.6 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.

## 31.6.7 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.

## 31.6.8 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.



### 31.6.9 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 Filter Status**

0 = The input glitch filter is disabled on the I/O line.

1 = The input glitch filter is enabled on the I/O line.

### 31.6.10 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.

## 31.6.11 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.

## 31.6.12 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.

### 31.6.13 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.

### 31.6.14 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.

## 31.6.15 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.

## 31.6.16 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.

### 31.6.17 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.

### 31.6.18 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.



## 31.6.19 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.

## 31.6.20 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.



### 31.6.21 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.

### 31.6.22 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.

## 31.6.23 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.

## 31.6.24 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.

### 31.6.25 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.

### 31.6.26 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.

## 31.6.27 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.

## 31.6.28 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.

### 31.6.29 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.

## 32. Serial Peripheral Interface (SPI)

### 32.1 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.

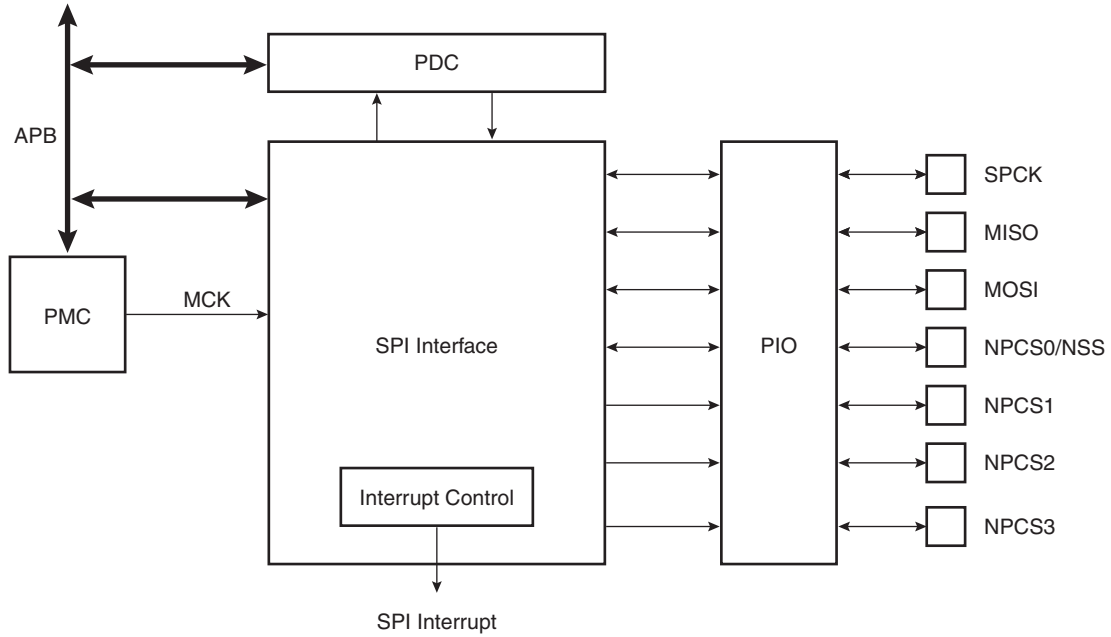
A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

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.

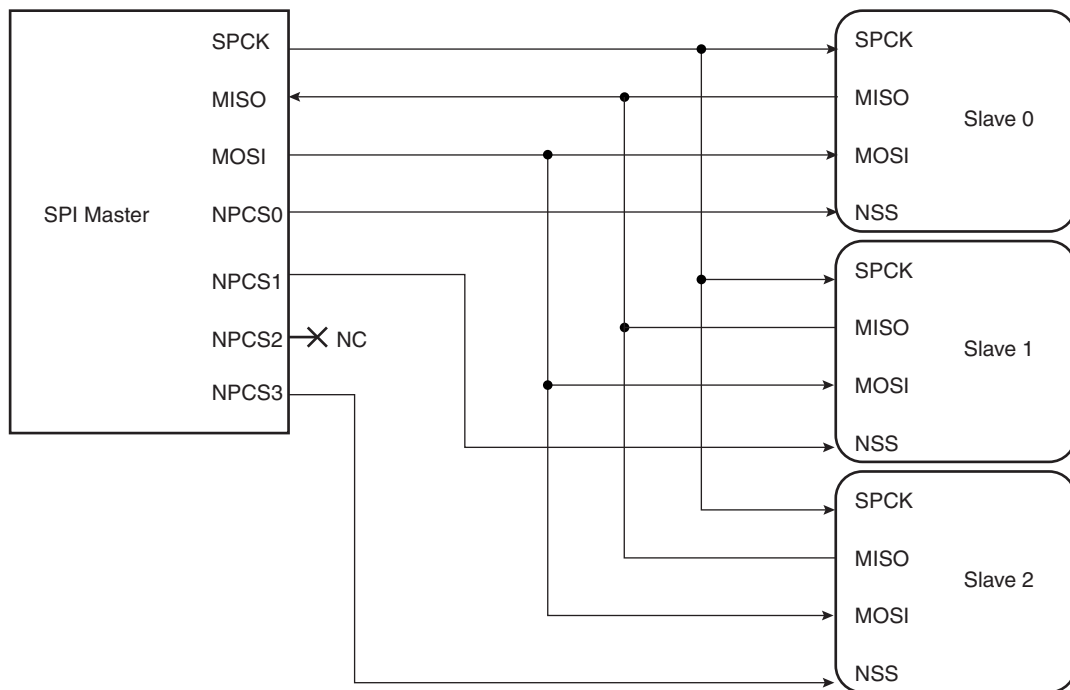
## 32.2 Block Diagram

Figure 32-1. Block Diagram



## 32.3 Application Block Diagram

Figure 32-2. Application Block Diagram: Single Master/Multiple Slave Implementation





## 32.4 Signal Description

**Table 32-1.** 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

## 32.5 Product Dependencies

### 32.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the SPI pins to their peripheral functions.

### 32.5.2 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.

### 32.5.3 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.

## 32.6 Functional Description

### 32.6.1 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.

### 32.6.2 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.

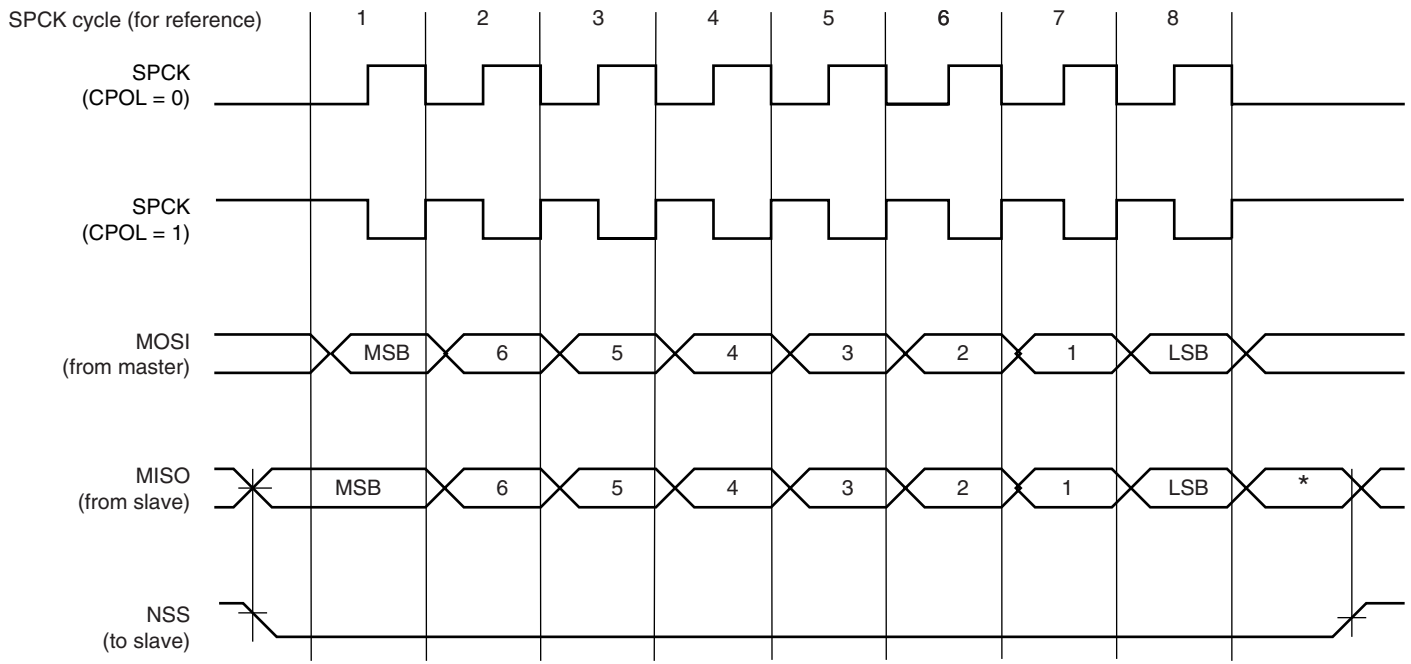
Table 32-2 shows the four modes and corresponding parameter settings.

**Table 32-2.** SPI Bus Protocol Mode

SPI Mode	CPOL	NCPHA
0	0	1
1	0	0
2	1	1
3	1	0

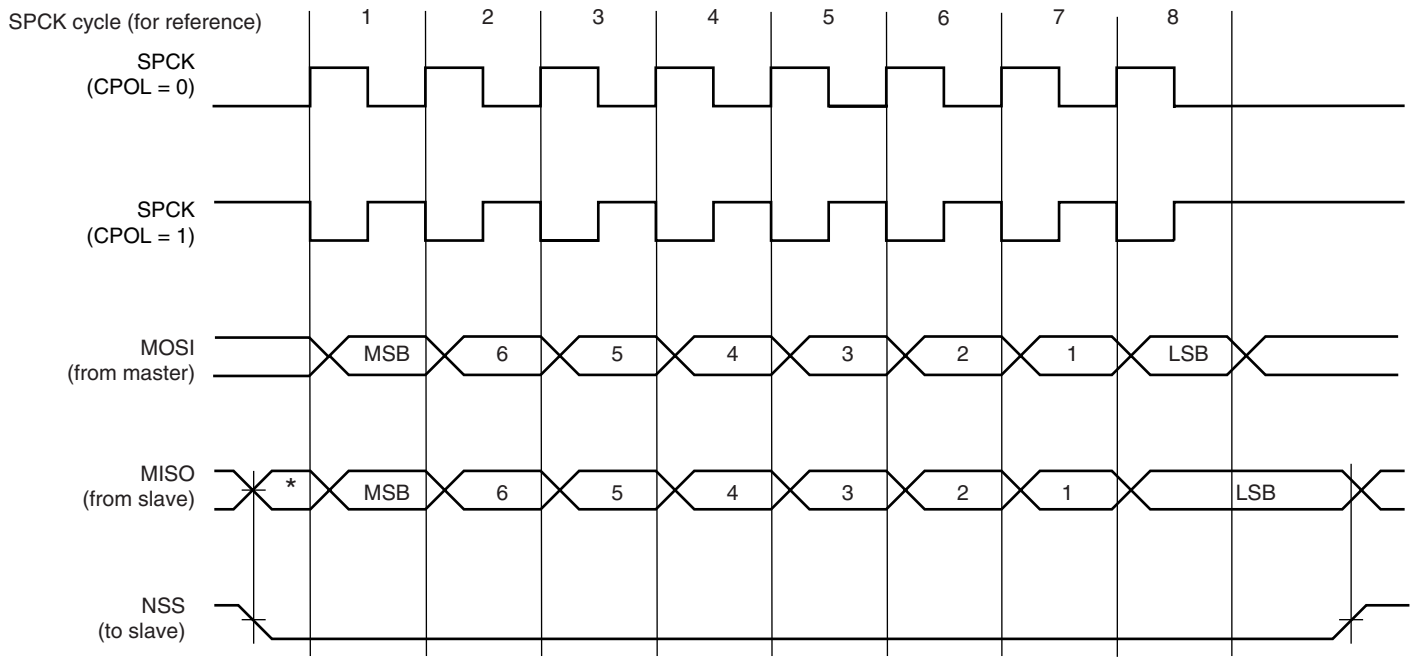
Figure 32-3 and Figure 32-4 show examples of data transfers.

**Figure 32-3.** SPI Transfer Format (NCPHA = 1, 8 bits per transfer)



\* Not defined, but normally MSB of previous character received.

**Figure 32-4.** SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



\* Not defined but normally LSB of previous character transmitted.

### 32.6.3 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 writing 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.

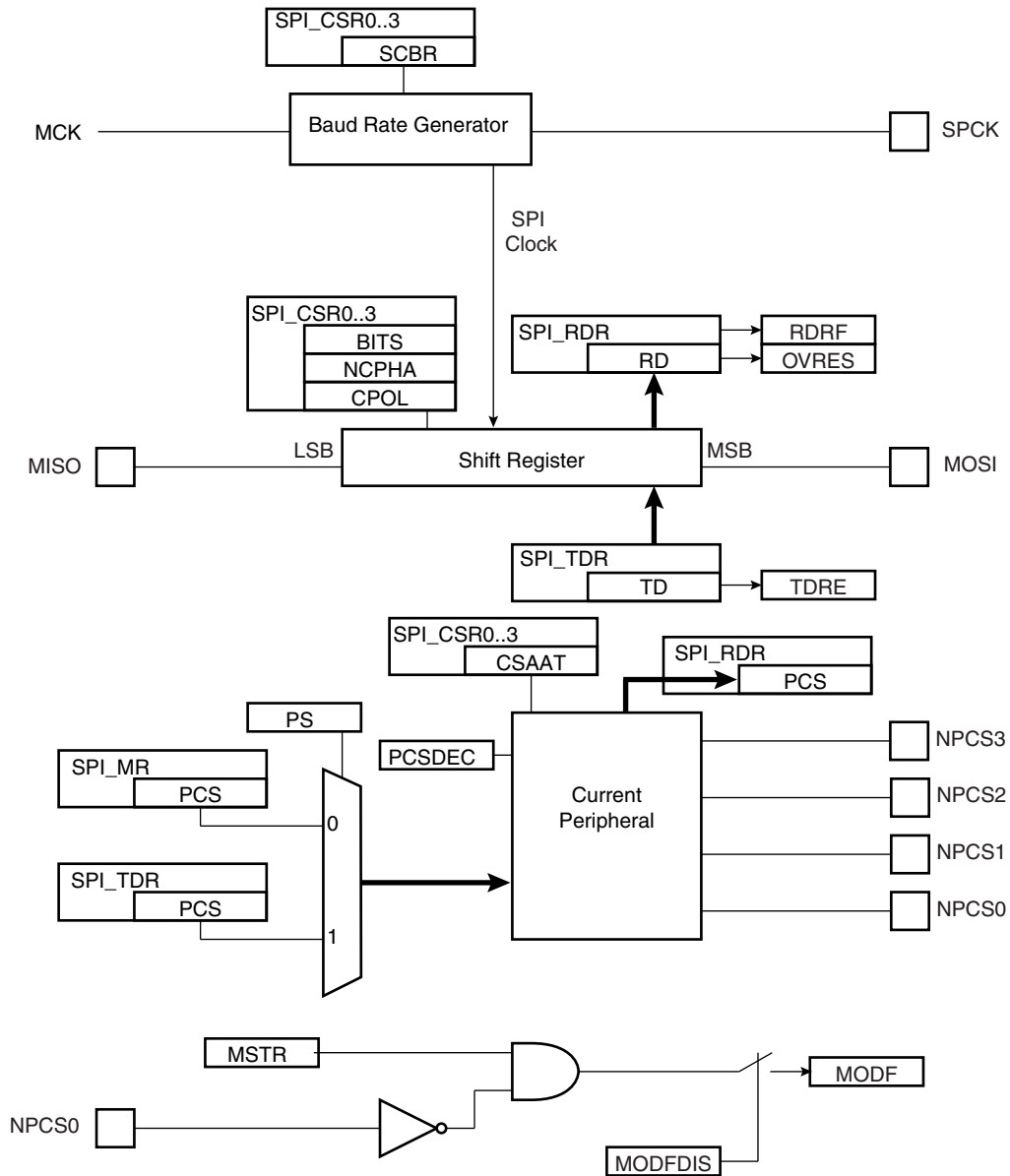
The transfer of received data from the Shift Register in SPI\_RDR is indicated by the RDRF bit (Receive Data Register Full) in the Status Register (SPI\_SR). When the received data is read, the RDRF bit is cleared.

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 32-6 on page 382](#) shows a block diagram of the SPI when operating in Master Mode. [Figure 32-6 on page 382](#) shows a flow chart describing how transfers are handled.

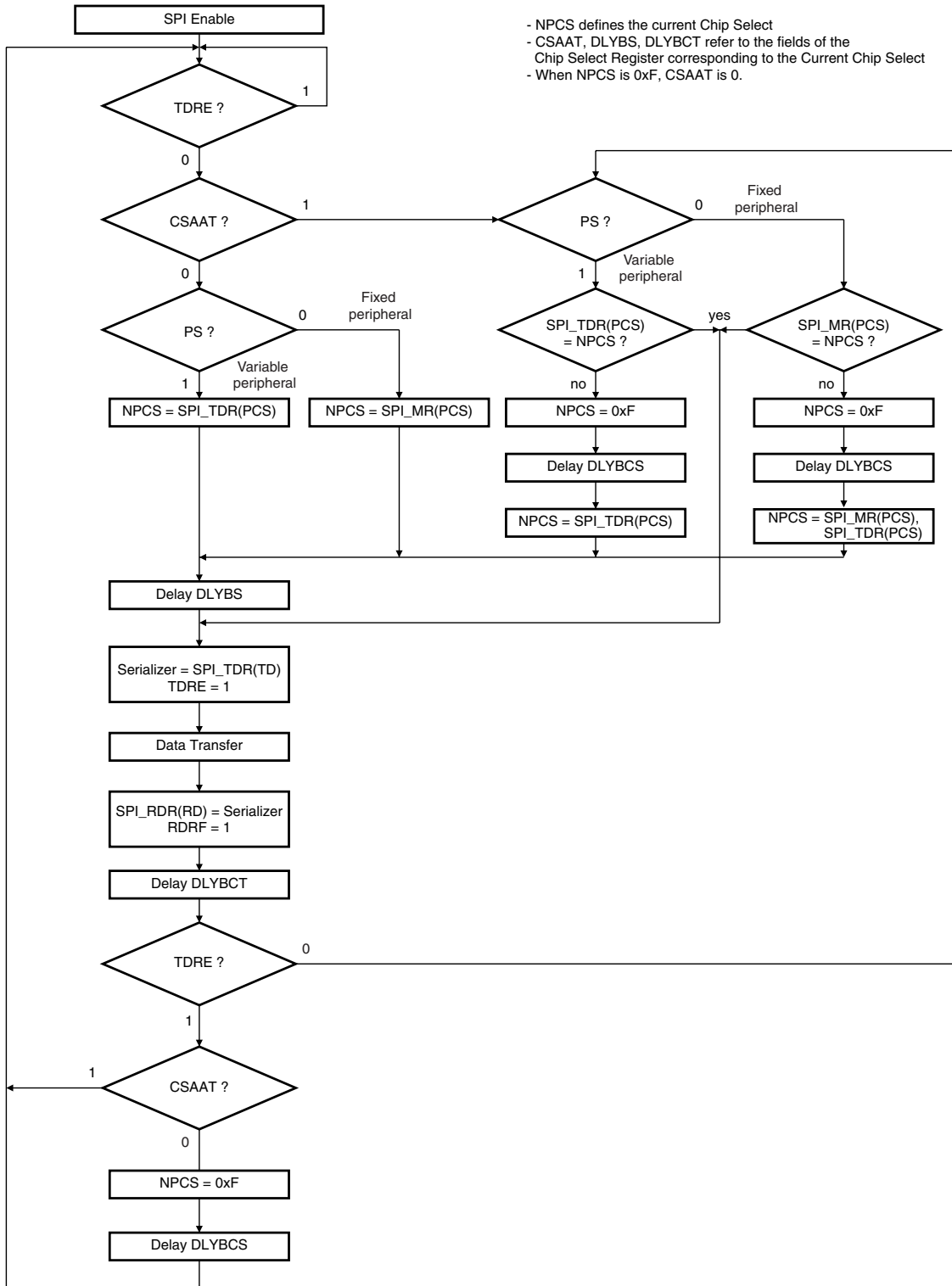
## 32.6.3.1 Master Mode Block Diagram

Figure 32-5. Master Mode Block Diagram



### 32.6.3.2 Master Mode Flow Diagram

Figure 32-6. Master Mode Flow Diagram S



### 32.6.3.3 Clock Generation

The SPI Baud rate clock is generated by dividing the Master Clock (MCK) by a value between 1 and 255.

This allows a maximum operating baud rate at up to Master Clock and a minimum operating baud rate of MCK divided by 255.

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.

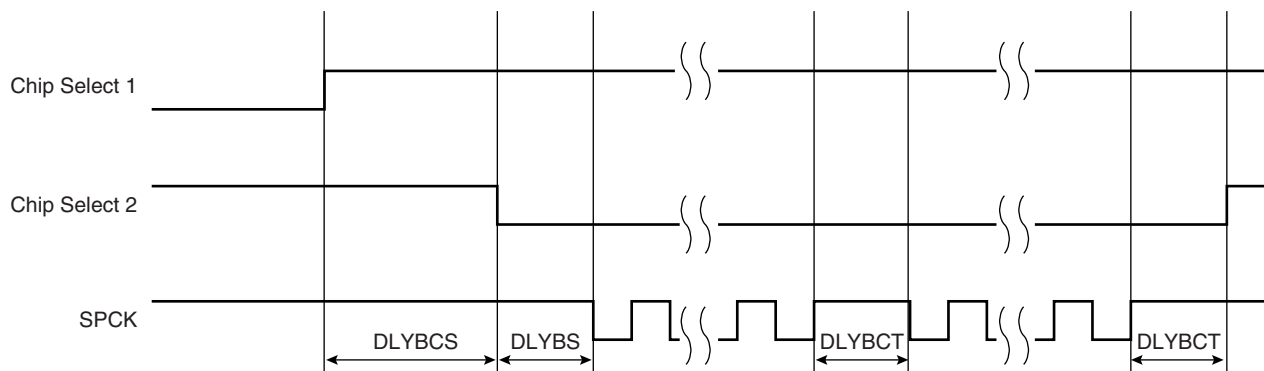
### 32.6.3.4 Transfer Delays

Figure 32-7 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 32-7.** Programmable Delays



### 32.6.3.5 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 to 16) 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.

### 32.6.3.6 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 PCS-DEC 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\_CRSD0 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.

### 32.6.3.7 Peripheral Deselection

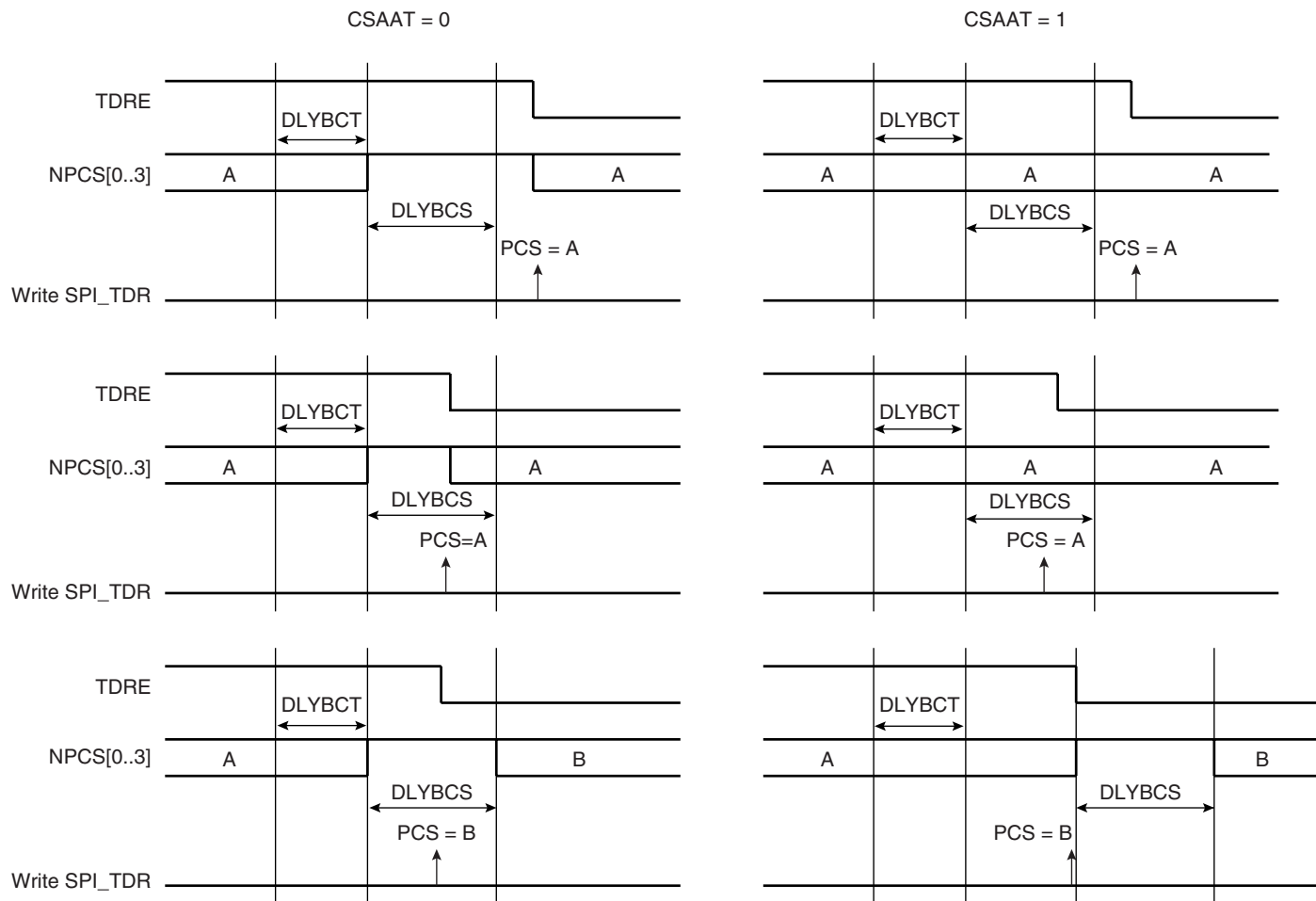
When operating normally, as soon as the transfer of the last data written in SPI\_TDR is completed, the NPCS lines all rise. This might lead to runtime error if the processor is too long in responding to an interrupt, and thus might lead to difficulties for interfacing with some serial peripherals requiring 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 32-8 shows different peripheral deselection cases and the effect of the CSAAT bit.

**Figure 32-8.** Peripheral Deselection



### 32.6.3.8 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).

### 32.6.4 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.

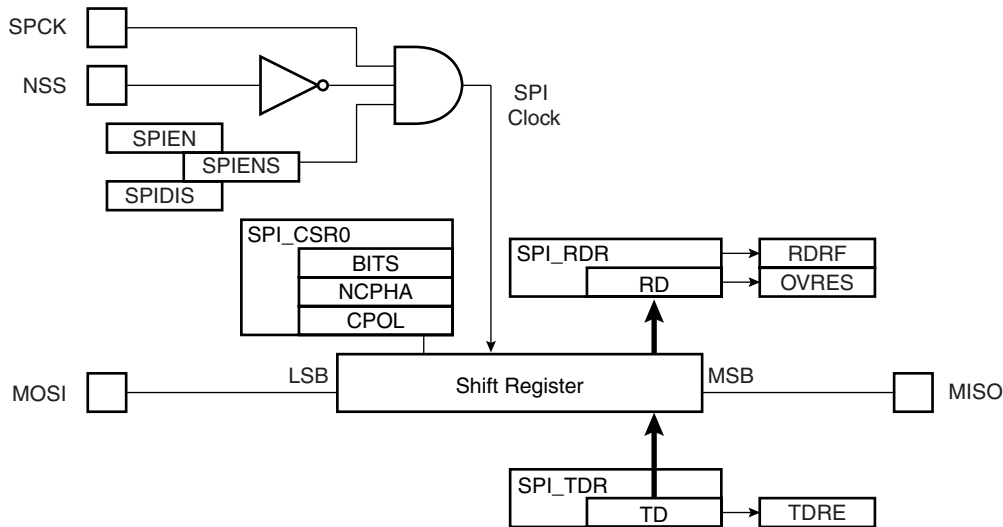
When a transfer starts, the data shifted out is the data present in the Shift Register. If no data has been written in the Transmit Data Register (SPI\_TDR), the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift Register resets 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 32-9 shows a block diagram of the SPI when operating in Slave Mode.

**Figure 32-9.** Slave Mode Functional Block Diagram



## 32.7 Serial Peripheral Interface (SPI) User Interface

**Table 32-3.** 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	0x000000F0
0x14	Interrupt Enable Register	SPI_IER	Write-only	---
0x18	Interrupt Disable Register	SPI_IDR	Write-only	---
0x1C	Interrupt Mask Register	SPI_IMR	Read-only	0x0
0x20 - 0x2C	Reserved			
0x30	Chip Select Register 0	SPI_CSR0	Read/Write	0x0
0x34	Chip Select Register 1	SPI_CSR1	Read/Write	0x0
0x38	Chip Select Register 2	SPI_CSR2	Read/Write	0x0
0x3C	Chip Select Register 3	SPI_CSR3	Read/Write	0x0
0x004C - 0x00F8	Reserved	–	–	–
0x004C - 0x00FC	Reserved	–	–	–
0x100 - 0x124	Reserved for the PDC			

### 32.7.1 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 transfer.

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.

## 32.7.2 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		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.

- **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 will be inserted by default.

Otherwise, the following equation determines the delay:

$$\text{Delay Between Chip Selects} = \frac{DLYBCS}{MCK}$$

## 32.7.3 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.

### 32.7.4 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).



## 32.7.5 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<sup>(1)</sup> or SPI\_RNCR<sup>(1)</sup>.

1 = The Receive Counter Register has reached 0 since the last write in SPI\_RCR<sup>(1)</sup> or SPI\_RNCR<sup>(1)</sup>.

- **ENDTX: End of TX buffer**

0 = The Transmit Counter Register has not reached 0 since the last write in SPI\_TCR<sup>(1)</sup> or SPI\_TNCR<sup>(1)</sup>.

1 = The Transmit Counter Register has reached 0 since the last write in SPI\_TCR<sup>(1)</sup> or SPI\_TNCR<sup>(1)</sup>.

- **RXBUFF: RX Buffer Full**

0 = SPI\_RCR<sup>(1)</sup> or SPI\_RNCR<sup>(1)</sup> has a value other than 0.

1 = Both SPI\_RCR<sup>(1)</sup> and SPI\_RNCR<sup>(1)</sup> have a value of 0.

- **TXBUFE: TX Buffer Empty**

0 = SPI\_TCR<sup>(1)</sup> or SPI\_TNCR<sup>(1)</sup> has a value other than 0.

1 = Both SPI\_TCR<sup>(1)</sup> and SPI\_TNCR<sup>(1)</sup> 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.

## 32.7.6 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.

### 32.7.7 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.

## 32.7.8 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.

## 32.7.9 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

BITS	Bits Per Transfer
1001	Reserved
1010	Reserved
1011	Reserved
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

- **SCBR: Serial Clock Baud Rate**

In Master Mode, the SPI Interface uses a modulus counter to derive the SPCK baud rate from the Master Clock MCK. The Baud rate is selected by writing a value from 1 to 255 in the SCBR field. The following equations determine the SPCK baud rate:

$$\text{SPCK Baudrate} = \frac{MCK}{SCBR}$$

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:

$$\text{Delay Before SPCK} = \frac{DLYBS}{MCK}$$

- **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:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{MCK}$$





## 33. Two-wire Interface (TWI)

### 33.1 Description

The Atmel Two-wire Interface (TWI) interconnects components on a unique two-wire bus, made up of one clock line and one data line with speeds of up to 400 Kbits per second, based on a byte-oriented transfer format. It can be used with any Atmel Two-wire Interface bus Serial EEPROM and I<sup>2</sup>C compatible device such as Real Time Clock (RTC), Dot Matrix/Graphic LCD Controllers and Temperature Sensor, to name but a few. The TWI is programmable as a master or a slave with sequential or single-byte access. Multiple master capability is supported. Arbitration of the bus is performed internally and puts the TWI in slave mode automatically if the bus arbitration is lost.

A configurable baud rate generator permits the output data rate to be adapted to a wide range of core clock frequencies.

Below, [Table 33-1](#) lists the compatibility level of the Atmel Two-wire Interface in Master Mode and a full I<sup>2</sup>C compatible device.

**Table 33-1.** Atmel TWI Compatibility with i2C Standard

I2C Standard	Atmel TWI
Standard Mode Speed (100 KHz)	Supported
Fast Mode Speed (400 KHz)	Supported
7 or 10 bits Slave Addressing	Supported
START BYTE <sup>(1)</sup>	Not Supported
Repeated Start (Sr) Condition	Supported
ACK and NACK Management	Supported
Slope control and input filtering (Fast mode)	Not Supported
Clock stretching	Supported

Note: 1. START + b000000001 + Ack + Sr

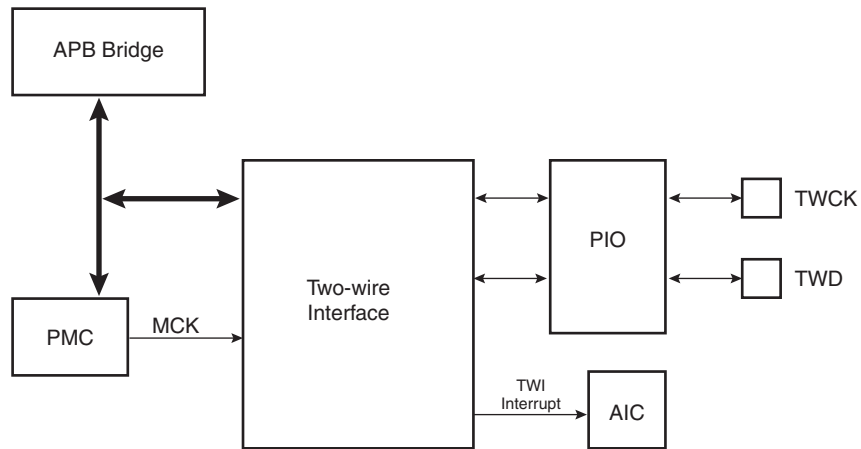
### 33.2 List of Abbreviations

**Table 33-2.** Abbreviations

Abbreviation	Description
TWI	Two-wire Interface
A	Acknowledge
NA	Non Acknowledge
P	Stop
S	Start
RS	Repeated Start
SADR	Slave Address
ADR	Any address except SADR
R	Read
W	Write

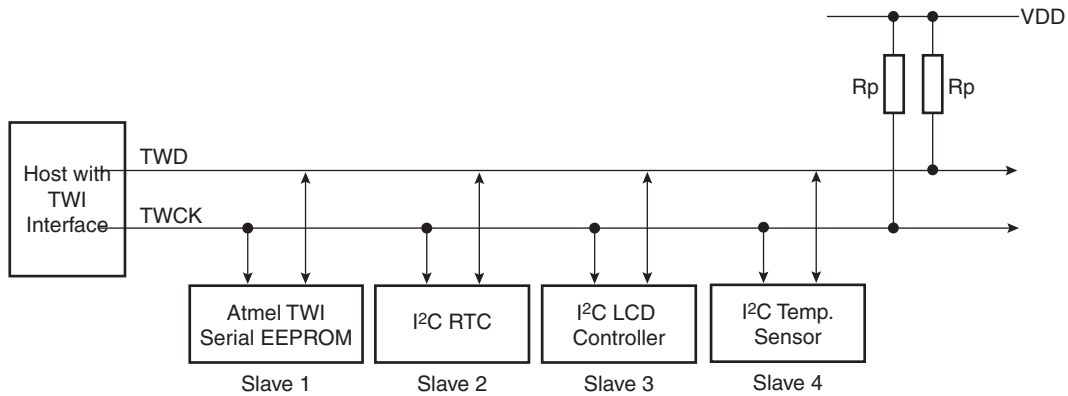
### 33.3 Block Diagram

**Figure 33-1.** Block Diagram



### 33.4 Application Block Diagram

Figure 33-2. Application Block Diagram



R<sub>p</sub>: Pull up value as given by the I<sup>2</sup>C Standard

#### 33.4.1 I/O Lines Description

Table 33-3. I/O Lines Description

Pin Name	Pin Description	Type
TWD	Two-wire Serial Data	Input/Output
TWCK	Two-wire Serial Clock	Input/Output

### 33.5 Product Dependencies

#### 33.5.1 I/O Lines

Both TWD and TWCK are bidirectional lines, connected to a positive supply voltage via a current source or pull-up resistor (see [Figure 33-2 on page 403](#)). When the bus is free, both lines are high. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function.

TWD and TWCK pins may be multiplexed with PIO lines. To enable the TWI, the programmer must perform the following steps:

- Program the PIO controller to:
  - Dedicate TWD and TWCK as peripheral lines.
  - Define TWD and TWCK as open-drain.

#### 33.5.2 Power Management

- Enable the peripheral clock.

The TWI interface may be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the TWI clock.

#### 33.5.3 Interrupt

The TWI interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). In order to handle interrupts, the AIC must be programmed before configuring the TWI.

## 33.6 Functional Description

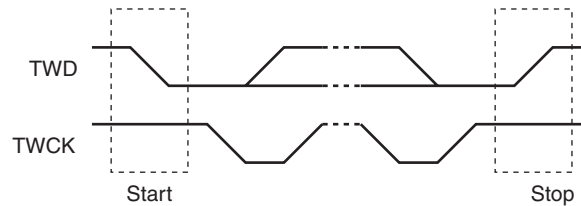
### 33.6.1 Transfer Format

The data put on the TWD line must be 8 bits long. Data is transferred MSB first; each byte must be followed by an acknowledgement. The number of bytes per transfer is unlimited (see [Figure 33-4](#)).

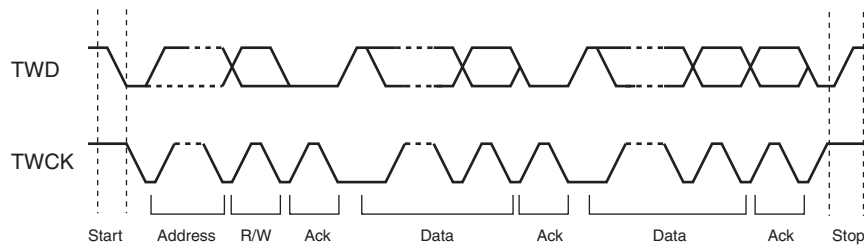
Each transfer begins with a START condition and terminates with a STOP condition (see [Figure 33-3](#)).

- A high-to-low transition on the TWD line while TWCK is high defines the START condition.
- A low-to-high transition on the TWD line while TWCK is high defines a STOP condition.

**Figure 33-3.** START and STOP Conditions



**Figure 33-4.** Transfer Format



### 33.6.2 Modes of Operation

The TWI has six modes of operations:

- Master transmitter mode
- Master receiver mode
- Multi-master transmitter mode
- Multi-master receiver mode
- Slave transmitter mode
- Slave receiver mode

These modes are described in the following chapters.

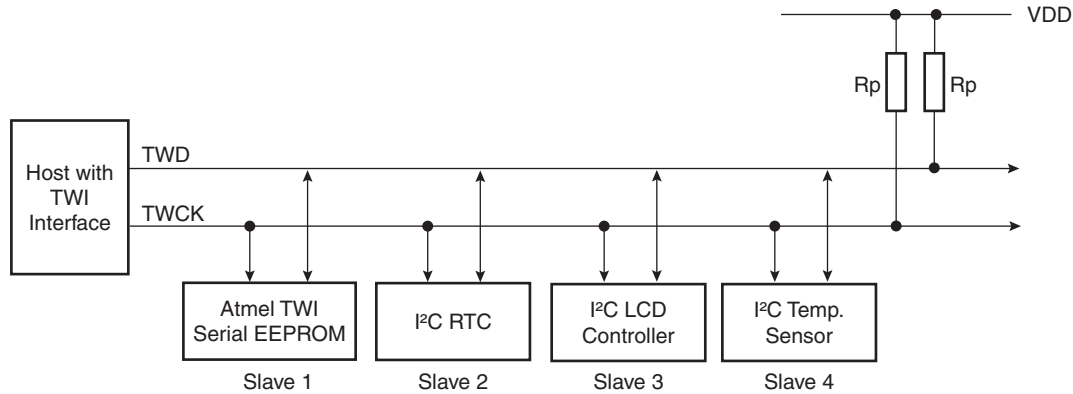
## 33.7 Master Mode

### 33.7.1 Definition

The Master is the device which starts a transfer, generates a clock and stops it.

### 33.7.2 Application Block Diagram

Figure 33-5. Master Mode Typical Application Block Diagram



Rp: Pull up value as given by the I²C Standard

### 33.7.3 Programming Master Mode

The following registers have to be programmed before entering Master mode:

1. DADR (+ IADRSZ + IADR if a 10 bit device is addressed): The device address is used to access slave devices in read or write mode.
2. CKDIV + CHDIV + CLDIV: Clock Waveform.
3. SVDIS: Disable the slave mode.
4. MSEN: Enable the master mode.

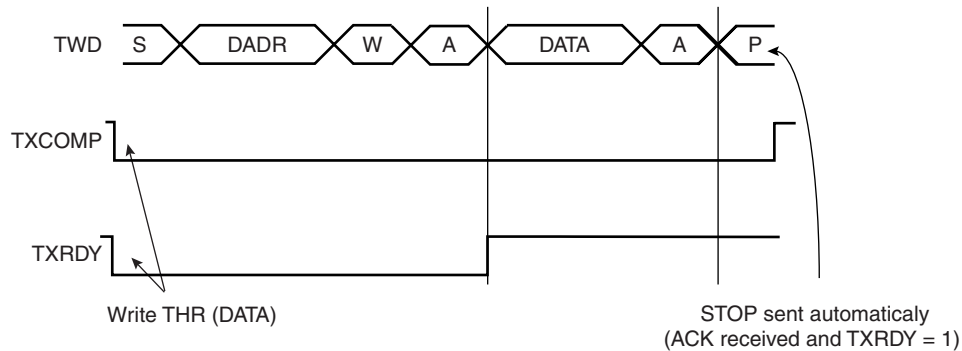
### 33.7.4 Master Transmitter Mode

After the master initiates a Start condition when writing into the Transmit Holding Register, TWI\_THR, it sends a 7-bit slave address, configured in the Master Mode register (DADR in TWI\_MMR), to notify the slave device. The bit following the slave address indicates the transfer direction, 0 in this case (MREAD = 0 in TWI\_MMR).

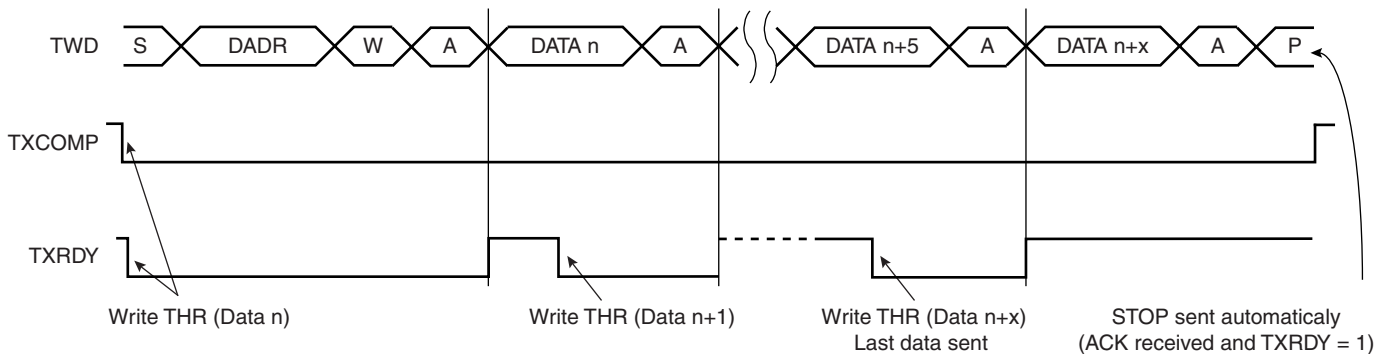
The TWI transfers require the slave to acknowledge each received byte. During the acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. The master polls the data line during this clock pulse and sets the Not Acknowledge bit (**NACK**) in the status register if the slave does not acknowledge the byte. As with the other status bits, an interrupt can be generated if enabled in the interrupt enable register (TWI\_IER). If the slave acknowledges the byte, the data written in the TWI\_THR, is then shifted in the internal shifter and transferred. When an acknowledge is detected, the TXRDY bit is set until a new write in the TWI\_THR. When no more data is written into the TWI\_THR, the master generates a stop condition to end the transfer. The end of the complete transfer is marked by the TWI\_TXCOMP bit set to one. See [Figure 33-6](#), [Figure 33-7](#), and [Figure 33-8](#).

TXRDY is used as Transmit Ready for the PDC transmit channel.

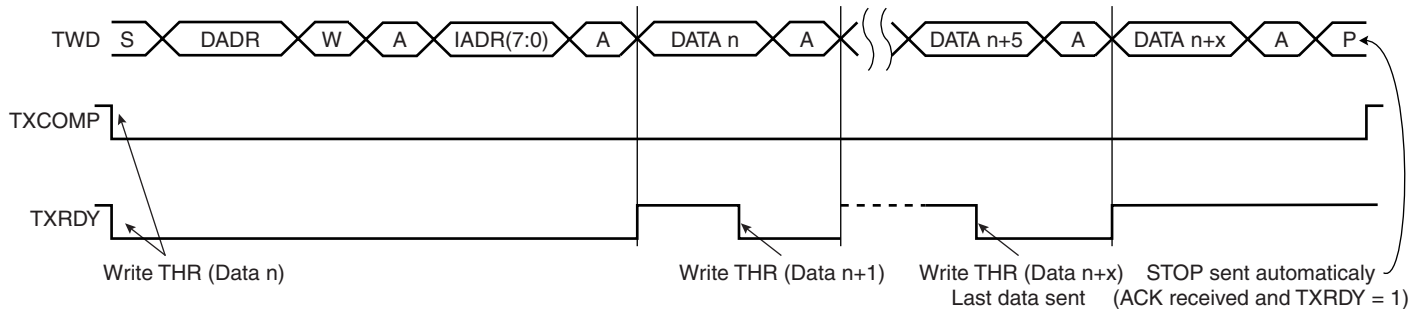
**Figure 33-6. Master Write with One Data Byte**



**Figure 33-7. Master Write with Multiple Data Byte**



**Figure 33-8. Master Write with One Byte Internal Address and Multiple Data Bytes**



### 33.7.5 Master Receiver Mode

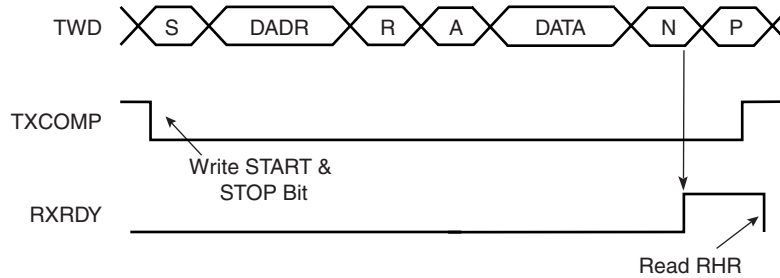
The read sequence begins by setting the START bit. After the start condition has been sent, the master sends a 7-bit slave address to notify the slave device. The bit following the slave address indicates the transfer direction, 1 in this case (MREAD = 1 in TWI\_MMR). During the acknowledge clock pulse (9th pulse), the master releases the data line (HIGH), enabling the slave to pull it down in order to generate the acknowledge. The master polls the data line during this clock pulse and sets the **NACK** bit in the status register if the slave does not acknowledge the byte.

If an acknowledge is received, the master is then ready to receive data from the slave. After data has been received, the master sends an acknowledge condition to notify the slave that the data has been received except for the last data, after the stop condition. See [Figure 33-9](#). When the

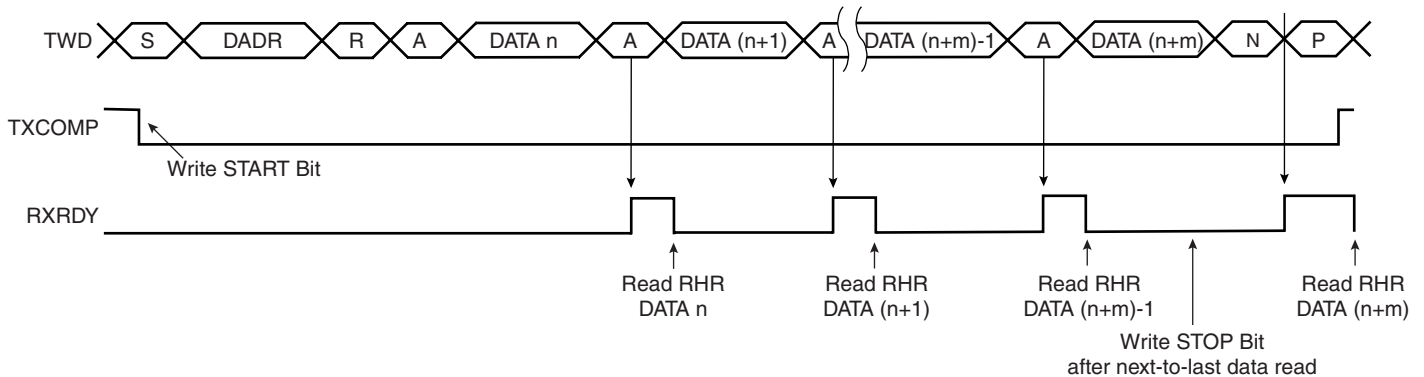
RXRDY bit is set in the status register, a character has been received in the receive-holding register (TWI\_RHR). The RXRDY bit is reset when reading the TWI\_RHR.

When a single data byte read is performed, with or without internal address (**IADR**), the START and STOP bits must be set at the same time. See [Figure 33-9](#). When a multiple data byte read is performed, with or without internal address (**IADR**), the STOP bit must be set after the next-to-last data received. See [Figure 33-10](#). For Internal Address usage see [Section 33.7.6](#).

**Figure 33-9.** Master Read with One Data Byte



**Figure 33-10.** Master Read with Multiple Data Bytes



RXRDY is used as Receive Ready for the PDC receive channel.

### 33.7.6 Internal Address

The TWI interface can perform various transfer formats: Transfers with 7-bit slave address devices and 10-bit slave address devices.

#### 33.7.6.1 7-bit Slave Addressing

When Addressing 7-bit slave devices, the internal address bytes are used to perform random address (read or write) accesses to reach one or more data bytes, within a memory page location in a serial memory, for example. When performing read operations with an internal address, the TWI performs a write operation to set the internal address into the slave device, and then switch to Master Receiver mode. Note that the second start condition (after sending the IADR) is sometimes called “repeated start” (Sr) in I2C fully-compatible devices. See [Figure 33-12](#). See [Figure 33-11](#) and [Figure 33-13](#) for Master Write operation with internal address.

The three internal address bytes are configurable through the Master Mode register (TWI\_MMR).

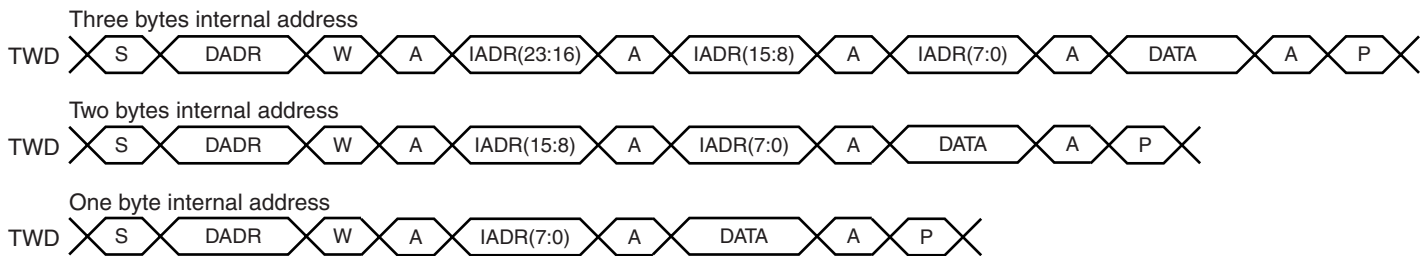
If the slave device supports only a 7-bit address, i.e. no internal address, **IADRSZ** must be set to 0.

In the figures below the following abbreviations are used:

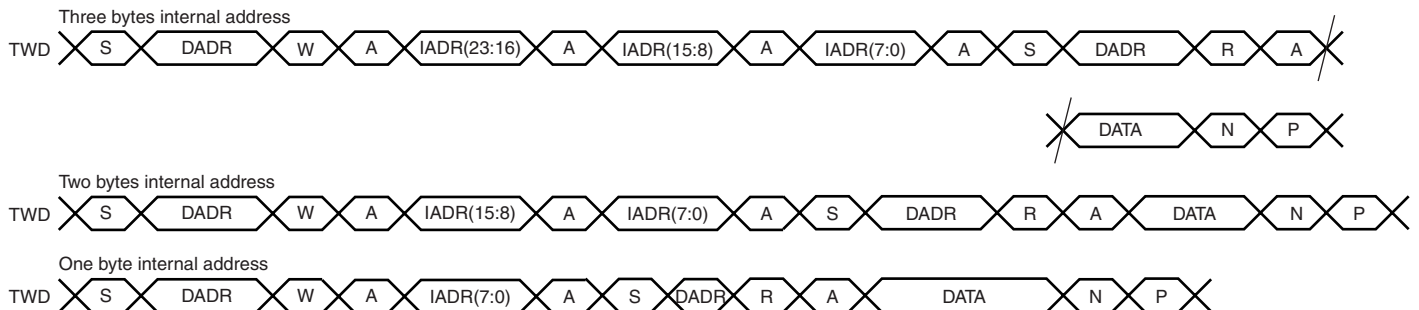
**Table 33-4.**

- S Start
- P Stop
- W Write
- R Read
- A Acknowledge
- N Not Acknowledge
- DADR Device Address
- IADR Internal Address

**Figure 33-11. Master Write with One, Two or Three Bytes Internal Address and One Data Byte**



**Figure 33-12. Master Read with One, Two or Three Bytes Internal Address and One Data Byte**



### 33.7.6.2 10-bit Slave Addressing

For a slave address higher than 7 bits, the user must configure the address size (**IADRSZ**) and set the other slave address bits in the internal address register (TWI\_IADR). The two remaining Internal address bytes, IADR[15:8] and IADR[23:16] can be used the same as in 7-bit Slave Addressing.

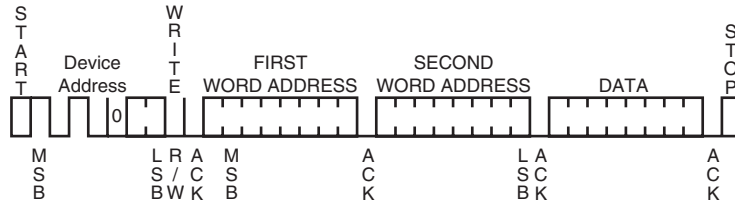


**Example:** Address a 10-bit device (10-bit device address is b1 b2 b3 b4 b5 b6 b7 b8 b9 b10)

1. Program IADRSZ = 1,
2. Program DADR with 1 1 1 1 0 b1 b2 (b1 is the MSB of the 10-bit address, b2, etc.)
3. Program TWI\_IADR with b3 b4 b5 b6 b7 b8 b9 b10 (b10 is the LSB of the 10-bit address)

Figure 33-13 below shows a byte write to an Atmel AT24LC512 EEPROM. This demonstrates the use of internal addresses to access the device.

**Figure 33-13.** Internal Address Usage



### 33.7.7 Using the Peripheral DMA Controller (PDC)

The use of the PDC significantly reduces the CPU load.

To assure correct implementation, respect the following programming sequences:

#### 33.7.7.1 *Data Transmit with the PDC*

1. Initialize the transmit PDC (memory pointers, size, etc.).
2. Configure the master mode (DADR, CKDIV, etc.).
3. Start the transfer by setting the PDC TXTEN bit.
4. Wait for the PDC end TX flag.
5. Disable the PDC by setting the PDC TXDIS bit.

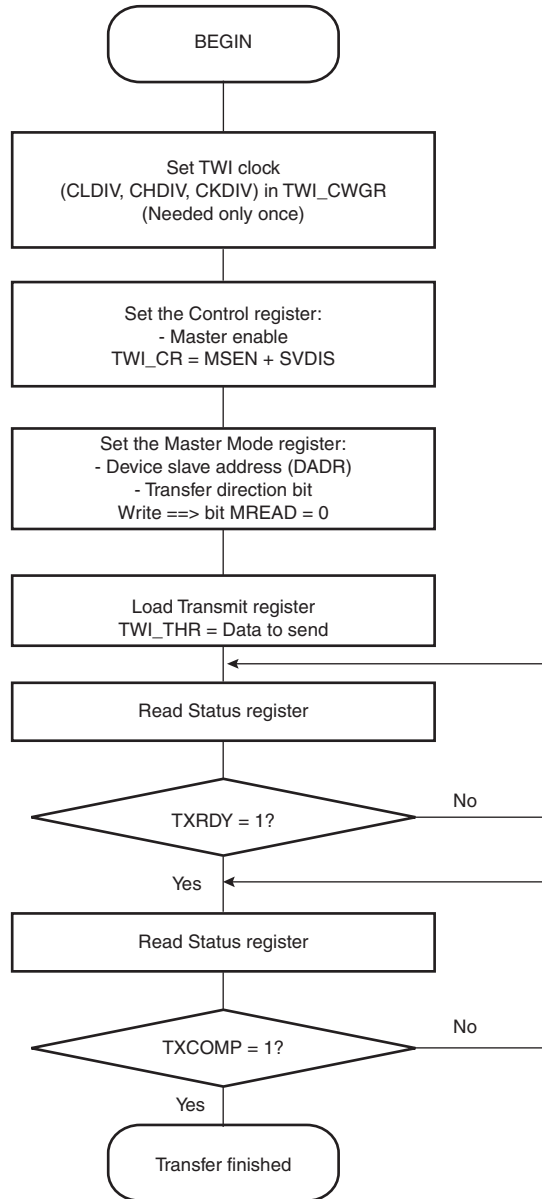
#### 33.7.7.2 *Data Receive with the PDC*

1. Initialize the receive PDC (memory pointers, size - 1, etc.).
2. Configure the master mode (DADR, CKDIV, etc.).
3. Start the transfer by setting the PDC RXTEN bit.
4. Wait for the PDC end RX flag.
5. Disable the PDC by setting the PDC RXDIS bit.

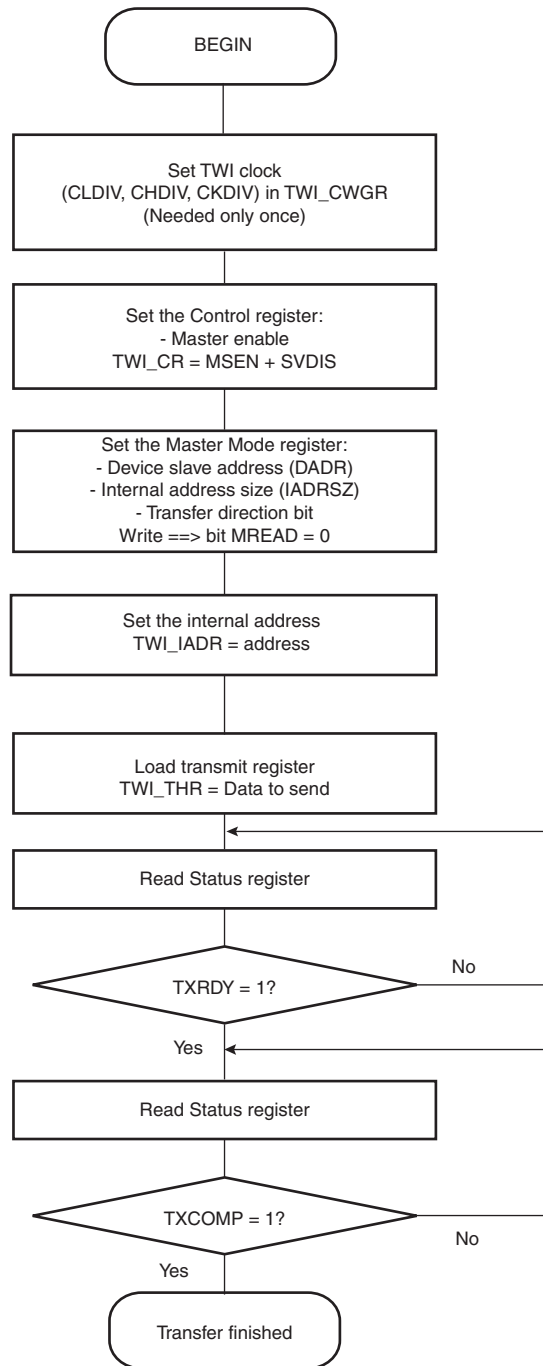
### 33.7.8 Read/Write Flowcharts

The following flowcharts shown in [Figure 33-14](#), [Figure 33-15 on page 412](#), [Figure 33-16 on page 413](#), [Figure 33-17 on page 414](#), [Figure 33-18 on page 415](#) and [Figure on page 415](#) give examples for read and write operations. A polling or interrupt method can be used to check the status bits. The interrupt method requires that the interrupt enable register (TWI\_IER) be configured first.

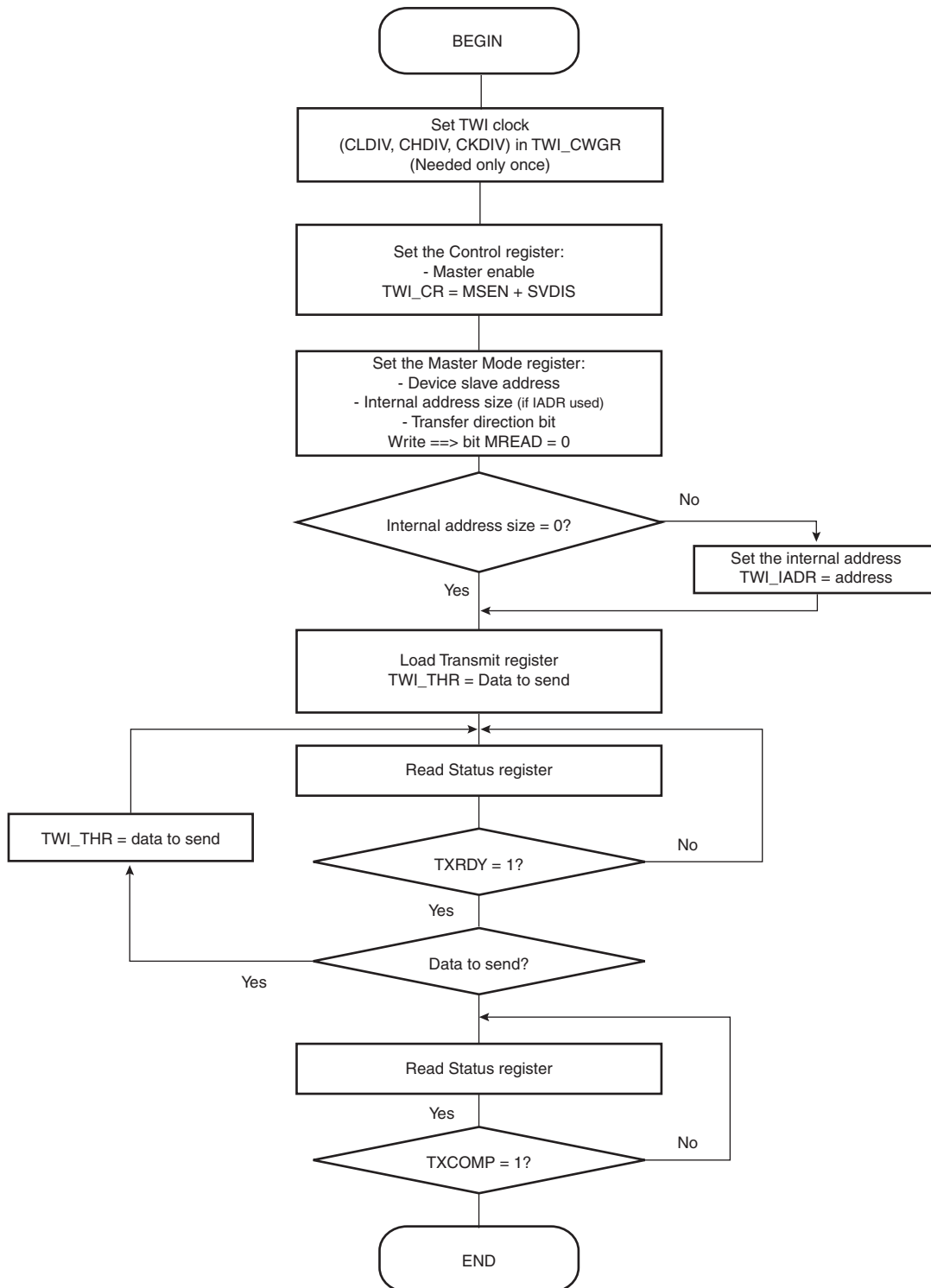
**Figure 33-14.** TWI Write Operation with Single Data Byte without Internal Address



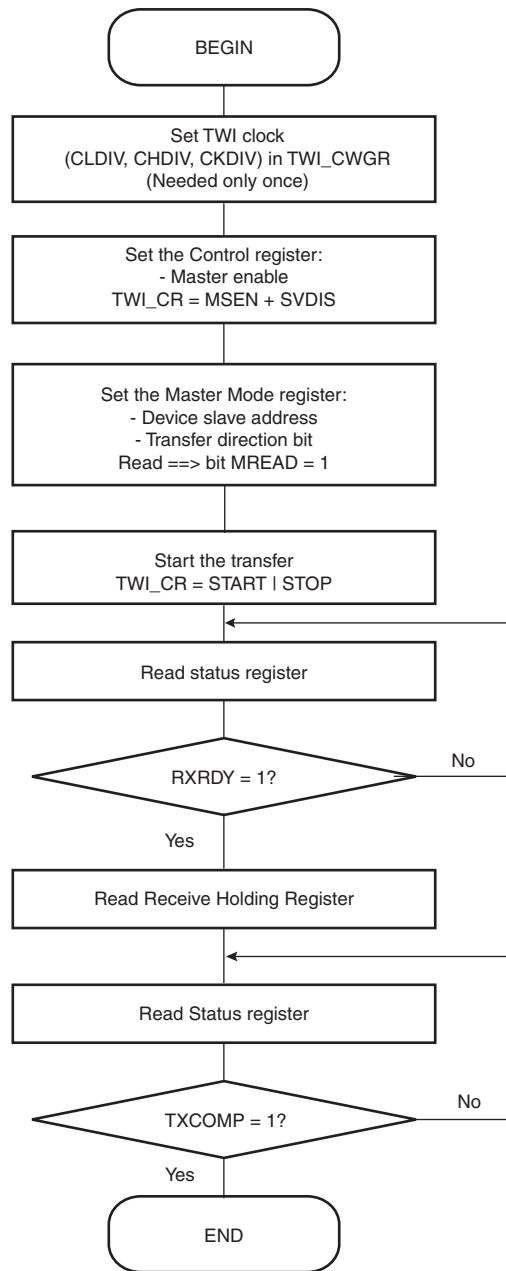
**Figure 33-15. TWI Write Operation with Single Data Byte and Internal Address**



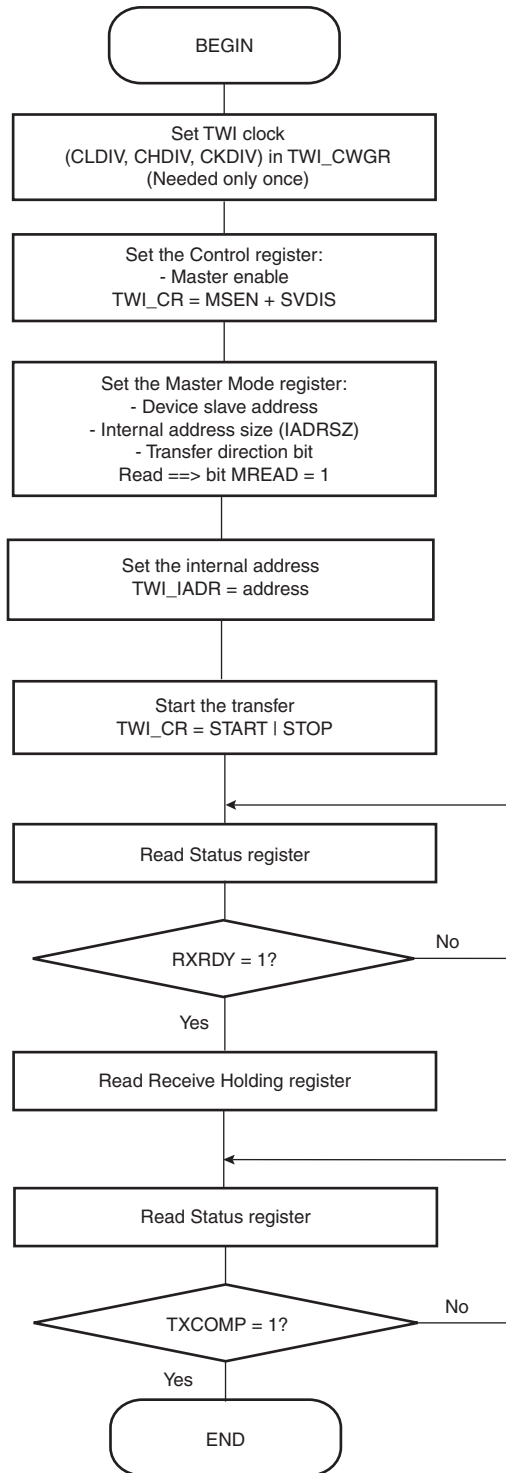
**Figure 33-16. TWI Write Operation with Multiple Data Bytes with or without Internal Address**



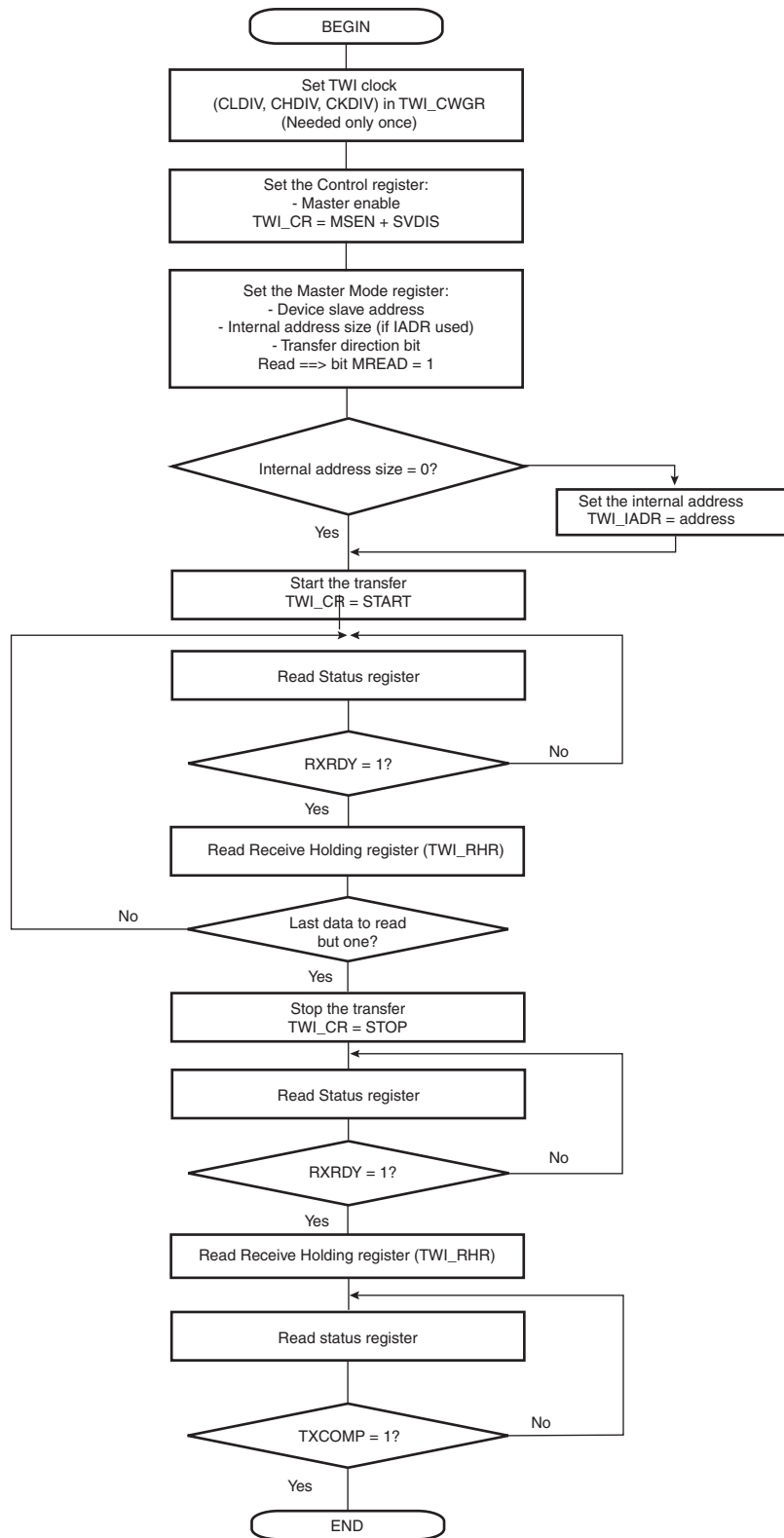
**Figure 33-17. TWI Read Operation with Single Data Byte without Internal Address**



**Figure 33-18.** TWI Read Operation with Single Data Byte and Internal Address



**Figure 33-19. TWI Read Operation with Multiple Data Bytes with or without Internal Address**





## 33.8 Multi-master Mode

### 33.8.1 Definition

More than one master may handle the bus at the same time without data corruption by using arbitration.

Arbitration starts as soon as two or more masters place information on the bus at the same time, and stops (arbitration is lost) for the master that intends to send a logical one while the other master sends a logical zero.

As soon as arbitration is lost by a master, it stops sending data and listens to the bus in order to detect a stop. When the stop is detected, the master who has lost arbitration may put its data on the bus by respecting arbitration.

Arbitration is illustrated in [Figure 33-21 on page 418](#).

### 33.8.2 Different Multi-master Modes

Two multi-master modes may be distinguished:

1. TWI is considered as a Master only and will never be addressed.
2. TWI may be either a Master or a Slave and may be addressed.

Note: In both Multi-master modes arbitration is supported.

#### 33.8.2.1 *TWI as Master Only*

In this mode, TWI is considered as a Master only (MSEN is always at one) and must be driven like a Master with the ARBLST (ARBitration Lost) flag in addition.

If arbitration is lost (ARBLST = 1), the programmer must reinitiate the data transfer.

If the user starts a transfer (ex.: DADR + START + W + Write in THR) and if the bus is busy, the TWI automatically waits for a STOP condition on the bus to initiate the transfer (see [Figure 33-20 on page 418](#)).

Note: The state of the bus (busy or free) is not indicated in the user interface.

#### 33.8.2.2 *TWI as Master or Slave*

The automatic reversal from Master to Slave is not supported in case of a lost arbitration.

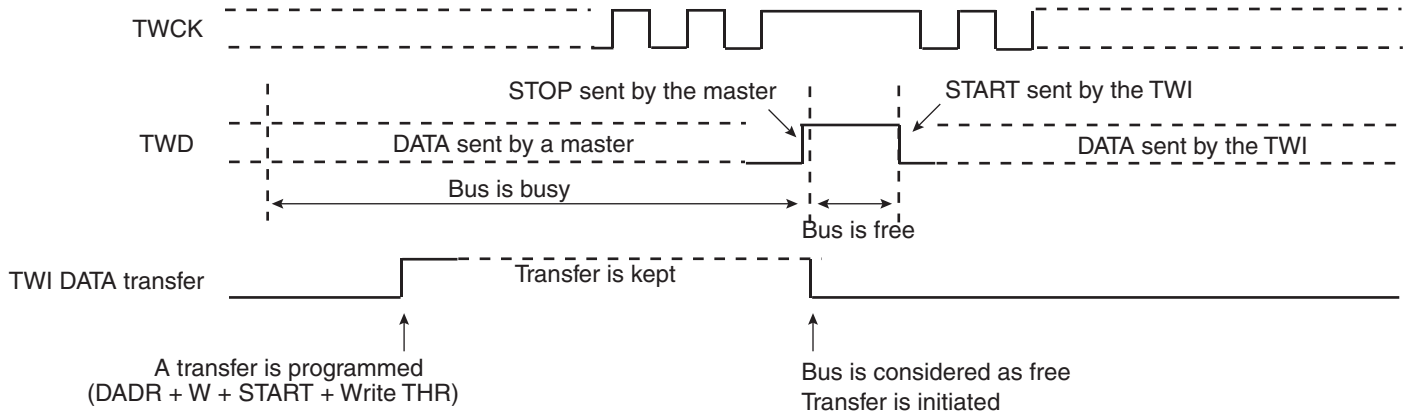
Then, in the case where TWI may be either a Master or a Slave, the programmer must manage the pseudo Multi-master mode described in the steps below.

1. Program TWI in Slave mode (SADR + MSDIS + SVEN) and perform Slave Access (if TWI is addressed).
2. If TWI has to be set in Master mode, wait until TXCOMP flag is at 1.
3. Program Master mode (DADR + SVDIS + MSEN) and start the transfer (ex: START + Write in THR).
4. As soon as the Master mode is enabled, TWI scans the bus in order to detect if it is busy or free. When the bus is considered as free, TWI initiates the transfer.
5. As soon as the transfer is initiated and until a STOP condition is sent, the arbitration becomes relevant and the user must monitor the ARBLST flag.
6. If the arbitration is lost (ARBLST is set to 1), the user must program the TWI in Slave mode in the case where the Master that won the arbitration wanted to access the TWI.

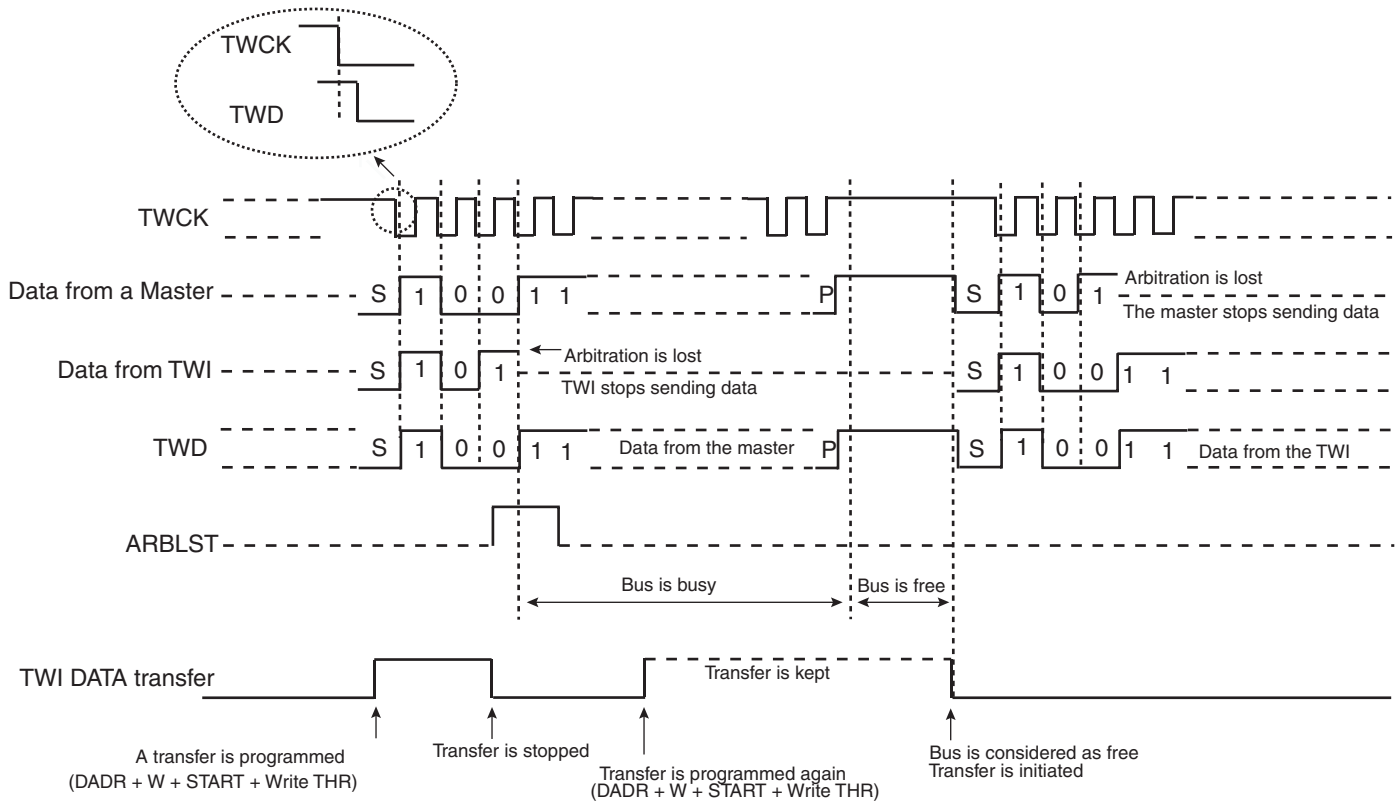
- If TWI has to be set in Slave mode, wait until TXCOMP flag is at 1 and then program the Slave mode.

Note: In the case where the arbitration is lost and TWI is addressed, TWI will not acknowledge even if it is programmed in Slave mode as soon as ARBLST is set to 1. Then, the Master must repeat SADR.

**Figure 33-20.** Programmer Sends Data While the Bus is Busy

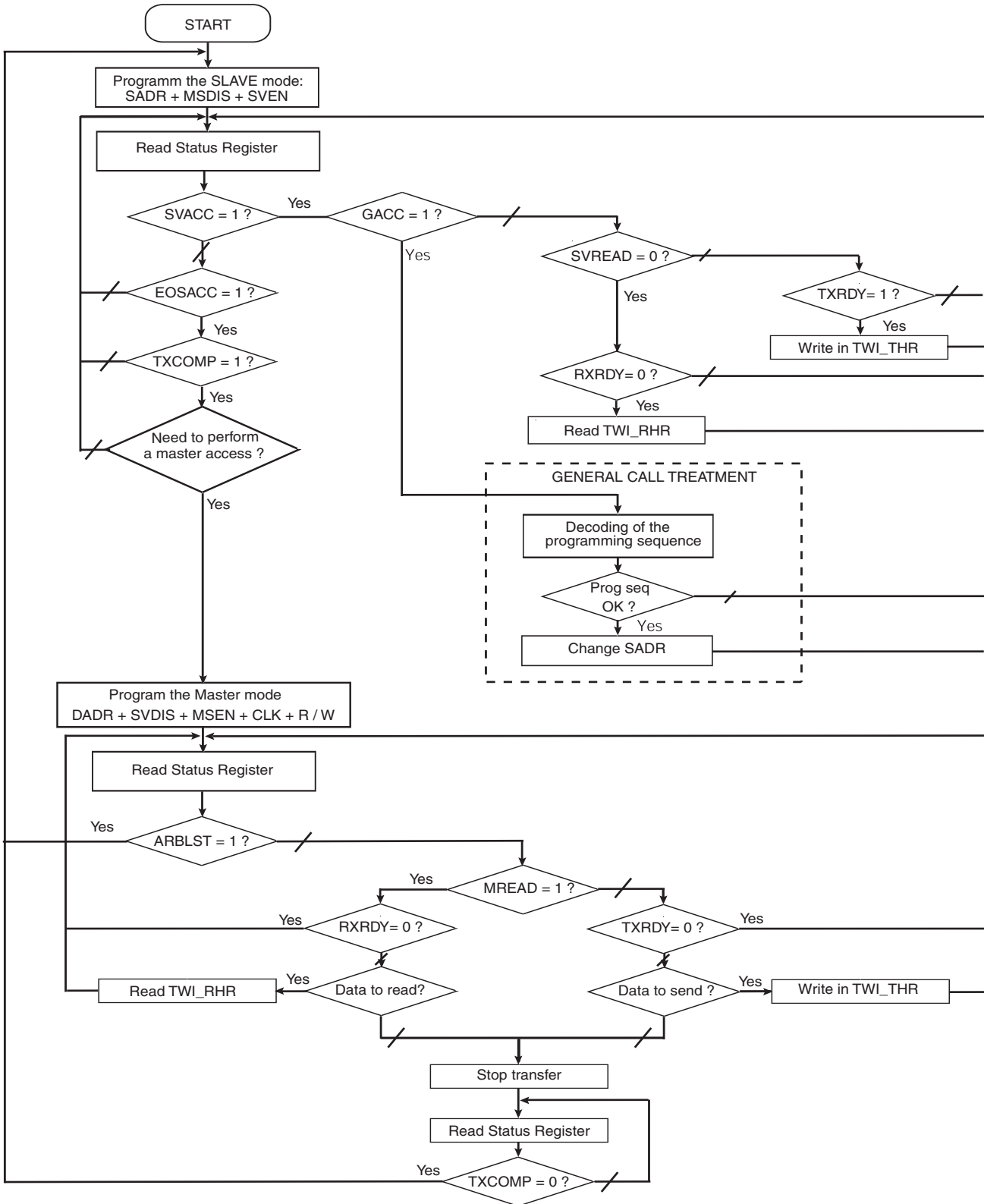


**Figure 33-21.** Arbitration Cases



The flowchart shown in [Figure 33-22 on page 419](#) gives an example of read and write operations in Multi-master mode.

Figure 33-22. Multi-master Flowchart



## 33.9 Slave Mode

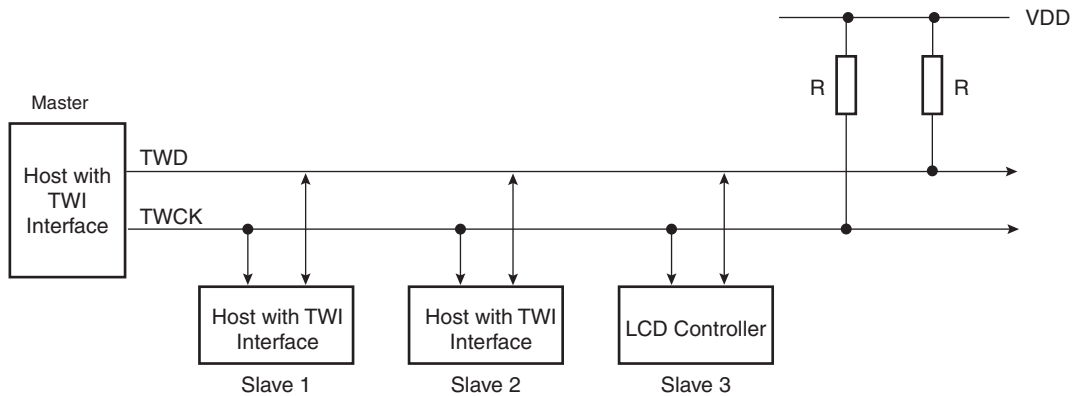
### 33.9.1 Definition

The Slave Mode is defined as a mode where the device receives the clock and the address from another device called the master.

In this mode, the device never initiates and never completes the transmission (START, REPEATED\_START and STOP conditions are always provided by the master).

### 33.9.2 Application Block Diagram

Figure 33-23. Slave Mode Typical Application Block Diagram



### 33.9.3 Programming Slave Mode

The following fields must be programmed before entering Slave mode:

1. SADR (TWI\_SMR): The slave device address is used in order to be accessed by master devices in read or write mode.
2. MSDIS (TWI\_CR): Disable the master mode.
3. SVEN (TWI\_CR): Enable the slave mode.

As the device receives the clock, values written in TWI\_CWGR are not taken into account.

### 33.9.4 Receiving Data

After a Start or Repeated Start condition is detected and if the address sent by the Master matches with the Slave address programmed in the SADR (Slave Address) field, SVACC (Slave Access) flag is set and SVREAD (Slave READ) indicates the direction of the transfer.

SVACC remains high until a STOP condition or a repeated START is detected. When such a condition is detected, EOSACC (End Of Slave Access) flag is set.

#### 33.9.4.1 Read Sequence

In the case of a Read sequence (SVREAD is high), TWI transfers data written in the TWI\_THR (TWI Transmit Holding Register) until a STOP condition or a REPEATED\_START + an address different from SADR is detected. Note that at the end of the read sequence TXCOMP (Transmission Complete) flag is set and SVACC reset.

As soon as data is written in the TWI\_THR, TXRDY (Transmit Holding Register Ready) flag is reset, and it is set when the shift register is empty and the sent data acknowledged or not. If the data is not acknowledged, the NACK flag is set.

Note that a STOP or a repeated START always follows a NACK.

See [Figure 33-24 on page 422](#).

#### 33.9.4.2 Write Sequence

In the case of a Write sequence (SVREAD is low), the RXRDY (Receive Holding Register Ready) flag is set as soon as a character has been received in the TWI\_RHR (TWI Receive Holding Register). RXRDY is reset when reading the TWI\_RHR.

TWI continues receiving data until a STOP condition or a REPEATED\_START + an address different from SADR is detected. Note that at the end of the write sequence TXCOMP flag is set and SVACC reset.

See [Figure 33-25 on page 422](#).

#### 33.9.4.3 Clock Synchronization Sequence

In the case where TWI\_THR or TWI\_RHR is not written/read in time, TWI performs a clock synchronization.

Clock stretching information is given by the SCLWS (Clock Wait state) bit.

See [Figure 33-27 on page 424](#) and [Figure 33-28 on page 425](#).

#### 33.9.4.4 General Call

In the case where a GENERAL CALL is performed, GACC (General Call ACCESS) flag is set.

After GACC is set, it is up to the programmer to interpret the meaning of the GENERAL CALL and to decode the new address programming sequence.

See [Figure 33-26 on page 423](#).

#### 33.9.4.5 PDC

As it is impossible to know the exact number of data to receive/send, the use of PDC is NOT recommended in SLAVE mode.

### 33.9.5 Data Transfer

#### 33.9.5.1 Read Operation

The read mode is defined as a data requirement from the master.

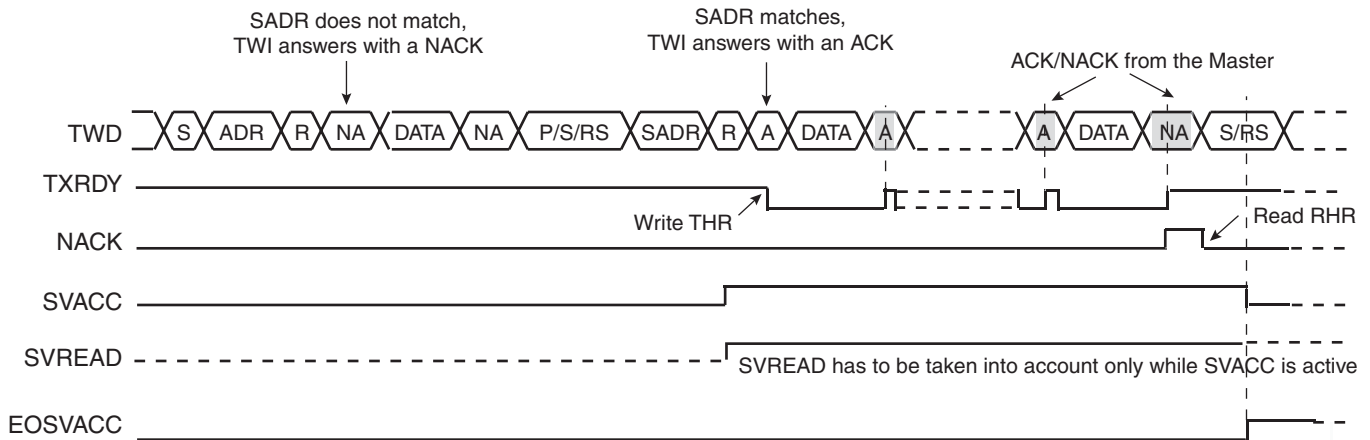
After a START or a REPEATED START condition is detected, the decoding of the address starts. If the slave address (SADR) is decoded, SVACC is set and SVREAD indicates the direction of the transfer.

Until a STOP or REPEATED START condition is detected, TWI continues sending data loaded in the TWI\_THR register.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

[Figure 33-24 on page 422](#) describes the write operation.

**Figure 33-24. Read Access Ordered by a MASTER**



- Notes:
1. When SVACC is low, the state of SVREAD becomes irrelevant.
  2. TXRDY is reset when data has been transmitted from TWI\_THR to the shift register and set when this data has been acknowledged or non acknowledged.

### 33.9.5.2 Write Operation

The write mode is defined as a data transmission from the master.

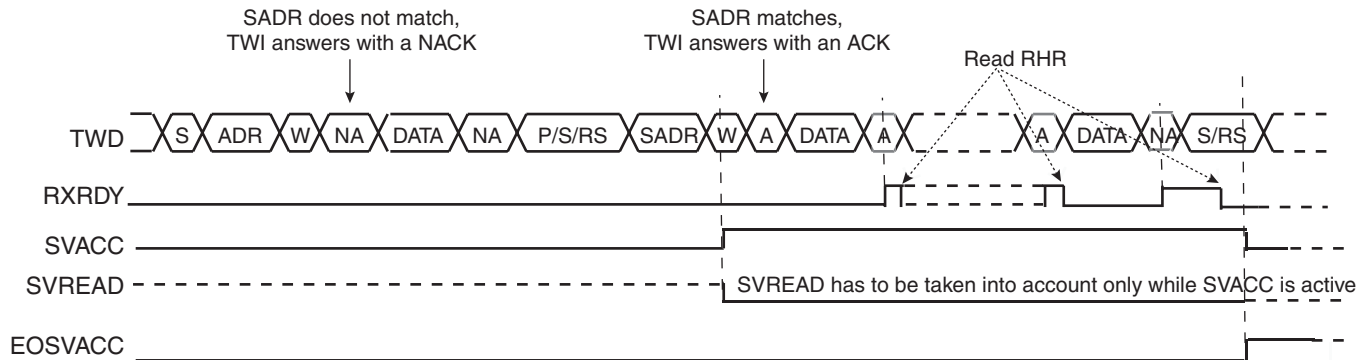
After a START or a REPEATED START, the decoding of the address starts. If the slave address is decoded, SVACC is set and SVREAD indicates the direction of the transfer (SVREAD is low in this case).

Until a STOP or REPEATED START condition is detected, TWI stores the received data in the TWI\_RHR register.

If a STOP condition or a REPEATED START + an address different from SADR is detected, SVACC is reset.

Figure 33-25 on page 422 describes the Write operation.

**Figure 33-25. Write Access Ordered by a Master**



- Notes:
1. When SVACC is low, the state of SVREAD becomes irrelevant.
  2. RXRDY is set when data has been transmitted from the shift register to the TWI\_RHR and reset when this data is read.

### 33.9.5.3 General Call

The general call is performed in order to change the address of the slave.

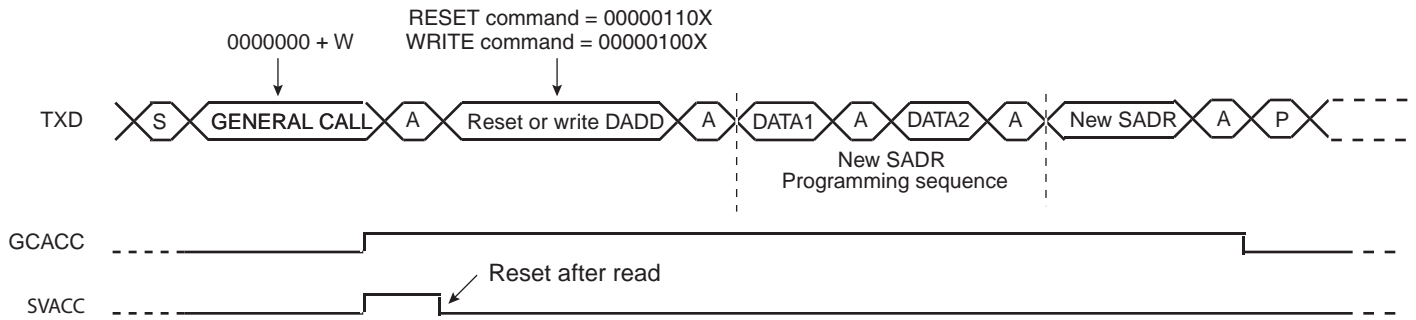
If a GENERAL CALL is detected, GACC is set.

After the detection of General Call, it is up to the programmer to decode the commands which come afterwards.

In case of a WRITE command, the programmer has to decode the programming sequence and program a new SADR if the programming sequence matches.

Figure 33-26 on page 423 describes the General Call access.

**Figure 33-26. Master Performs a General Call**



Note: This method allows the user to create an own programming sequence by choosing the programming bytes and the number of them. The programming sequence has to be provided to the master.

### 33.9.5.4 Clock Synchronization

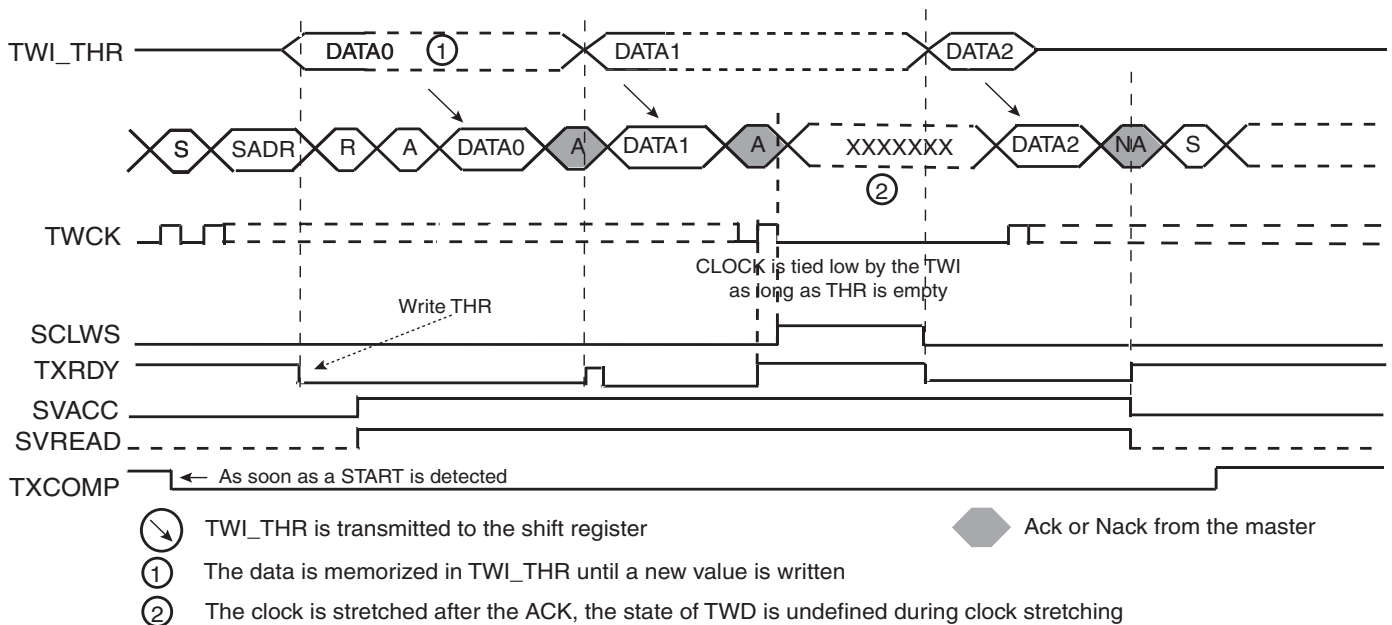
In both read and write modes, it may happen that TWI\_THR/TWI\_RHR buffer is not filled /emptied before the emission/reception of a new character. In this case, to avoid sending/receiving undesired data, a clock stretching mechanism is implemented.

### 33.9.5.5 Clock Synchronization in Read Mode

The clock is tied low if the shift register is empty and if a STOP or REPEATED START condition was not detected. It is tied low until the shift register is loaded.

Figure 33-27 on page 424 describes the clock synchronization in Read mode.

**Figure 33-27.** Clock Synchronization in Read Mode



- Notes:
1. TXRDY is reset when data has been written in the TWI\_TH to the shift register and set when this data has been acknowledged or non acknowledged.
  2. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED\_START + an address different from SADR.
  3. SCLWS is automatically set when the clock synchronization mechanism is started.

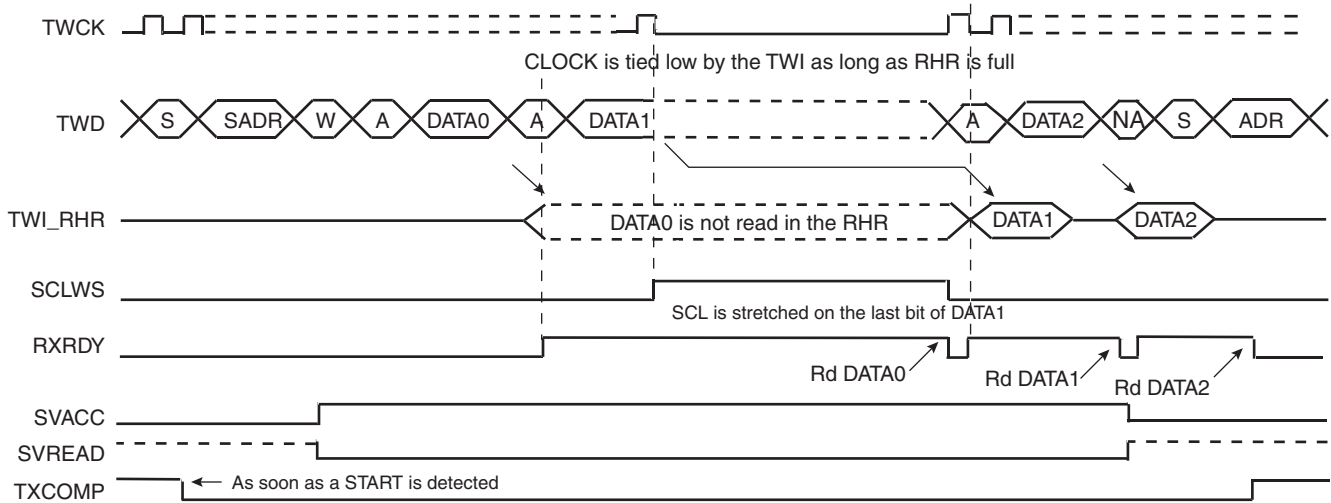


### 33.9.5.6 Clock Synchronization in Write Mode

The clock is tied low if the shift register and the TWI\_RHR is full. If a STOP or REPEATED\_START condition was not detected, it is tied low until TWI\_RHR is read.

Figure 33-28 on page 425 describes the clock synchronization in Read mode.

**Figure 33-28.** Clock Synchronization in Write Mode



- Notes:
1. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED\_START + an address different from SADR.
  2. SCLWS is automatically set when the clock synchronization mechanism is started and automatically reset when the mechanism is finished.

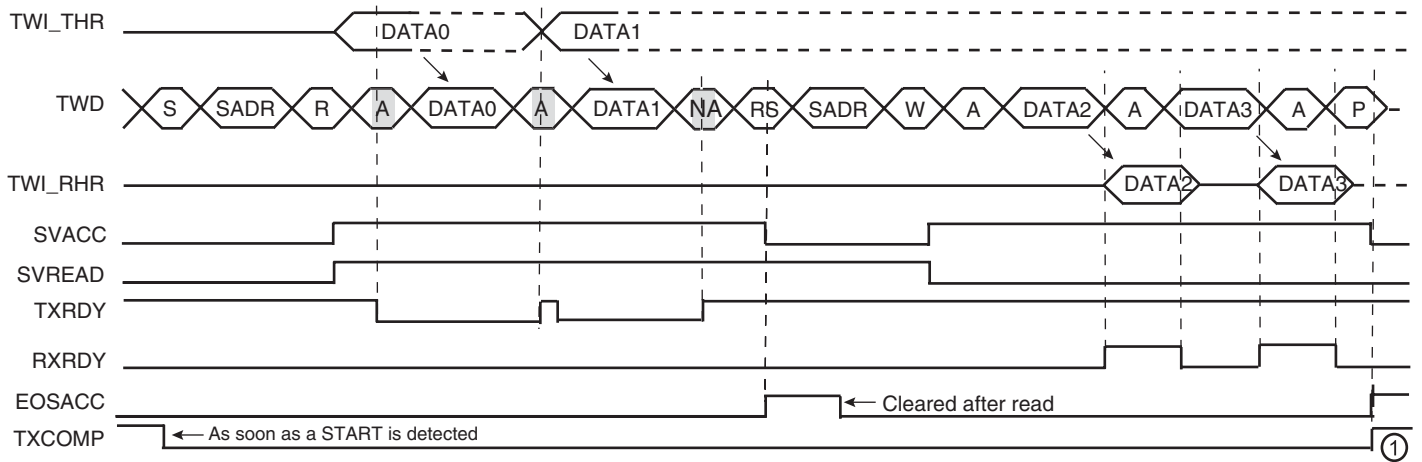
### 33.9.5.7 Reversal after a Repeated Start

### 33.9.5.8 Reversal of Read to Write

The master initiates the communication by a read command and finishes it by a write command.

Figure 33-29 on page 426 describes the repeated start + reversal from Read to Write mode.

**Figure 33-29.** Repeated Start + Reversal from Read to Write Mode

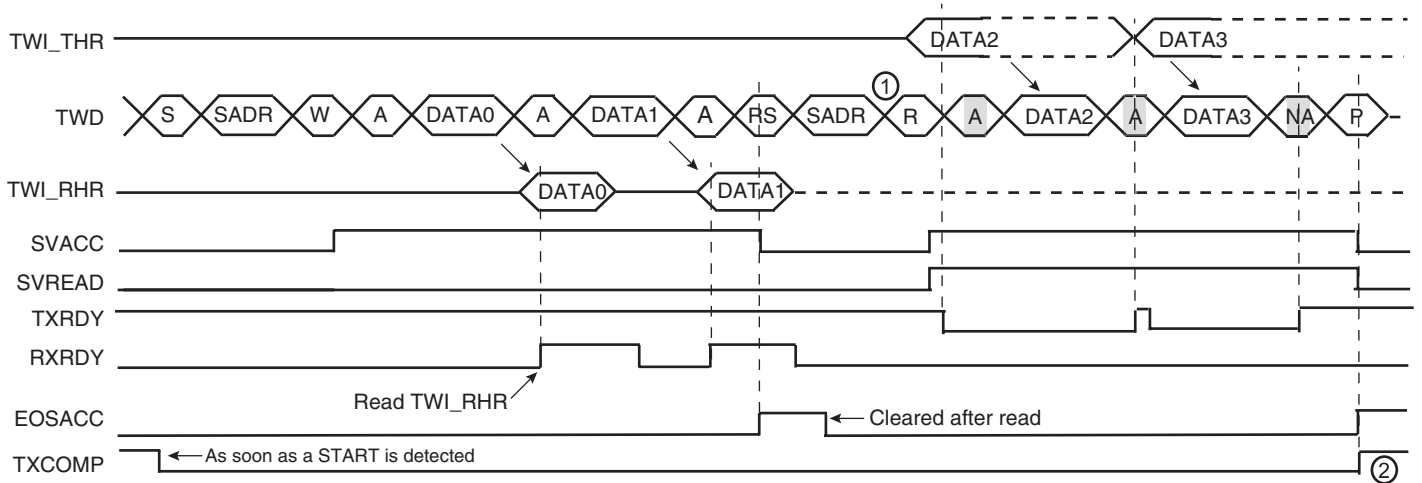


1. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

### 33.9.5.9 Reversal of Write to Read

The master initiates the communication by a write command and finishes it by a read command. Figure 33-30 on page 426 describes the repeated start + reversal from Write to Read mode.

**Figure 33-30.** Repeated Start + Reversal from Write to Read Mode

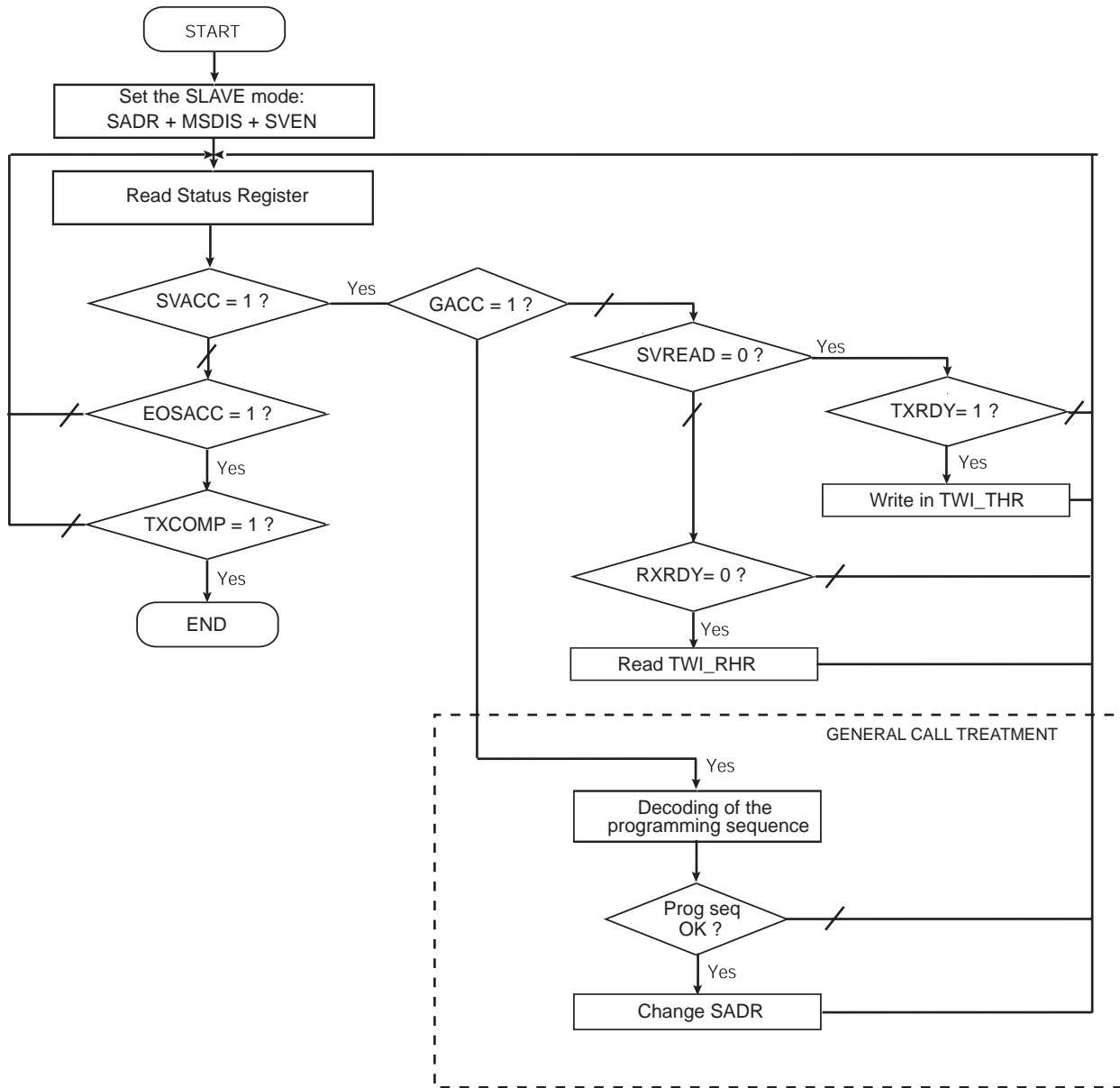


- Notes:
1. In this case, if TWI\_THR has not been written at the end of the read command, the clock is automatically stretched before the ACK.
  2. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

### 33.9.6 Read Write Flowcharts

The flowchart shown in [Figure 33-31 on page 427](#) gives an example of read and write operations in Slave mode. A polling or interrupt method can be used to check the status bits. The interrupt method requires that the interrupt enable register (TWI\_IER) be configured first.

**Figure 33-31.** Read Write Flowchart in Slave Mode



### 33.10 Two-wire Interface (TWI) User Interface

**Table 33-5.** Two-wire Interface (TWI) User Interface

Offset	Register	Name	Access	Reset State
0x00	Control Register	TWI_CR	Write-only	N / A
0x04	Master Mode Register	TWI_MMR	Read/Write	0x00000000
0x08	Slave Mode Register	TWI_SMR	Read/Write	0x00000000
0x0C	Internal Address Register	TWI_IADR	Read/Write	0x00000000
0x10	Clock Waveform Generator Register	TWI_CWGR	Read/Write	0x00000000
0x20	Status Register	TWI_SR	Read-only	0x0000F009
0x24	Interrupt Enable Register	TWI_IER	Write-only	N / A
0x28	Interrupt Disable Register	TWI_IDR	Write-only	N / A
0x2C	Interrupt Mask Register	TWI_IMR	Read-only	0x00000000
0x30	Receive Holding Register	TWI_RHR	Read-only	0x00000000
0x34	Transmit Holding Register	TWI_THR	Write-only	0x00000000
0x38 - 0xFC	Reserved	–	–	–
0x100 - 0x124	Reserved for the PDC	–	–	–

### 33.10.1 TWI Control Register

Name: TWI\_CR

Access: Write-only

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
SWRST	–	SVDIS	SVEN	MSDIS	MSEN	STOP	START

- **START: Send a START Condition**

0 = No effect.

1 = A frame beginning with a START bit is transmitted according to the features defined in the mode register.

This action is necessary when the TWI peripheral wants to read data from a slave. When configured in Master Mode with a write operation, a frame is sent as soon as the user writes a character in the Transmit Holding Register (TWI\_THR).

- **STOP: Send a STOP Condition**

0 = No effect.

1 = STOP Condition is sent just after completing the current byte transmission in master read mode.

- In single data byte master read, the START and STOP must both be set.
- In multiple data bytes master read, the STOP must be set after the last data received but one.
- In master read mode, if a NACK bit is received, the STOP is automatically performed.
- In multiple data write operation, when both THR and shift register are empty, a STOP condition is automatically sent.

- **MSEN: TWI Master Mode Enabled**

0 = No effect.

1 = If MSDIS = 0, the master mode is enabled.

Note: Switching from Slave to Master mode is only permitted when TXCOMP = 1.

- **MSDIS: TWI Master Mode Disabled**

0 = No effect.

1 = The master mode is disabled, all pending data is transmitted. The shifter and holding characters (if it contains data) are transmitted in case of write operation. In read operation, the character being transferred must be completely received before disabling.

- **SVEN: TWI Slave Mode Enabled**

0 = No effect.

1 = If SVDIS = 0, the slave mode is enabled.

Note: Switching from Master to Slave mode is only permitted when TXCOMP = 1.

- **SVDIS: TWI Slave Mode Disabled**

0 = No effect.

1 = The slave mode is disabled. The shifter and holding characters (if it contains data) are transmitted in case of read operation. In write operation, the character being transferred must be completely received before disabling.

- **SWRST: Software Reset**

0 = No effect.

1 = Equivalent to a system reset.



### 33.10.2 TWI Master Mode Register

Name: TWI\_MMR

Access: Read/Write

Reset Value: 0x00000000

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

• **IADRSZ: Internal Device Address Size**

IADRSZ[9:8]		
0	0	No internal device address
0	1	One-byte internal device address
1	0	Two-byte internal device address
1	1	Three-byte internal device address

• **MREAD: Master Read Direction**

0 = Master write direction.

1 = Master read direction.

• **DADR: Device Address**

The device address is used to access slave devices in read or write mode. Those bits are only used in Master mode.



### 33.10.3 TWI Slave Mode Register

Name: TWI\_SMR

Access: Read/Write

Reset Value: 0x00000000

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

- **SADR: Slave Address**

The slave device address is used in Slave mode in order to be accessed by master devices in read or write mode.

SADR must be programmed before enabling the Slave mode or after a general call. Writes at other times have no effect.



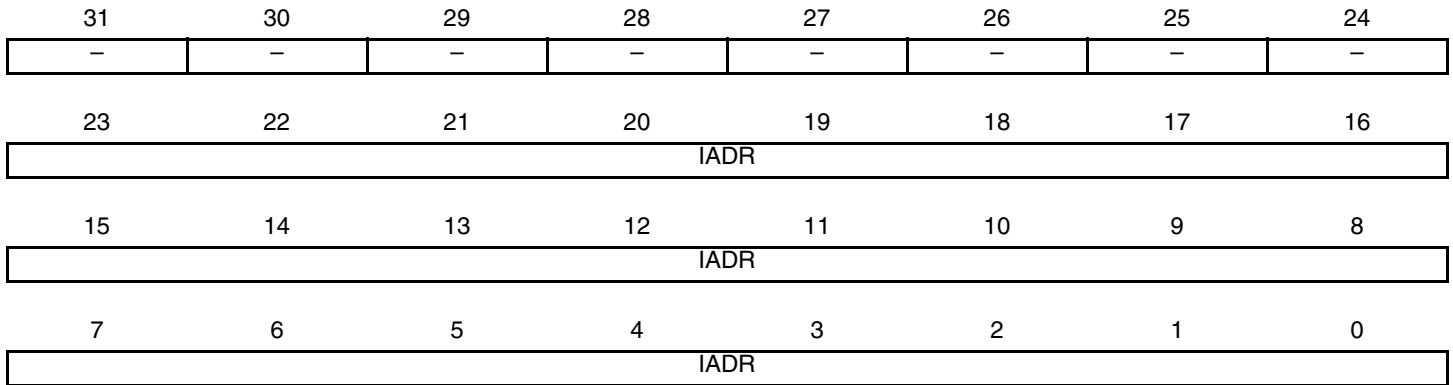


### 33.10.4 TWI Internal Address Register

Name: TWI\_IADR

Access: Read/Write

Reset Value: 0x00000000



- **IADR: Internal Address**

0, 1, 2 or 3 bytes depending on IADRSZ.



### 33.10.5 TWI Clock Waveform Generator Register

Name: TWI\_CWGR

Access: Read/Write

Reset Value: 0x00000000

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

TWI\_CWGR is only used in Master mode.

- **CLDIV: Clock Low Divider**

The SCL low period is defined as follows:

$$T_{low} = ((CLDIV \times 2^{CKDIV}) + 4) \times T_{MCK}$$

- **CHDIV: Clock High Divider**

The SCL high period is defined as follows:

$$T_{high} = ((CHDIV \times 2^{CKDIV}) + 4) \times T_{MCK}$$

- **CKDIV: Clock Divider**

The CKDIV is used to increase both SCL high and low periods.

### 33.10.6 TWI Status Register

Name: TWI\_SR

Access: Read-only

Reset Value: 0x0000F009

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
TXBUFE	RXBUFF	ENDTX	ENDRX	EOSACC	SCLWS	ARBLST	NACK
7	6	5	4	3	2	1	0
–	OVRE	GACC	SVACC	SVREAD	TXRDY	RXRDY	TXCOMP

- **TXCOMP: Transmission Completed (automatically set / reset)**

TXCOMP used in Master mode:

0 = During the length of the current frame.

1 = When both holding and shifter registers are empty and STOP condition has been sent.

*TXCOMP behavior in Master mode* can be seen in [Figure 33-8 on page 406](#) and in [Figure 33-10 on page 407](#).

TXCOMP used in Slave mode:

0 = As soon as a Start is detected.

1 = After a Stop or a Repeated Start + an address different from SADR is detected.

*TXCOMP behavior in Slave mode* can be seen in [Figure 33-27 on page 424](#), [Figure 33-28 on page 425](#), [Figure 33-29 on page 426](#) and [Figure 33-30 on page 426](#).

- **RXRDY: Receive Holding Register Ready (automatically set / reset)**

0 = No character has been received since the last TWI\_RHR read operation.

1 = A byte has been received in the TWI\_RHR since the last read.

*RXRDY behavior in Master mode* can be seen in [Figure 33-10 on page 407](#).

*RXRDY behavior in Slave mode* can be seen in [Figure 33-25 on page 422](#), [Figure 33-28 on page 425](#), [Figure 33-29 on page 426](#) and [Figure 33-30 on page 426](#).

- **TXRDY: Transmit Holding Register Ready (automatically set / reset)**

TXRDY used in Master mode:

0 = The transmit holding register has not been transferred into shift register. Set to 0 when writing into TWI\_THR register.

1 = As soon as a data byte is transferred from TWI\_THR to internal shifter or if a NACK error is detected, TXRDY is set at the same time as TXCOMP and NACK. TXRDY is also set when MSEN is set (enable TWI).

*TXRDY behavior in Master mode* can be seen in [Figure 33-8 on page 406](#).

#### TXRDY used in Slave mode:

0 = As soon as data is written in the TWI\_THR, until this data has been transmitted and acknowledged (ACK or NACK).

1 = It indicates that the TWI\_THR is empty and that data has been transmitted and acknowledged.

If TXRDY is high and if a NACK has been detected, the transmission will be stopped. Thus when TRDY = NACK = 1, the programmer must not fill TWI\_THR to avoid losing it.

*TXRDY behavior in Slave mode* can be seen in [Figure 33-24 on page 422](#), [Figure 33-27 on page 424](#), [Figure 33-29 on page 426](#) and [Figure 33-30 on page 426](#).

- **SVREAD: Slave Read (automatically set / reset)**

This bit is only used in Slave mode. When SVACC is low (no Slave access has been detected) SVREAD is irrelevant.

0 = Indicates that a write access is performed by a Master.

1 = Indicates that a read access is performed by a Master.

*SVREAD behavior* can be seen in [Figure 33-24 on page 422](#), [Figure 33-25 on page 422](#), [Figure 33-29 on page 426](#) and [Figure 33-30 on page 426](#).

- **SVACC: Slave Access (automatically set / reset)**

This bit is only used in Slave mode.

0 = TWI is not addressed. SVACC is automatically cleared after a NACK or a STOP condition is detected.

1 = Indicates that the address decoding sequence has matched (A Master has sent SADR). SVACC remains high until a NACK or a STOP condition is detected.

*SVACC behavior* can be seen in [Figure 33-24 on page 422](#), [Figure 33-25 on page 422](#), [Figure 33-29 on page 426](#) and [Figure 33-30 on page 426](#).

- **GACC: General Call Access (clear on read)**

This bit is only used in Slave mode.

0 = No General Call has been detected.

1 = A General Call has been detected. After the detection of General Call, the programmer decoded the commands that follow and the programming sequence.

*GACC behavior* can be seen in [Figure 33-26 on page 423](#).

- **OVRE: Overrun Error (clear on read)**

This bit is only used in Master mode.

0 = TWI\_RHR has not been loaded while RXRDY was set

1 = TWI\_RHR has been loaded while RXRDY was set. Reset by read in TWI\_SR when TXCOMP is set.

- **NACK: Not Acknowledged (clear on read)**

#### NACK used in Master mode:

0 = Each data byte has been correctly received by the far-end side TWI slave component.

1 = A data byte has not been acknowledged by the slave component. Set at the same time as TXCOMP.

#### NACK used in Slave Read mode:

0 = Each data byte has been correctly received by the Master.

1 = In read mode, a data byte has not been acknowledged by the Master. When NACK is set the programmer must not fill TWI\_THR even if TXRDY is set, because it means that the Master will stop the data transfer or re initiate it.

Note that in Slave Write mode all data are acknowledged by the TWI.

- **ARBLST: Arbitration Lost (clear on read)**

This bit is only used in Master mode.

0: Arbitration won.

1: Arbitration lost. Another master of the TWI bus has won the multi-master arbitration. TXCOMP is set at the same time.

- **SCLWS: Clock Wait State (automatically set / reset)**

This bit is only used in Slave mode.

0 = The clock is not stretched.

1 = The clock is stretched. TWI\_THR / TWI\_RHR buffer is not filled / emptied before the emission / reception of a new character.

*SCLWS behavior* can be seen in [Figure 33-27 on page 424](#) and [Figure 33-28 on page 425](#).

- **EOSACC: End Of Slave Access (clear on read)**

This bit is only used in Slave mode.

0 = A slave access is being performing.

1 = The Slave Access is finished. End Of Slave Access is automatically set as soon as SVACC is reset.

*EOSACC behavior* can be seen in [Figure 33-29 on page 426](#) and [Figure 33-30 on page 426](#)

- **ENDRX: End of RX buffer**

This bit is only used in Master mode.

0 = The Receive Counter Register has not reached 0 since the last write in TWI\_RCR or TWI\_RNCR.

1 = The Receive Counter Register has reached 0 since the last write in TWI\_RCR or TWI\_RNCR.

- **ENDTX: End of TX buffer**

This bit is only used in Master mode.

0 = The Transmit Counter Register has not reached 0 since the last write in TWI\_TCR or TWI\_TNCR.

1 = The Transmit Counter Register has reached 0 since the last write in TWI\_TCR or TWI\_TNCR.

- **RXBUFF: RX Buffer Full**

This bit is only used in Master mode.

0 = TWI\_RCR or TWI\_RNCR have a value other than 0.

1 = Both TWI\_RCR and TWI\_RNCR have a value of 0.

- **TXBUFE: TX Buffer Empty**

This bit is only used in Master mode.

0 = TWI\_TCR or TWI\_TNCR have a value other than 0.



1 = Both TWI\_TCR and TWI\_TNCR have a value of 0.

### 33.10.7 TWI Interrupt Enable Register

Name: TWI\_IER

Access: Write-only

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
TXBUFE	RXBUFF	ENDTX	ENDRX	EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
-	OVRE	GACC	SVACC	-	TXRDY	RXRDY	TXCOMP

- **TXCOMP: Transmission Completed Interrupt Enable**
- **RXRDY: Receive Holding Register Ready Interrupt Enable**
- **TXRDY: Transmit Holding Register Ready Interrupt Enable**
- **SVACC: Slave Access Interrupt Enable**
- **GACC: General Call Access Interrupt Enable**
- **OVRE: Overrun Error Interrupt Enable**
- **NACK: Not Acknowledge Interrupt Enable**
- **ARBLST: Arbitration Lost Interrupt Enable**
- **SCL\_WS: Clock Wait State Interrupt Enable**
- **ENDRX: End of Receive Buffer Interrupt Enable**
- **ENDTX: End of Transmit Buffer Interrupt Enable**
- **RXBUFF: Receive Buffer Full Interrupt Enable**
- **TXBUFE: Transmit Buffer Empty Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

### 33.10.8 TWI Interrupt Disable Register

Name: TWI\_IDR

Access: Write-only

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
TXBUFE	RXBUFF	ENDTX	ENDRX	EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
–	OVRE	GACC	SVACC	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP:** Transmission Completed Interrupt Disable
- **RXRDY:** Receive Holding Register Ready Interrupt Disable
- **TXRDY:** Transmit Holding Register Ready Interrupt Disable
- **SVACC:** Slave Access Interrupt Disable
- **GACC:** General Call Access Interrupt Disable
- **OVRE:** Overrun Error Interrupt Disable
- **NACK:** Not Acknowledge Interrupt Disable
- **ARBLST:** Arbitration Lost Interrupt Disable
- **SCL\_WS:** Clock Wait State Interrupt Disable
- **EOSACC:** End Of Slave Access Interrupt Disable
- **ENDRX:** End of Receive Buffer Interrupt Disable
- **ENDTX:** End of Transmit Buffer Interrupt Disable
- **RXBUFF:** Receive Buffer Full Interrupt Disable
- **TXBUFE:** Transmit Buffer Empty Interrupt Disable

0 = No effect.

1 = Disables the corresponding interrupt.



### 33.10.9 TWI Interrupt Mask Register

Name: TWI\_IMR

Access: Read-only

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
TXBUFE	RXBUFF	ENDTX	ENDRX	EOSACC	SCL_WS	ARBLST	NACK
7	6	5	4	3	2	1	0
–	OVRE	GACC	SVACC	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP: Transmission Completed Interrupt Mask**
- **RXRDY: Receive Holding Register Ready Interrupt Mask**
- **TXRDY: Transmit Holding Register Ready Interrupt Mask**
- **SVACC: Slave Access Interrupt Mask**
- **GACC: General Call Access Interrupt Mask**
- **OVRE: Overrun Error Interrupt Mask**
- **NACK: Not Acknowledge Interrupt Mask**
- **ARBLST: Arbitration Lost Interrupt Mask**
- **SCL\_WS: Clock Wait State Interrupt Mask**
- **EOSACC: End Of Slave Access Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask**
- **ENDTX: End of Transmit Buffer Interrupt Mask**
- **RXBUFF: Receive Buffer Full Interrupt Mask**
- **TXBUFE: Transmit Buffer Empty Interrupt Mask**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.





### 33.10.10 TWI Receive Holding Register

Name: TWI\_RHR

Access: Read-only

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
RXDATA							

- RXDATA: Master or Slave Receive Holding Data

### 33.10.11 TWI Transmit Holding Register

Name: TWI\_THR

Access: 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
TXDATA							

- TXDATA: Master or Slave Transmit Holding Data





## 34. Universal Synchronous/Asynchronous Receiver/Transceiver

### 34.1 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.

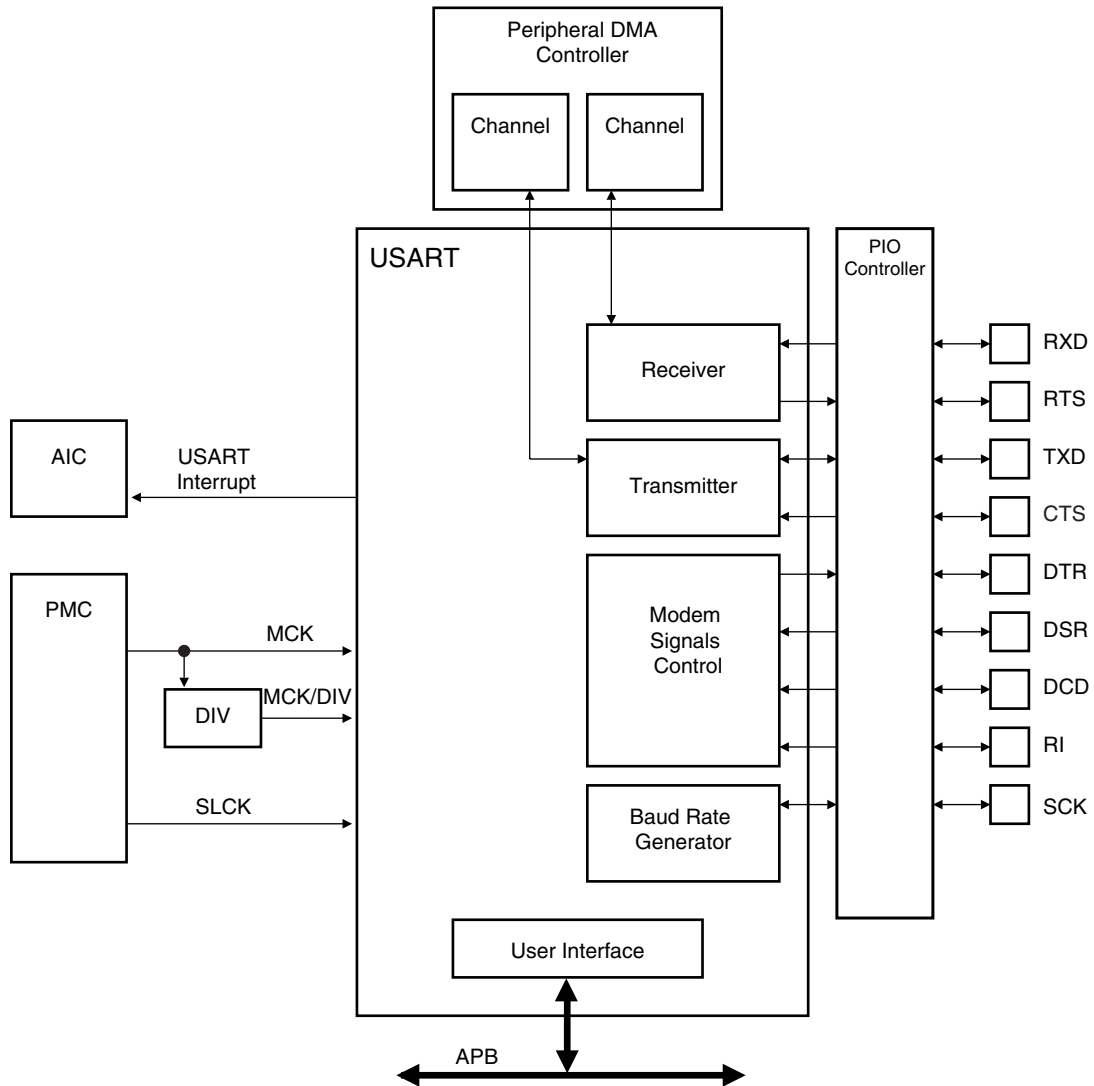
The USART features three test modes: remote loopback, local loopback and automatic echo.

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.

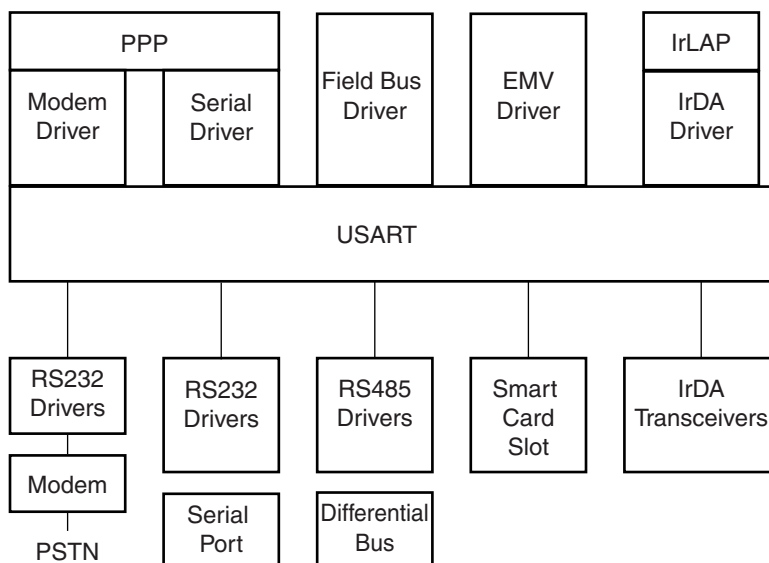
## 34.2 Block Diagram

Figure 34-1. USART Block Diagram



## 34.3 Application Block Diagram

Figure 34-2. Application Block Diagram



## 34.4 I/O Lines Description

Table 34-1. 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

## 34.5 Product Dependencies

### 34.5.1 I/O Lines

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

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. Only USART0 is fully equipped with all the modem signals. On USARTs not equipped with the corresponding pin, the associated control bits and statuses have no effect on the behavior of the USART.

### 34.5.2 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.

### 34.5.3 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.

## 34.6 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

### 34.6.1 Baud Rate Generator

The Baud Rate Generator provides the bit period clock named the Baud Rate Clock to both the receiver and the transmitter.

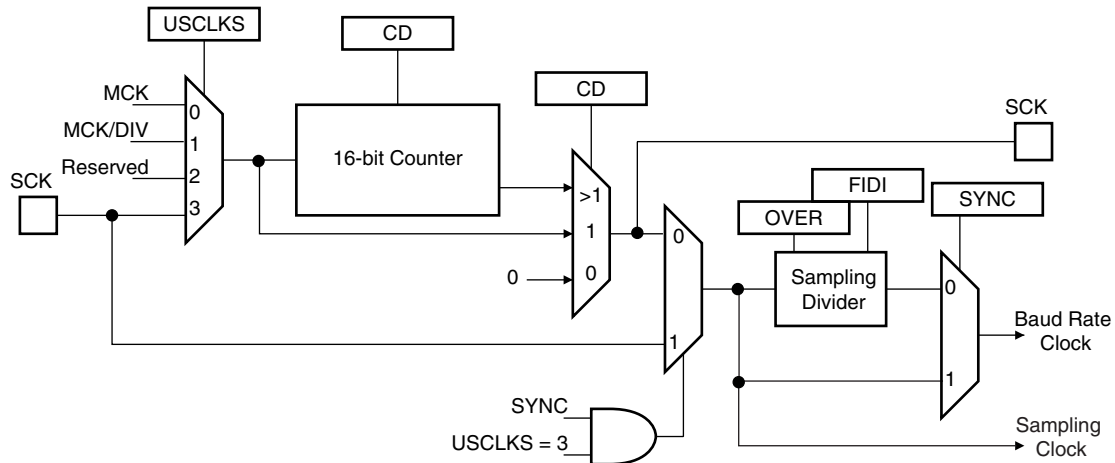
The Baud Rate Generator clock source can be selected by setting the USCLKS field in the Mode Register (US\_MR) between:

- 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 34-3.** Baud Rate Generator



### 34.6.1.1 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.

$$\text{Baudrate} = \frac{\text{SelectedClock}}{(8(2 - \text{Over})\text{CD})}$$

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 34-2 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 34-2.** 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%



**Table 34-2.** Baud Rate Example (OVER = 0) (Continued)

Source Clock	Expected Baud Rate	Calculation Result	CD	Actual Baud Rate	Error
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%.

$$Error = 1 - \left( \frac{ExpectedBaudRate}{ActualBaudRate} \right)$$

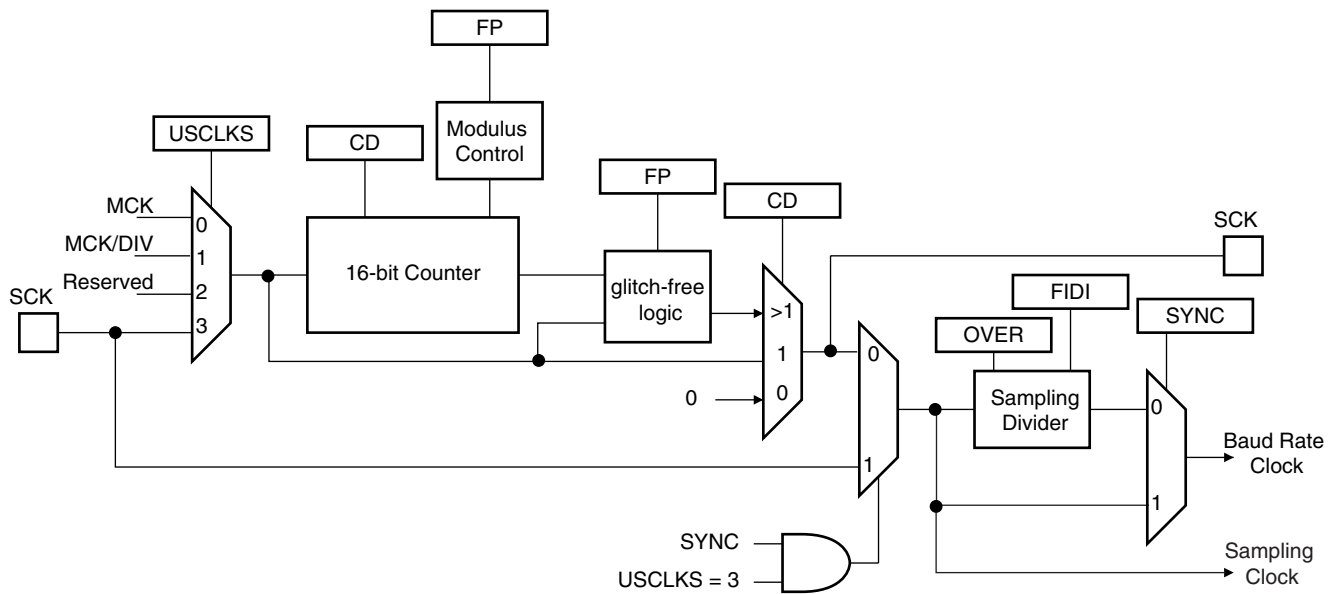
### 34.6.1.2 Fractional Baud Rate in Asynchronous Mode

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 34-4.** Fractional Baud Rate Generator



### 34.6.1.3 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.

$$\text{BaudRate} = \frac{\text{SelectedClock}}{CD}$$

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.

### 34.6.1.4 Baud Rate in ISO 7816 Mode

The ISO7816 specification defines the bit rate with the following formula:

$$B = \frac{D_i}{F_i} \times f$$

where:

- B is the bit rate
- Di is the bit-rate adjustment factor
- Fi is the clock frequency division factor
- f is the ISO7816 clock frequency (Hz)

Di is a binary value encoded on a 4-bit field, named DI, as represented in [Table 34-3](#).

**Table 34-3.** 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 34-4](#).

**Table 34-4.** 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 34-5](#) shows the resulting Fi/Di Ratio, which is the ratio between the ISO7816 clock and the baud rate clock.

**Table 34-5.** 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

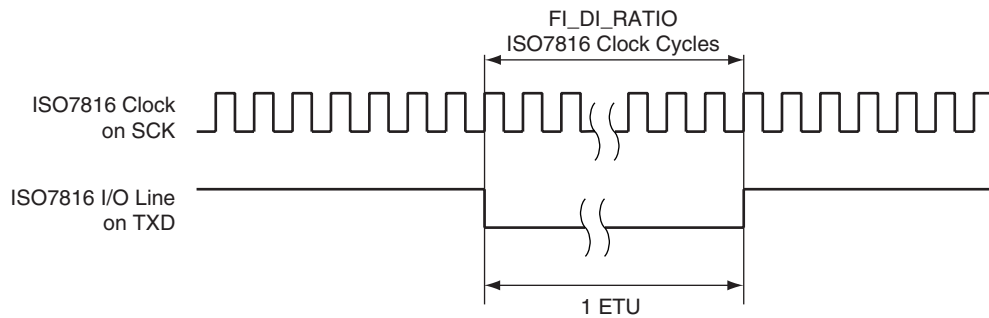
If the USART is configured in ISO7816 Mode, the clock selected by the USCLKS field in the Mode Register (US\_MR) is first divided by the value programmed in the field CD in the Baud Rate Generator Register (US\_BRGR). The resulting clock can be provided to the SCK pin to feed the smart card clock inputs. 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 34-5](#) shows the relation between the Elementary Time Unit, corresponding to a bit time, and the ISO 7816 clock.

**Figure 34-5.** Elementary Time Unit (ETU)



## 34.6.2 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.

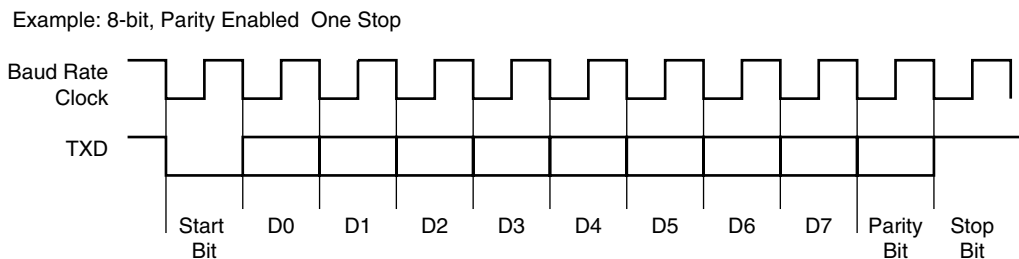
## 34.6.3 Synchronous and Asynchronous Modes

### 34.6.3.1 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.

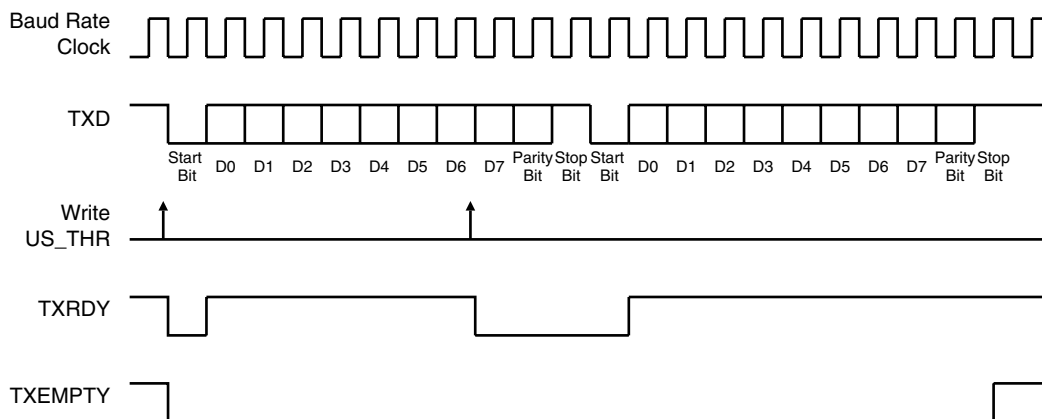
**Figure 34-6.** Character Transmit



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.

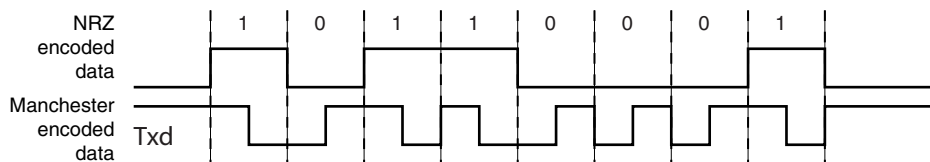
**Figure 34-7.** Transmitter Status



### 34.6.3.2 Manchester Encoder

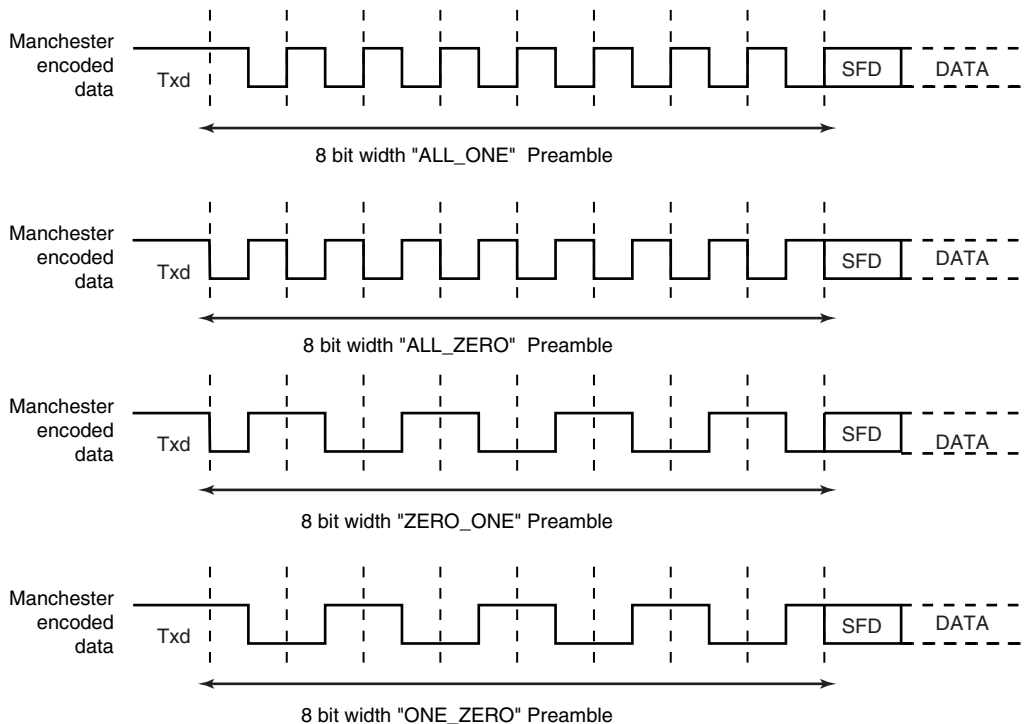
When the Manchester encoder is in use, characters transmitted through the USART are encoded based on biphase Manchester II format. To enable this mode, set the MAN field in the US\_MR register to 1. Depending on polarity configuration, a logic level (zero or one), is transmitted as a coded signal one-to-zero 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 34-8](#) illustrates this coding scheme.

**Figure 34-8.** NRZ to Manchester Encoding



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 34-9 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.

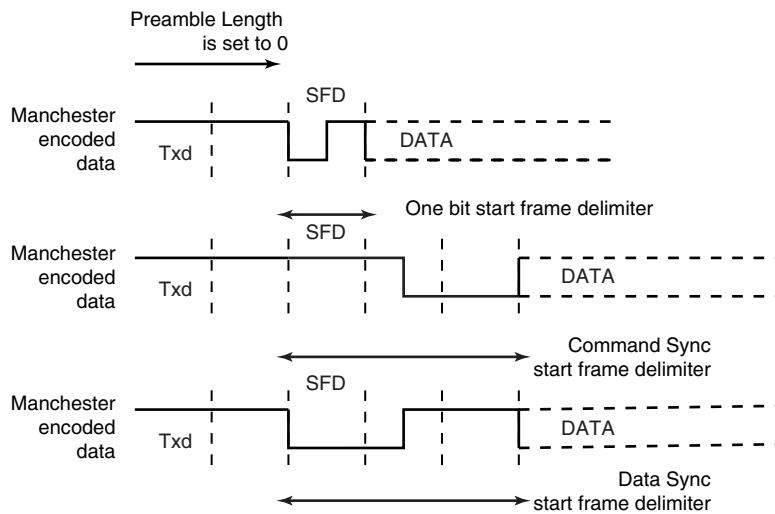
**Figure 34-9.** Preamble Patterns, Default Polarity Assumed



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 34-10 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.

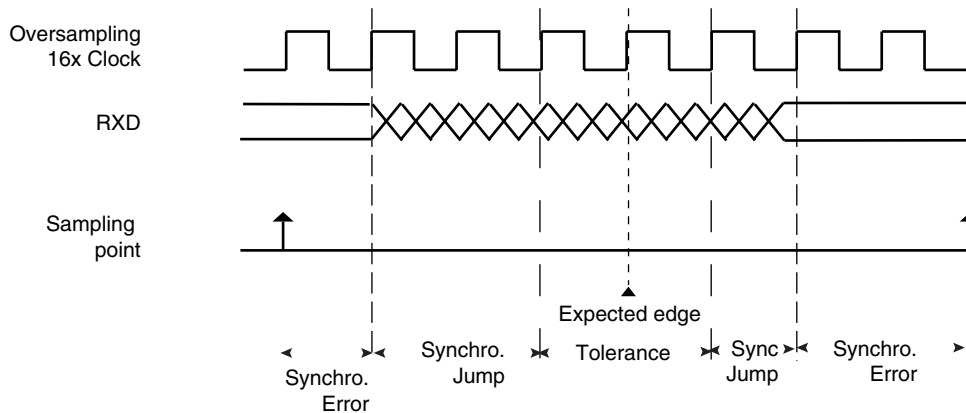
**Figure 34-10.** Start Frame Delimiter



### 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 34-11. Bit Resynchronization**



### 34.6.3.3 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.

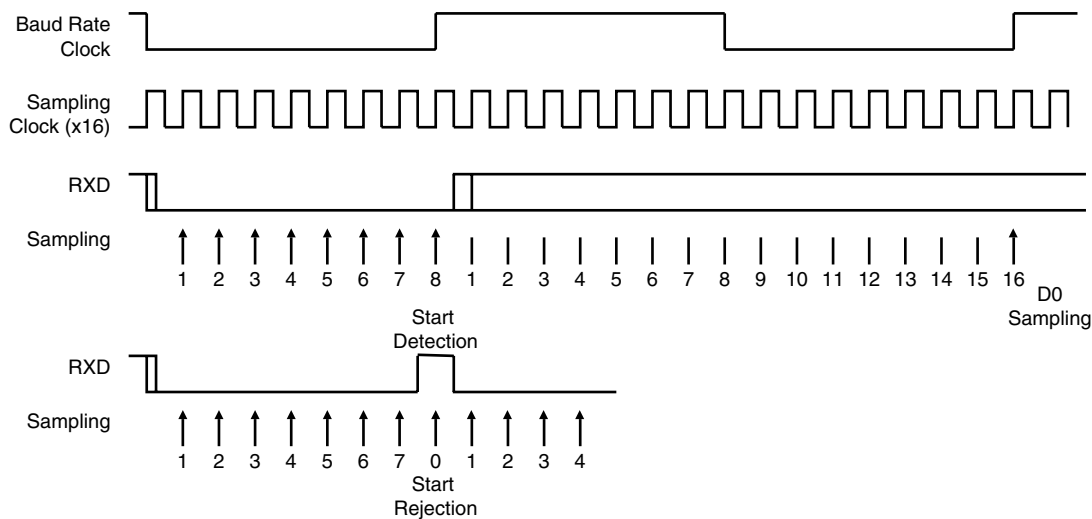
If the oversampling is 16, (OVER at 0), a start is detected at the eighth sample at 0. Then, data bits, parity bit and stop bit are sampled on each 16 sampling clock cycle. If the oversampling is 8 (OVER at 1), a start bit is detected at the fourth sample at 0. Then, data bits, parity bit and stop bit 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 34-12 and Figure 34-13 illustrate start detection and character reception when USART operates in asynchronous mode.

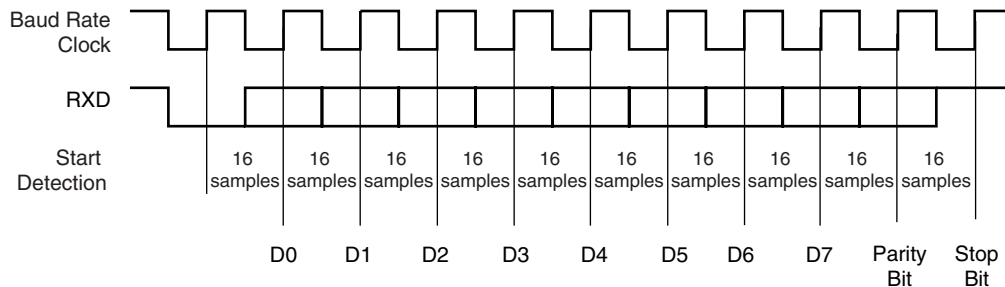


**Figure 34-12. Asynchronous Start Detection**



**Figure 34-13. Asynchronous Character Reception**

Example: 8-bit, Parity Enabled



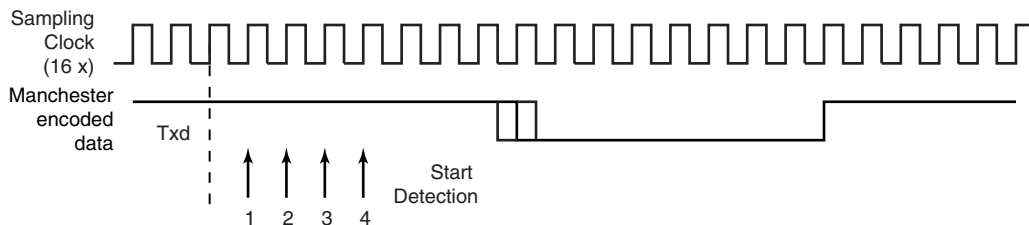
### 34.6.3.4 Manchester Decoder

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 34-9](#) 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 34-14](#). The sample pulse rejection mechanism applies.

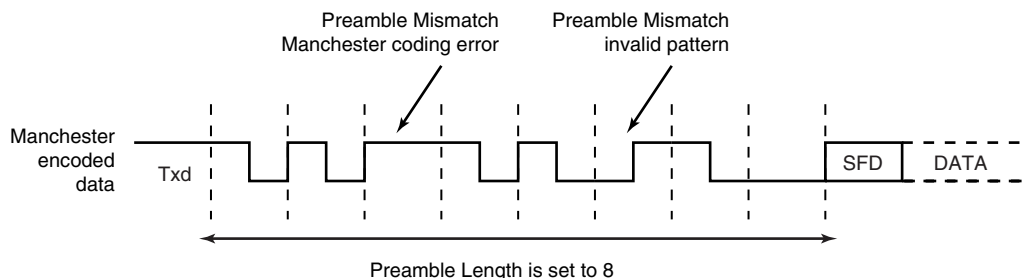
**Figure 34-14. Asynchronous Start Bit Detection**



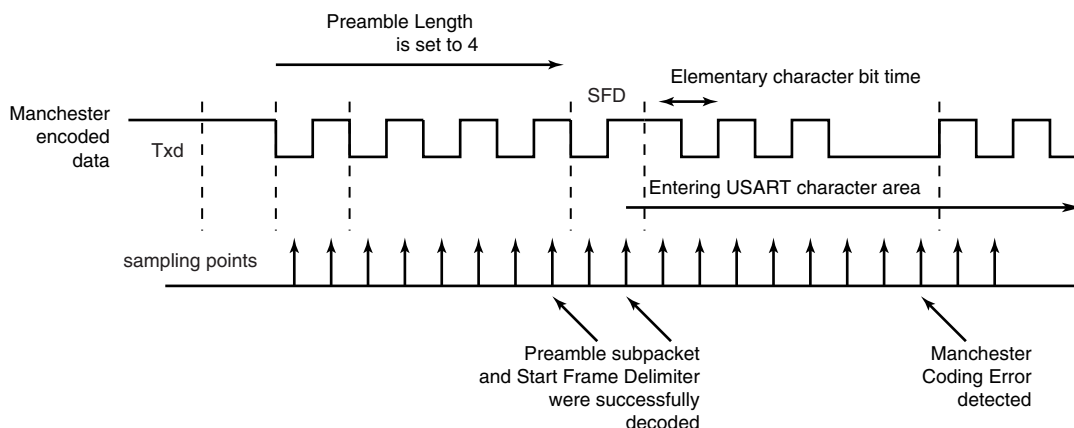
The receiver is activated and starts Preamble and Frame Delimiter detection, sampling the data at one quarter and then three quarters. If a valid preamble pattern or start frame delimiter is detected, the receiver continues decoding with the same synchronization. If the stream does not match a valid 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 34-15 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 34-16 for an example of Manchester error detection during data phase.

**Figure 34-15. Preamble Pattern Mismatch**



**Figure 34-16. Manchester Error Flag**



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.

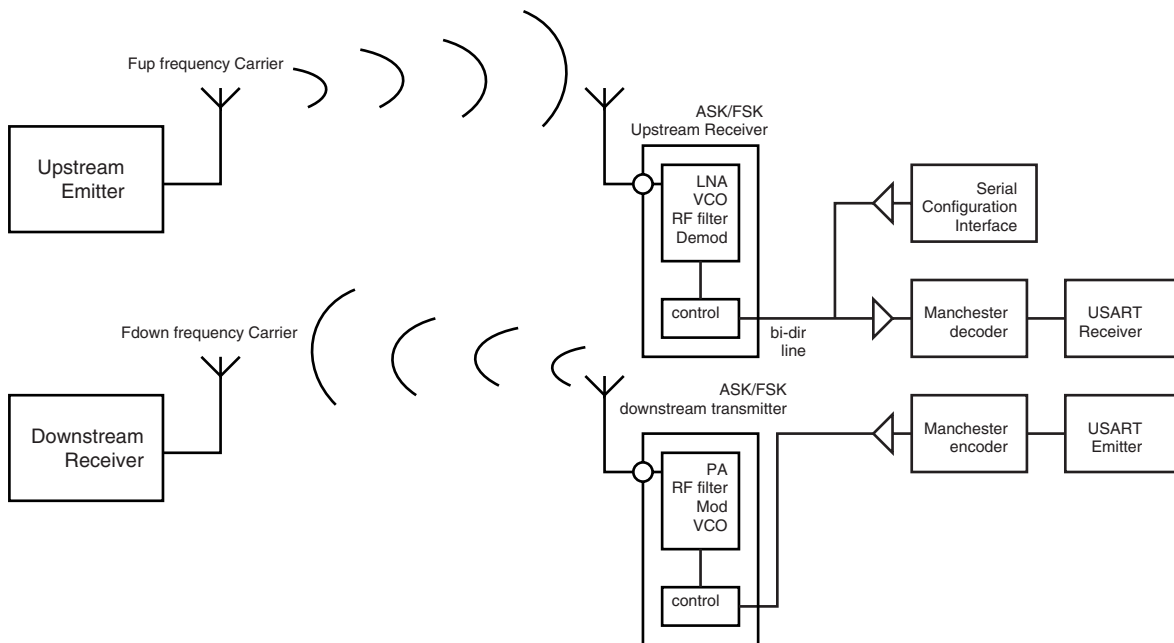
As the decoder is setup to be used in unipolar mode, the first bit of the frame has to be a zero-to-one transition.

### 34.6.3.5 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 34-17](#).

**Figure 34-17.** Manchester Encoded Characters RF Transmission

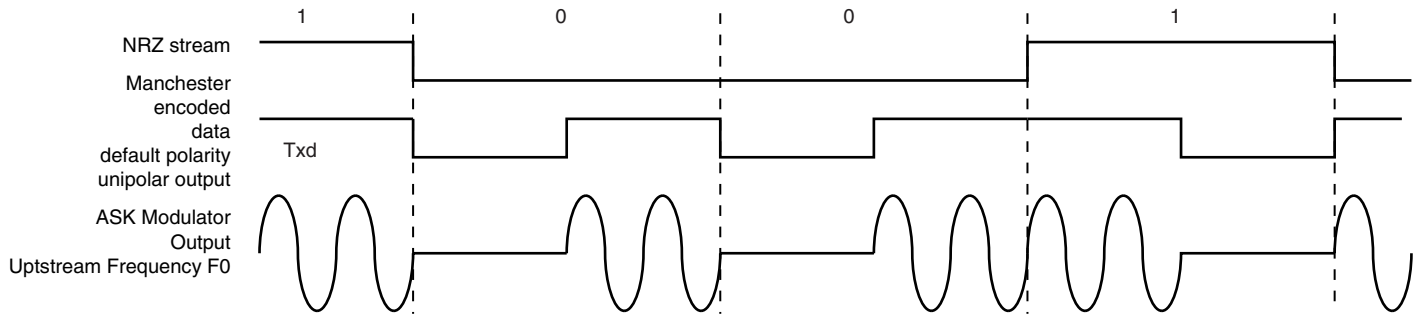


The USART module is configured as a Manchester encoder/decoder. Looking at the downstream communication channel, Manchester encoded characters are serially sent to the RF emitter. This may also include a user defined preamble and a start frame delimiter. Mostly, preamble is used in the RF receiver to distinguish between a valid data from a transmitter and signals due to noise. The Manchester stream is then modulated. See [Figure 34-18](#) 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 34-19](#).

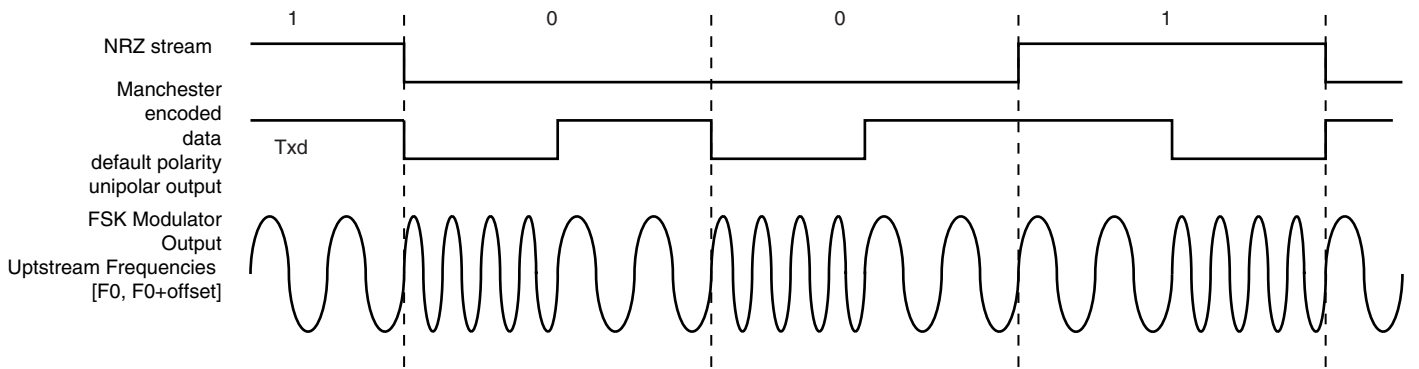
From the receiver side, another carrier frequency is used. The RF receiver performs a bit check operation examining demodulated data stream. If a valid pattern is detected, the receiver

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 34-18. ASK Modulator Output**



**Figure 34-19. FSK Modulator Output**



### 34.6.3.6 Synchronous Receiver

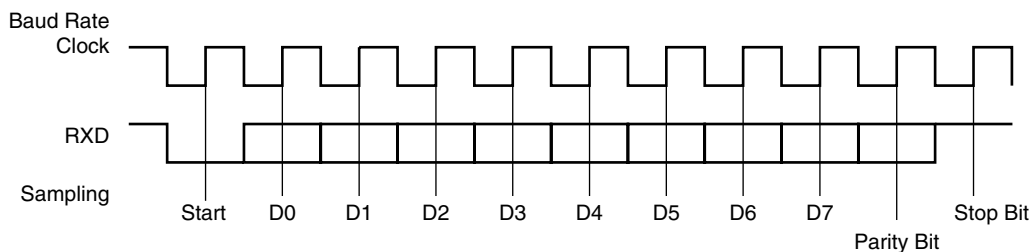
In synchronous mode (SYNC = 1), the receiver samples the RXD signal on each rising edge of the Baud Rate Clock. If a low level is detected, it is considered as a start. All data bits, the parity bit and the stop bits are sampled and the receiver waits for the next start bit. Synchronous mode operations provide a high speed transfer capability.

Configuration fields and bits are the same as in asynchronous mode.

Figure 34-20 illustrates a character reception in synchronous mode.

**Figure 34-20. Synchronous Mode Character Reception**

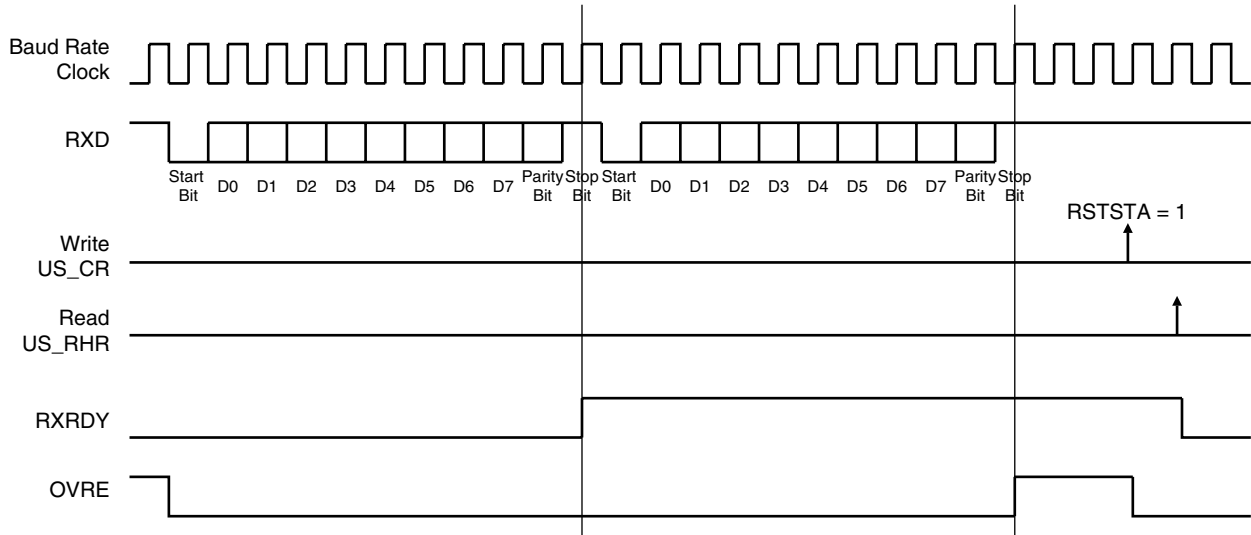
Example: 8-bit, Parity Enabled 1 Stop



## 34.6.3.7 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 34-21.** Receiver Status



## 34.6.3.8 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 463](#). 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.

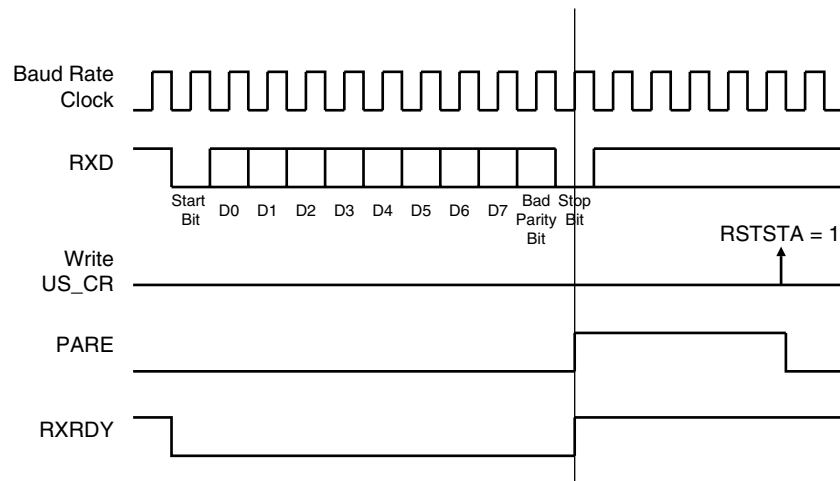
[Table 34-6](#) shows an example of the parity bit for the character 0x41 (character ASCII “A”) depending on the configuration of the USART. Because there are two bits at 1, 1 bit is added when a parity is odd, or 0 is added when a parity is even.

**Table 34-6.** 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 34-22](#) illustrates the parity bit status setting and clearing.

**Figure 34-22.** Parity Error



### 34.6.3.9 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.

### 34.6.3.10 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 34-23](#), 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 34-23.** Timeguard Operations

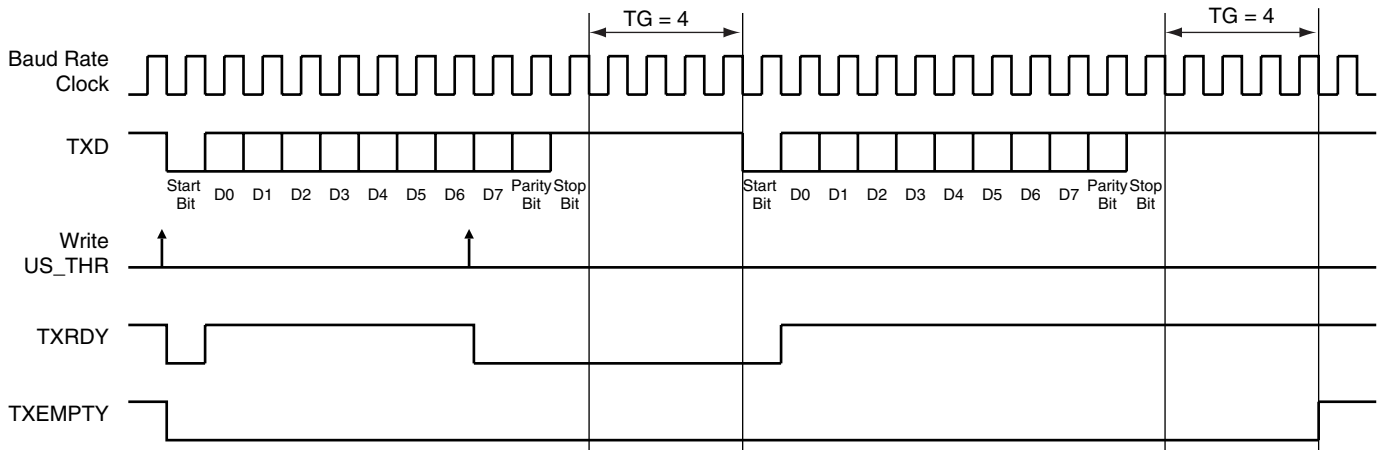


Table 34-7 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the Baud Rate.

**Table 34-7.** 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

### 34.6.3.11 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 34-24 shows the block diagram of the Receiver Time-out feature.

**Figure 34-24.** Receiver Time-out Block Diagram

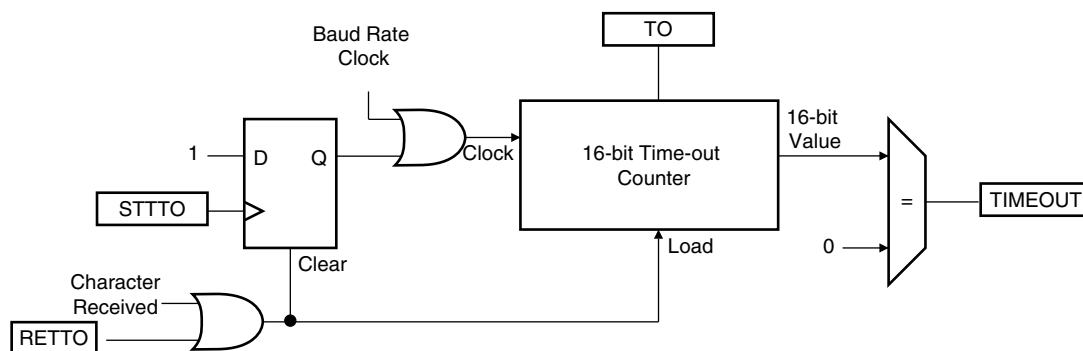


Table 34-8 gives the maximum time-out period for some standard baud rates.

**Table 34-8.** 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 34-8.** Maximum Time-out Period (Continued)

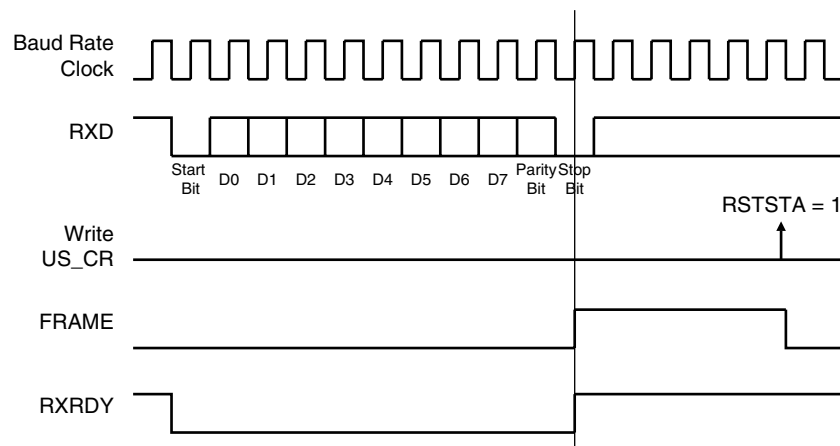
Baud Rate	Bit Time	Time-out
56000	18	1 170
57600	17	1 138
200000	5	328

### 34.6.3.12 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 34-25.** Framing Error Status



### 34.6.3.13 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 STTBK bit at 1. This can be performed at any time, either while the transmitter is empty (no character in either the Shift Register or in US\_THR) or when a character is being transmitted. If a break is requested while a character is being shifted out, the character is first completed before the TXD line is held low.

Once STTBK command is requested further STTBK commands are ignored until the end of the break is completed.

The break condition is removed by writing US\_CR with the STPBK bit at 1. If the STPBK is requested before the end of the minimum break duration (one character, including start, data, parity and stop bits), the transmitter ensures that the break condition completes.

The transmitter considers the break as though it is a character, i.e. the 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.

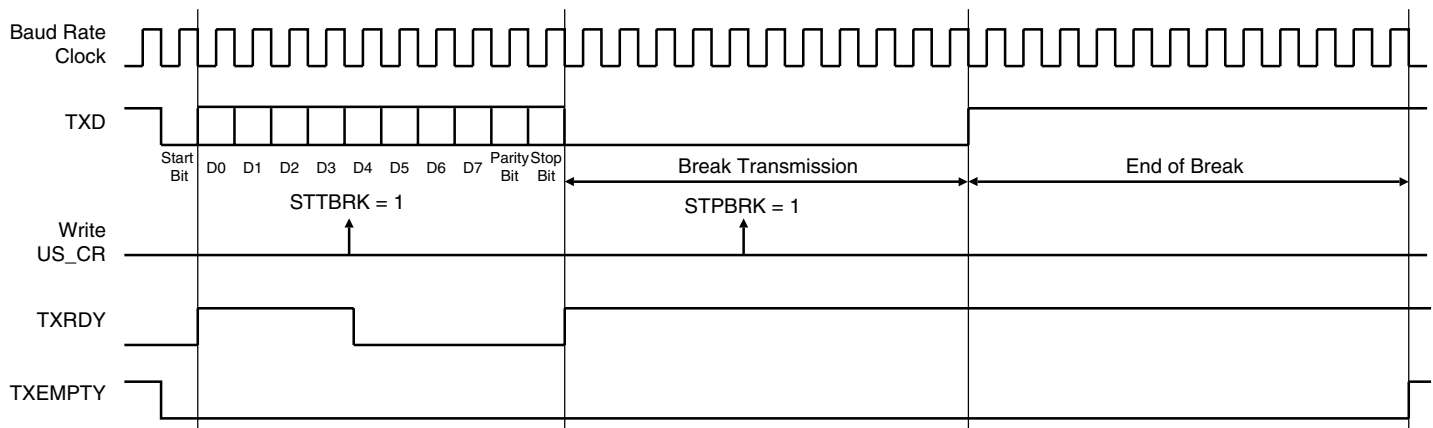
Writing US\_CR with the both STTBRK and STPBRK bits at 1 can lead to an unpredictable result. All STPBRK commands requested without a previous STTBRK command are ignored. A byte written into the Transmit Holding Register while a break is pending, but not started, is ignored.

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 34-26 illustrates the effect of both the Start Break (STTBRK) and Stop Break (STPBRK) commands on the TXD line.

**Figure 34-26.** Break Transmission



### 34.6.3.14 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.

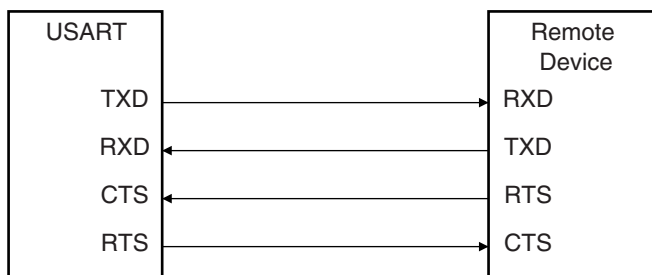
When the low stop bit is detected, the receiver asserts the RXBRK bit in US\_CSR. This bit may be cleared by writing the Control Register (US\_CR) with the bit RSTSTA at 1.

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.

### 34.6.3.15 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 34-27.

**Figure 34-27.** 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 34-28 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 34-28.** Receiver Behavior when Operating with Hardware Handshaking

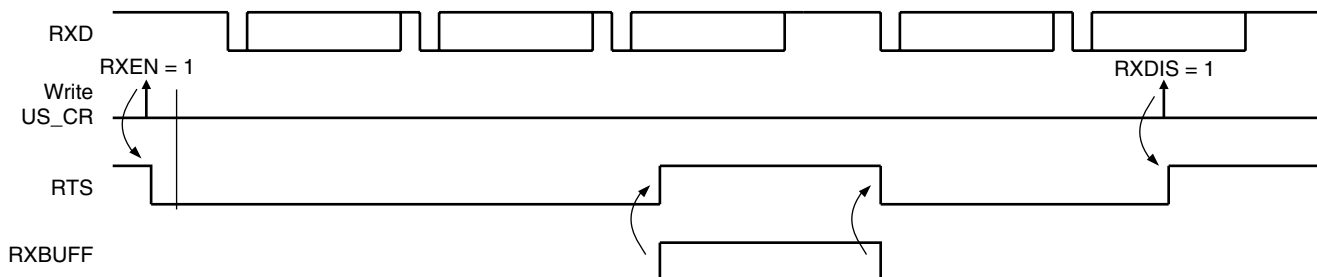


Figure 34-29 shows how the transmitter operates if hardware handshaking is enabled. The CTS pin disables the transmitter. If a character is being processing, the transmitter is disabled only after the completion of the current character and transmission of the next character happens as soon as the pin CTS falls.

**Figure 34-29.** Transmitter Behavior when Operating with Hardware Handshaking



## 34.6.4 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.

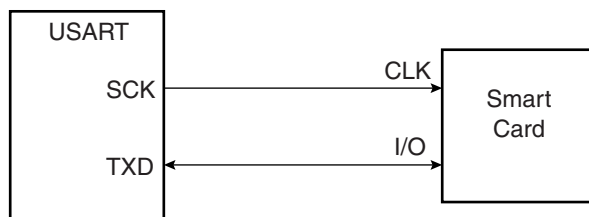
Setting the USART in ISO7816 mode is performed by writing the USART\_MODE field in the Mode Register (US\_MR) to the value 0x4 for protocol T = 0 and to the value 0x5 for protocol T = 1.

### 34.6.4.1 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 447](#)).

The USART connects to a smart card as shown in [Figure 34-30](#). 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 34-30.** 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 481](#) and [“PAR: Parity Type” on page 482](#).

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).

### 34.6.4.2 Protocol T = 0

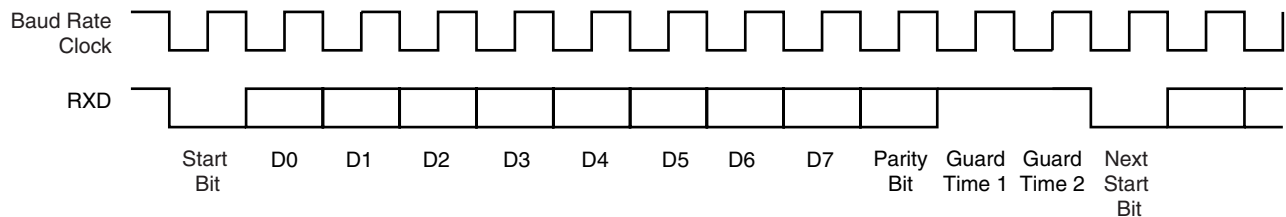
In T = 0 protocol, a character is made up of one start bit, eight data bits, one parity bit and one guard time, which lasts two bit times. The transmitter shifts out the bits and does not drive the I/O line during the guard time.

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 34-31](#).

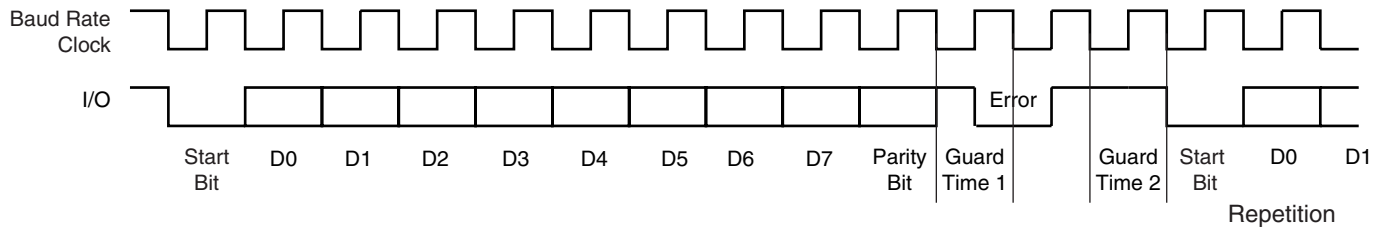
If a parity error is detected by the receiver, it drives the I/O line at 0 during the guard time, as shown in Figure 34-32. This error bit is also named NACK, for Non Acknowledge. In this case, the character lasts 1 bit time more, as the guard time length is the same and is added to the error bit time 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 34-31.** T = 0 Protocol without Parity Error



**Figure 34-32.** T = 0 Protocol with Parity Error



### Receive Error Counter

The USART receiver also records the total number of errors. This can be read in the Number of Error (US\_NER) register. The NB\_ERRORS field can record up to 255 errors. Reading US\_NER automatically clears the NB\_ERRORS field.

### Receive NACK Inhibit

The USART can also be configured to inhibit an error. This can be achieved by setting the INACK bit in the Mode Register (US\_MR). If INACK is at 1, no error signal is driven on the I/O line even if a parity bit is detected, but the INACK bit is set in the Status Register (US\_SR). The INACK bit can be cleared by writing the Control Register (US\_CR) with the RSTNACK bit at 1.

Moreover, if INACK is set, the erroneous received character is stored in the Receive Holding Register, as if no error occurred. However, the RXRDY bit does not raise.

### Transmit Character Repetition

When the USART is transmitting a character and gets a NACK, it can automatically repeat the character before moving on to the next one. Repetition is enabled by writing the MAX\_ITERATION field in the Mode Register (US\_MR) at a value higher than 0. Each character can be transmitted up to eight times; the first transmission plus seven repetitions.

If MAX\_ITERATION does not equal zero, the USART repeats the character as many times as the value loaded in MAX\_ITERATION.

When the USART repetition number reaches MAX\_ITERATION, the ITERATION bit is set in the Channel Status Register (US\_CSR). If the repetition of the character is acknowledged by the receiver, the repetitions are stopped and the iteration counter is cleared.

The ITERATION bit in US\_CSR can be cleared by writing the Control Register with the RSIT bit at 1.

### Disable Successive Receive NACK

The receiver can limit the number of successive NACKs sent back to the remote transmitter. This is programmed by setting the bit DSNACK in the Mode Register (US\_MR). The maximum number of NACK transmitted is programmed in the MAX\_ITERATION field. As soon as MAX\_ITERATION is reached, the character is considered as correct, an acknowledge is sent on the line and the ITERATION bit in the Channel Status Register is set.

### 34.6.4.3 Protocol T = 1

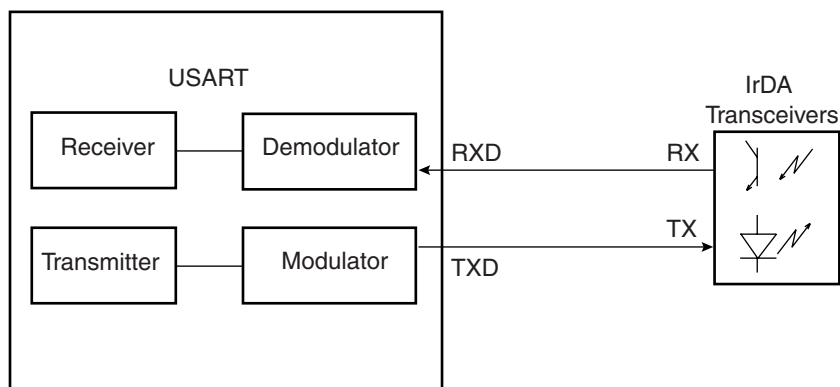
When operating in ISO7816 protocol T = 1, the transmission is similar to an asynchronous format with only one stop bit. The parity is generated when transmitting and checked when receiving. Parity error detection sets the PARE bit in the Channel Status Register (US\_CSR).

### 34.6.5 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 34-33. 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 34-33.** Connection to IrDA Transceivers



The receiver and the transmitter must be enabled or disabled according to the direction of the transmission to be managed.

## 34.6.5.1 IrDA Modulation

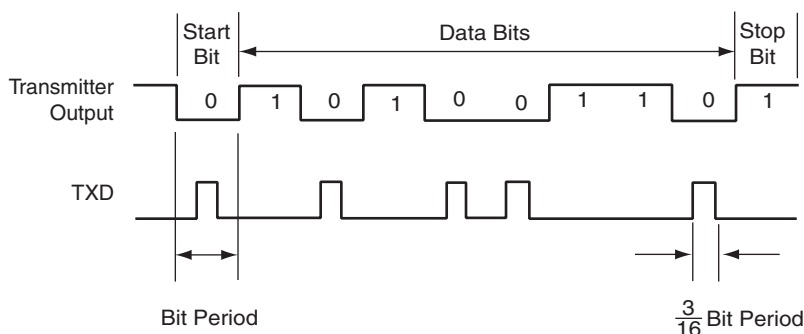
For baud rates up to and including 115.2 Kbits/sec, the RZI modulation scheme is used. “0” is represented by a light pulse of 3/16th of a bit time. Some examples of signal pulse duration are shown in [Table 34-9](#).

**Table 34-9.** IrDA Pulse Duration

Baud Rate	Pulse Duration (3/16)
2.4 Kb/s	78.13 $\mu$ s
9.6 Kb/s	19.53 $\mu$ s
19.2 Kb/s	9.77 $\mu$ s
38.4 Kb/s	4.88 $\mu$ s
57.6 Kb/s	3.26 $\mu$ s
115.2 Kb/s	1.63 $\mu$ s

[Figure 34-34](#) shows an example of character transmission.

**Figure 34-34.** IrDA Modulation



## 34.6.5.2 IrDA Baud Rate

[Table 34-10](#) gives some examples of CD values, baud rate error and pulse duration. Note that the requirement on the maximum acceptable error of  $\pm 1.87\%$  must be met.

**Table 34-10.** 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 34-10.** IrDA Baud Rate Error (Continued)

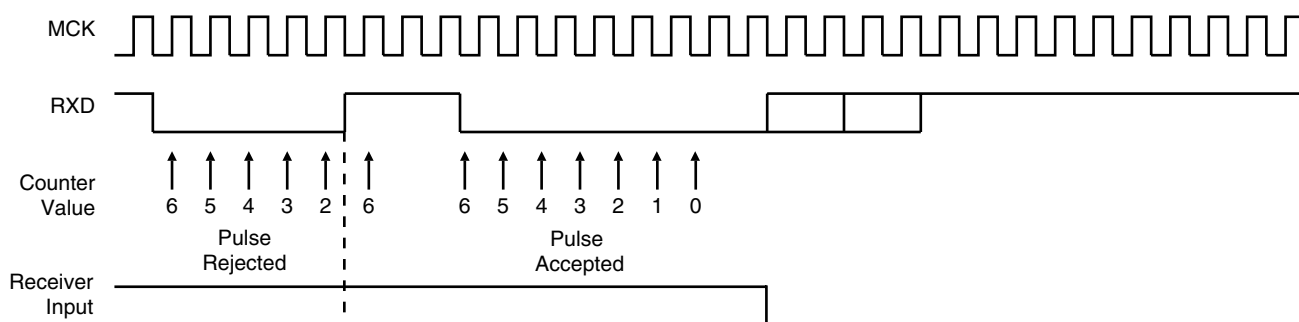
Peripheral Clock	Baud Rate	CD	Baud Rate Error	Pulse Time
32 768 000	38 400	53	0.63%	4.88
40 000 000	38 400	65	0.16%	4.88
3 686 400	19 200	12	0.00%	9.77
20 000 000	19 200	65	0.16%	9.77
32 768 000	19 200	107	0.31%	9.77
40 000 000	19 200	130	0.16%	9.77
3 686 400	9 600	24	0.00%	19.53
20 000 000	9 600	130	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

### 34.6.5.3 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 34-35 illustrates the operations of the IrDA demodulator.

**Figure 34-35.** IrDA Demodulator Operations

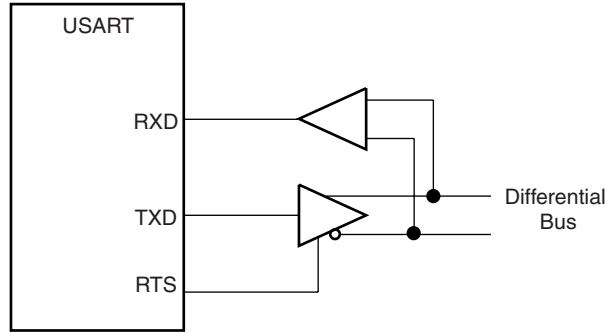


As the IrDA mode uses the same logic as the ISO7816, note that the FI\_DI\_RATIO field in US\_FIDI must be set to a value higher than 0 in order to assure IrDA communications operate correctly.

## 34.6.6 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 34-36](#).

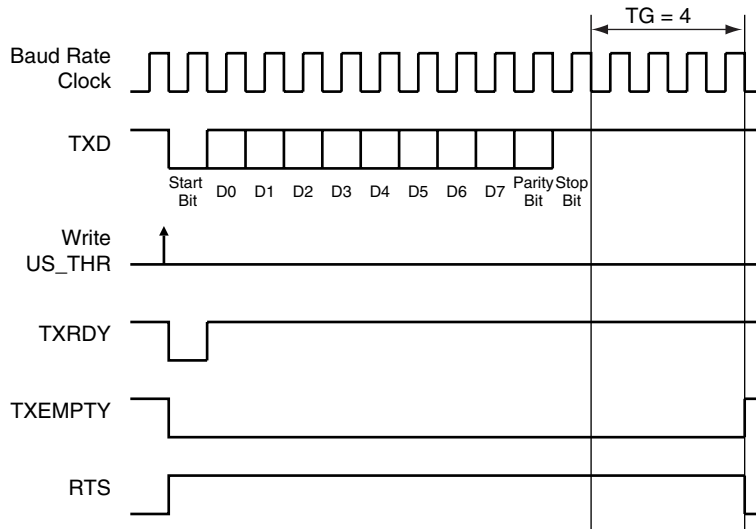
**Figure 34-36.** Typical Connection to a RS485 Bus



The USART is set in RS485 mode by programming the USART\_MODE field in the Mode Register (US\_MR) to the value 0x1.

The RTS pin is at a level inverse 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 34-37](#) gives an example of the RTS waveform during a character transmission when the timeguard is enabled.

**Figure 34-37.** Example of RTS Drive with Timeguard



## 34.6.7 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 34-11 gives the correspondence of the USART signals with modem connection standards.

**Table 34-11.** 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.

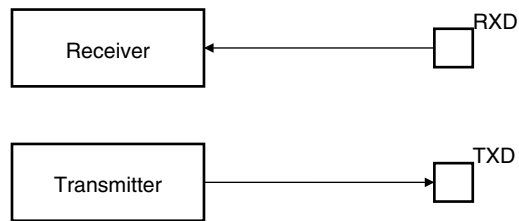
## 34.6.8 Test Modes

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.

### 34.6.8.1 Normal Mode

Normal mode connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

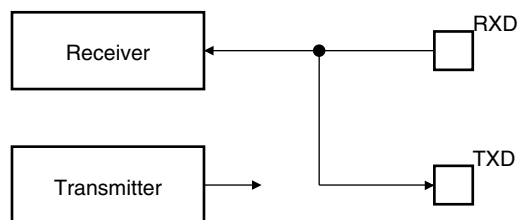
**Figure 34-38.** Normal Mode Configuration



### 34.6.8.2 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 34-39](#). 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.

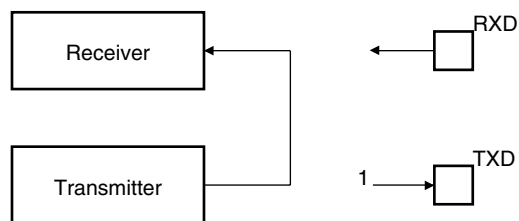
**Figure 34-39.** Automatic Echo Mode Configuration



### 34.6.8.3 Local Loopback Mode

Local loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in [Figure 34-40](#). 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.

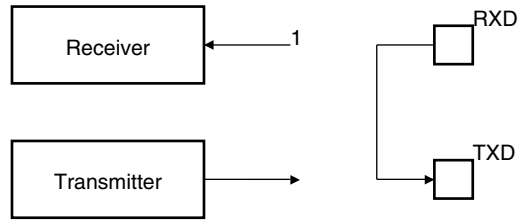
**Figure 34-40.** Local Loopback Mode Configuration



### 34.6.8.4 Remote Loopback Mode

Remote loopback mode directly connects the RXD pin to the TXD pin, as shown in [Figure 34-41](#). The transmitter and the receiver are disabled and have no effect. This mode allows bit-by-bit retransmission.

Figure 34-41. Remote Loopback Mode Configuration



## 34.7 USART User Interface

**Table 34-12.** 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 Decoder Register	US_MAN	Read/Write	0x30011004
0x5C - 0xFC	Reserved	–	–	–
0x100 - 0x128	Reserved for PDC Registers	–	–	–

## 34.7.1 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.



## 34.7.2 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

USART_MODE				Mode of the USART
0	0	0	0	Normal
0	0	0	1	RS485
0	0	1	0	Hardware Handshaking
0	0	1	1	Modem
0	1	0	0	ISO7816 Protocol: T = 0
0	1	0	1	Reserved
0	1	1	0	ISO7816 Protocol: T = 1
0	1	1	1	Reserved
1	0	0	0	IrDA
1	1	x	x	Reserved

### • USCLKS: Clock Selection

USCLKS		Selected Clock
0	0	MCK
0	1	MCK/DIV (DIV = 8)
1	0	Reserved
1	1	SCK

### • CHRL: Character Length.

CHRL		Character Length
0	0	5 bits

0	1	6 bits
1	0	7 bits
1	1	8 bits

- **SYNC: Synchronous Mode Select**

0: USART operates in Asynchronous Mode.

1: USART operates in Synchronous Mode.

- **PAR: Parity Type**

PAR			Parity Type
0	0	0	Even parity
0	0	1	Odd parity
0	1	0	Parity forced to 0 (Space)
0	1	1	Parity forced to 1 (Mark)
1	0	x	No parity
1	1	x	Multidrop mode

- **NBSTOP: Number of Stop Bits**

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

- **CHMODE: Channel Mode**

CHMODE		Mode Description
0	0	Normal Mode
0	1	Automatic Echo. Receiver input is connected to the TXD pin.
1	0	Local Loopback. Transmitter output is connected to the Receiver Input..
1	1	Remote Loopback. RXD pin is internally connected to the TXD pin.

- **MSBF: Bit Order**

0: Least Significant Bit is sent/received first.

1: Most Significant Bit is sent/received first.

- **MODE9: 9-bit Character Length**

0: CHRL defines character length.

1: 9-bit character length.

- **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.

## 34.7.3 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.

## 34.7.4 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.

## 34.7.5 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.

## 34.7.6 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.

## 34.7.7 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.

### 34.7.8 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.

## 34.7.9 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**

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  $FP \times 1/8$ .

### 34.7.10 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.



## 34.7.11 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.

### 34.7.12 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.

### 34.7.13 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.

## 34.7.14 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**

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**

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.

## 34.7.15 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.



## 35. Serial Synchronous Controller (SSC)

### 35.1 Description

The Atmel Synchronous Serial Controller (SSC) provides a synchronous communication link with external devices. It supports many serial synchronous communication protocols generally used in audio and telecom applications such as I2S, Short Frame Sync, Long Frame Sync, etc.

The SSC contains an independent receiver and transmitter and a common clock divider. The receiver and the transmitter each interface with three signals: the TD/RD signal for data, the TK/RK signal for the clock and the TF/RF signal for the Frame Sync. The transfers can be programmed to start automatically or on different events detected on the Frame Sync signal.

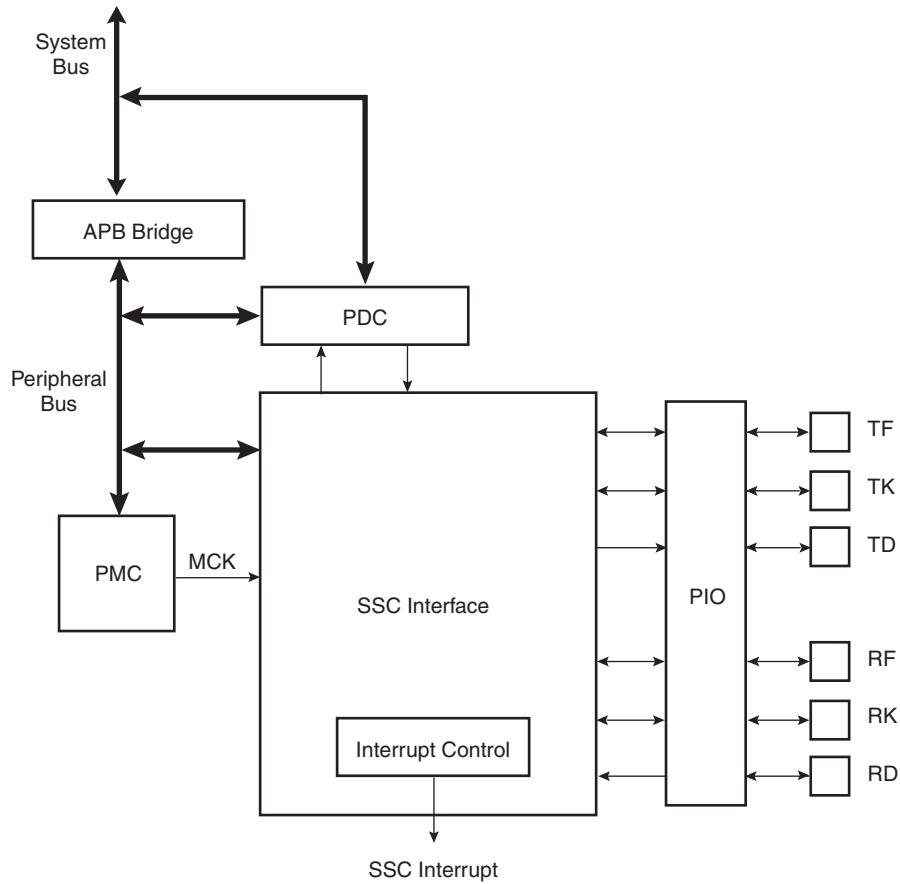
The SSC's high-level of programmability and its two dedicated PDC channels of up to 32 bits permit a continuous high bit rate data transfer without processor intervention.

Featuring connection to two PDC channels, the SSC permits interfacing with low processor overhead to the following:

- CODEC's in master or slave mode
- DAC through dedicated serial interface, particularly I2S
- Magnetic card reader

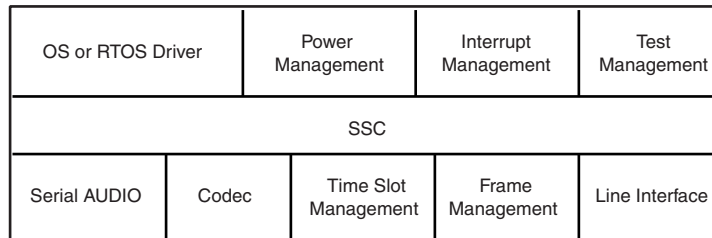
## 35.2 Block Diagram

Figure 35-1. Block Diagram



## 35.3 Application Block Diagram

Figure 35-2. Application Block Diagram





## 35.4 Pin Name List

Table 35-1. I/O Lines Description

Pin Name	Pin Description	Type
RF	Receiver Frame Synchro	Input/Output
RK	Receiver Clock	Input/Output
RD	Receiver Data	Input
TF	Transmitter Frame Synchro	Input/Output
TK	Transmitter Clock	Input/Output
TD	Transmitter Data	Output

## 35.5 Product Dependencies

### 35.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines.

Before using the SSC receiver, the PIO controller must be configured to dedicate the SSC receiver I/O lines to the SSC peripheral mode.

Before using the SSC transmitter, the PIO controller must be configured to dedicate the SSC transmitter I/O lines to the SSC peripheral mode.

### 35.5.2 Power Management

The SSC is not continuously clocked. The SSC interface may be clocked through the Power Management Controller (PMC), therefore the programmer must first configure the PMC to enable the SSC clock.

### 35.5.3 Interrupt

The SSC interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling interrupts requires programming the AIC before configuring the SSC.

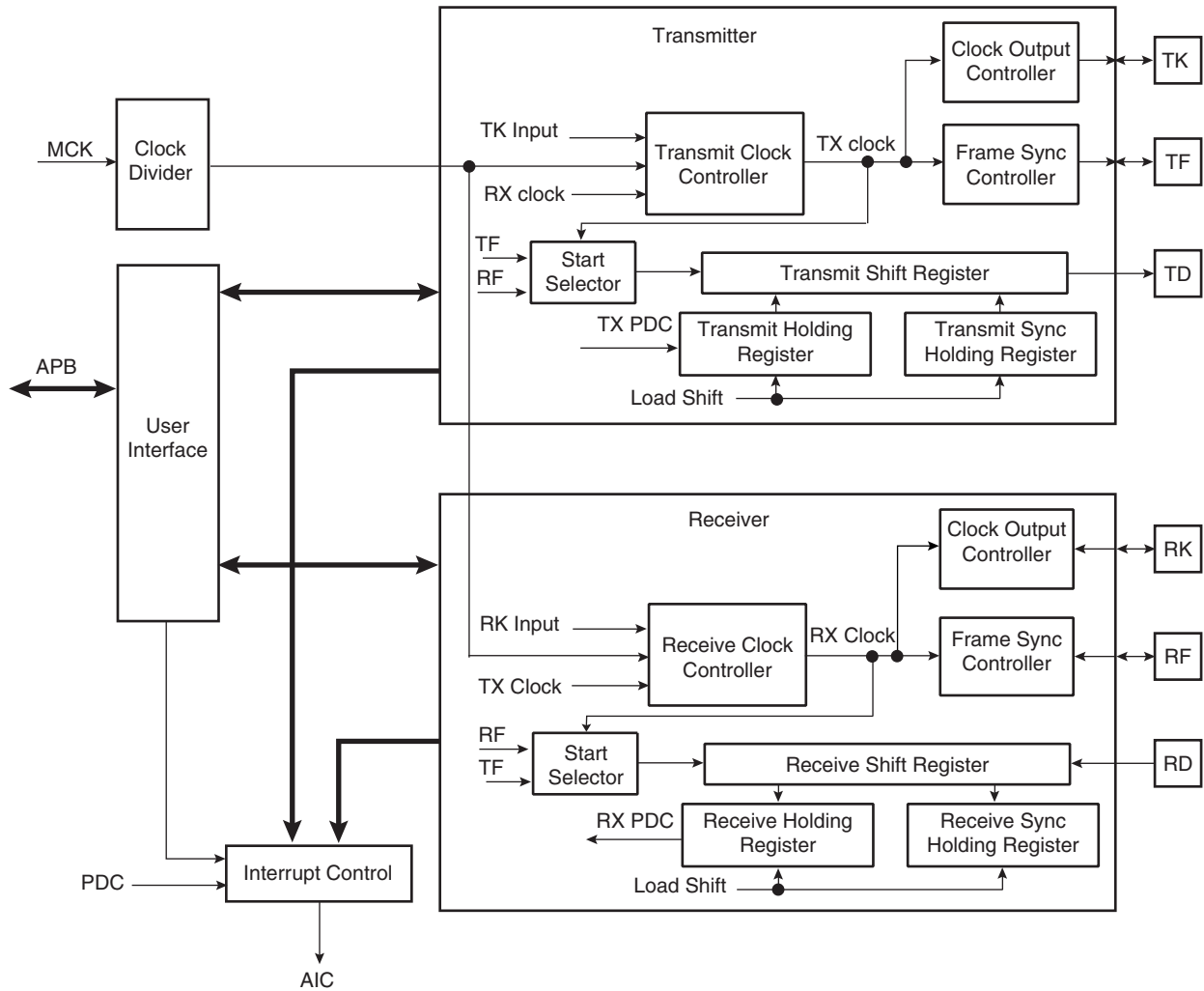
All SSC interrupts can be enabled/disabled configuring the SSC Interrupt mask register. Each pending and unmasked SSC interrupt will assert the SSC interrupt line. The SSC interrupt service routine can get the interrupt origin by reading the SSC interrupt status register.

## 35.6 Functional Description

This chapter contains the functional description of the following: SSC Functional Block, Clock Management, Data format, Start, Transmitter, Receiver and Frame Sync.

The receiver and transmitter operate separately. However, they can work synchronously by programming the receiver to use the transmit clock and/or to start a data transfer when transmission starts. Alternatively, this can be done by programming the transmitter to use the receive clock and/or to start a data transfer when reception starts. The transmitter and the receiver can be programmed to operate with the clock signals provided on either the TK or RK pins. This allows the SSC to support many slave-mode data transfers. The maximum clock speed allowed on the TK and RK pins is the master clock divided by 2.

**Figure 35-3. SSC Functional Block Diagram**



### 35.6.1 Clock Management

The transmitter clock can be generated by:

- an external clock received on the TK I/O pad
- the receiver clock
- the internal clock divider

The receiver clock can be generated by:

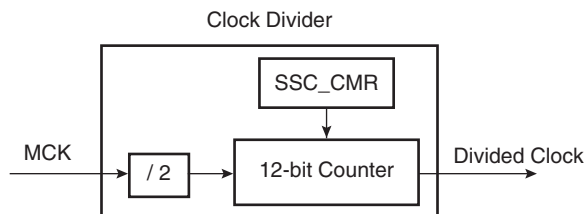
- an external clock received on the RK I/O pad
- the transmitter clock
- the internal clock divider

Furthermore, the transmitter block can generate an external clock on the TK I/O pad, and the receiver block can generate an external clock on the RK I/O pad.

This allows the SSC to support many Master and Slave Mode data transfers.

## 35.6.1.1 Clock Divider

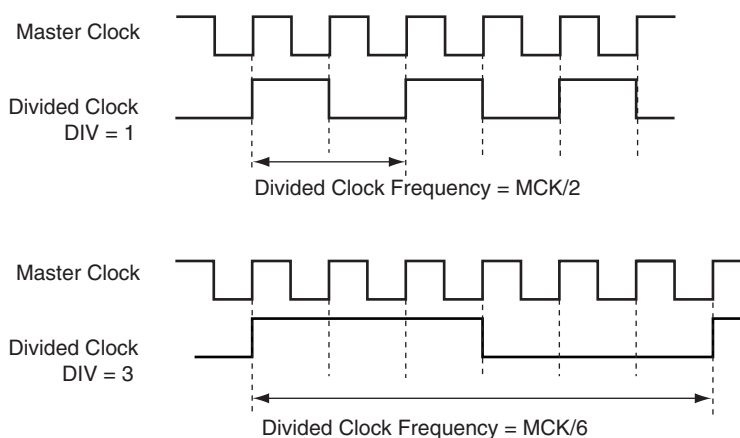
**Figure 35-4.** Divided Clock Block Diagram



The Master Clock divider is determined by the 12-bit field DIV counter and comparator (so its maximal value is 4095) in the Clock Mode Register SSC\_CMCR, allowing a Master Clock division by up to 8190. The Divided Clock is provided to both the Receiver and Transmitter. When this field is programmed to 0, the Clock Divider is not used and remains inactive.

When DIV is set to a value equal to or greater than 1, the Divided Clock has a frequency of Master Clock divided by 2 times DIV. Each level of the Divided Clock has a duration of the Master Clock multiplied by DIV. This ensures a 50% duty cycle for the Divided Clock regardless of whether the DIV value is even or odd.

**Figure 35-5.** Divided Clock Generation



**Table 35-2.**

Maximum	Minimum
MCK / 2	MCK / 8190

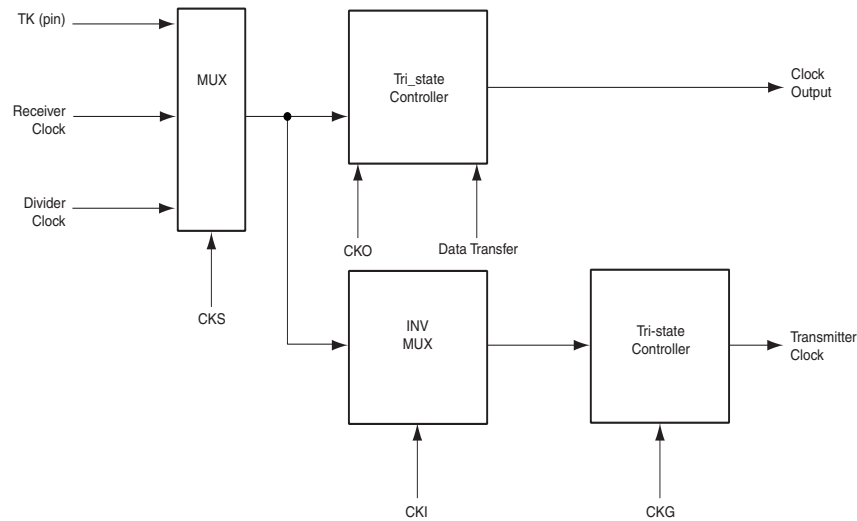
## 35.6.1.2 Transmitter Clock Management

The transmitter clock is generated from the receiver clock or the divider clock or an external clock scanned on the TK I/O pad. The transmitter clock is selected by the CKS field in SSC\_TCMR (Transmit Clock Mode Register). Transmit Clock can be inverted independently by the CKI bits in SSC\_TCMR.

The transmitter can also drive the TK I/O pad continuously or be limited to the actual data transfer. The clock output is configured by the SSC\_TCMR register. The Transmit Clock Inversion (CKI) bits have no effect on the clock outputs. Programming the TCMR register to select TK pin

(CKS field) and at the same time Continuous Transmit Clock (CKO field) might lead to unpredictable results.

**Figure 35-6.** Transmitter Clock Management

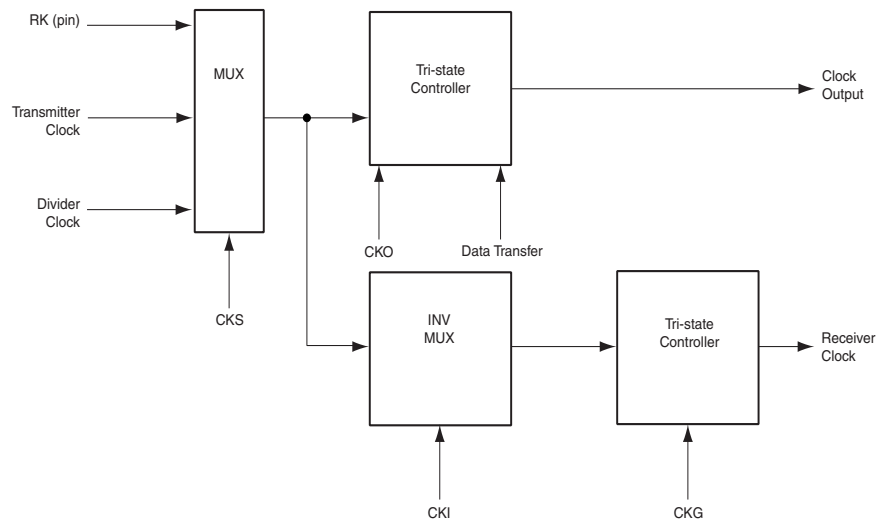


### 35.6.1.3 Receiver Clock Management

The receiver clock is generated from the transmitter clock or the divider clock or an external clock scanned on the RK I/O pad. The Receive Clock is selected by the CKS field in SSC\_RCMR (Receive Clock Mode Register). Receive Clocks can be inverted independently by the CKI bits in SSC\_RCMR.

The receiver can also drive the RK I/O pad continuously or be limited to the actual data transfer. The clock output is configured by the SSC\_RCMR register. The Receive Clock Inversion (CKI) bits have no effect on the clock outputs. Programming the RCMR register to select RK pin (CKS field) and at the same time Continuous Receive Clock (CKO field) can lead to unpredictable results.

**Figure 35-7.** Receiver Clock Management



## 35.6.1.4 Serial Clock Ratio Considerations

The Transmitter and the Receiver can be programmed to operate with the clock signals provided on either the TK or RK pins. This allows the SSC to support many slave-mode data transfers. In this case, the maximum clock speed allowed on the RK pin is:

- Master Clock divided by 2 if Receiver Frame Synchro is input
- Master Clock divided by 3 if Receiver Frame Synchro is output

In addition, the maximum clock speed allowed on the TK pin is:

- Master Clock divided by 6 if Transmit Frame Synchro is input
- Master Clock divided by 2 if Transmit Frame Synchro is output

## 35.6.2 Transmitter Operations

A transmitted frame is triggered by a start event and can be followed by synchronization data before data transmission.

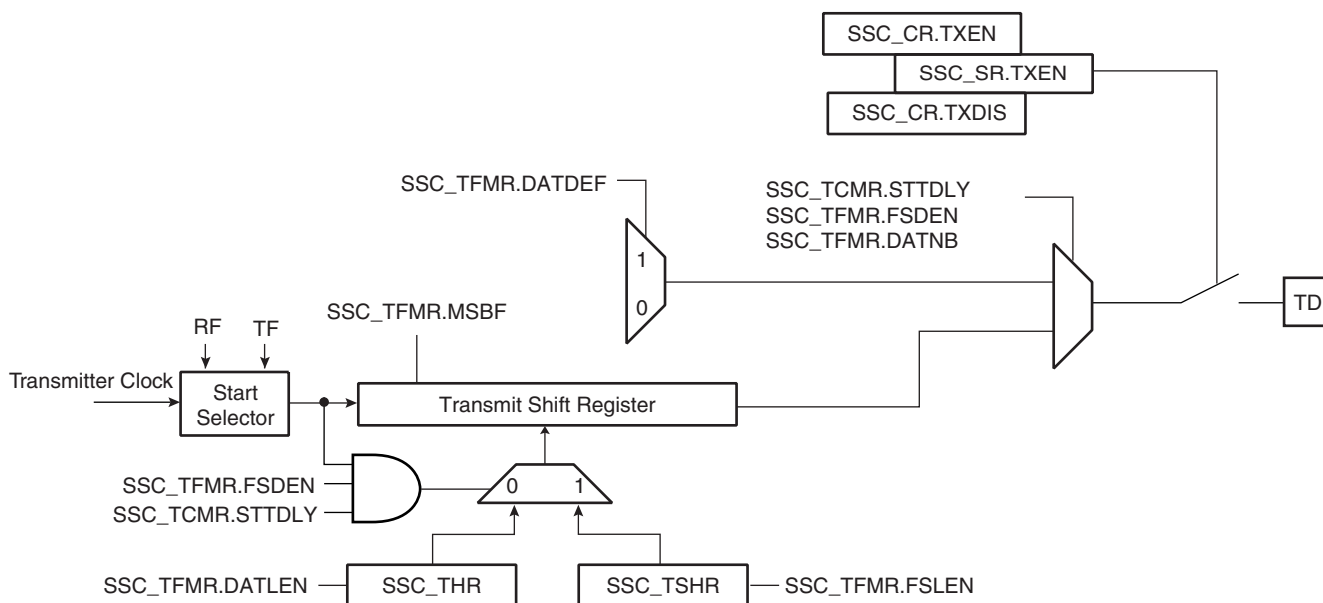
The start event is configured by setting the Transmit Clock Mode Register (SSC\_TCMR). See “Start” on page 510.

The frame synchronization is configured setting the Transmit Frame Mode Register (SSC\_TFMR). See “Frame Sync” on page 512.

To transmit data, the transmitter uses a shift register clocked by the transmitter clock signal and the start mode selected in the SSC\_TCMR. Data is written by the application to the SSC\_THR register then transferred to the shift register according to the data format selected.

When both the SSC\_THR and the transmit shift register are empty, the status flag TXEMPTY is set in SSC\_SR. When the Transmit Holding register is transferred in the Transmit shift register, the status flag TXRDY is set in SSC\_SR and additional data can be loaded in the holding register.

**Figure 35-8.** Transmitter Block Diagram



### 35.6.3 Receiver Operations

A received frame is triggered by a start event and can be followed by synchronization data before data transmission.

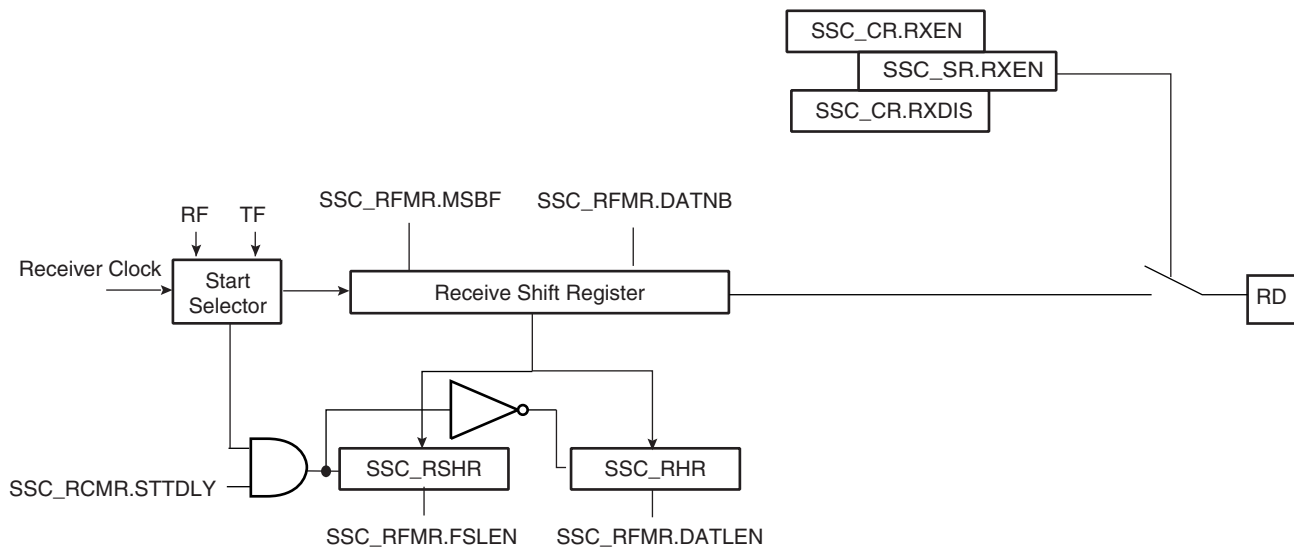
The start event is configured setting the Receive Clock Mode Register (SSC\_RCMR). See “Start” on page 510.

The frame synchronization is configured setting the Receive Frame Mode Register (SSC\_RFMR). See “Frame Sync” on page 512.

The receiver uses a shift register clocked by the receiver clock signal and the start mode selected in the SSC\_RCMR. The data is transferred from the shift register depending on the data format selected.

When the receiver shift register is full, the SSC transfers this data in the holding register, the status flag RXRDY is set in SSC\_SR and the data can be read in the receiver holding register. If another transfer occurs before read of the RHR register, the status flag OVERUN is set in SSC\_SR and the receiver shift register is transferred in the RHR register.

**Figure 35-9.** Receiver Block Diagram



### 35.6.4 Start

The transmitter and receiver can both be programmed to start their operations when an event occurs, respectively in the Transmit Start Selection (START) field of SSC\_TCMR and in the Receive Start Selection (START) field of SSC\_RCMR.

Under the following conditions the start event is independently programmable:

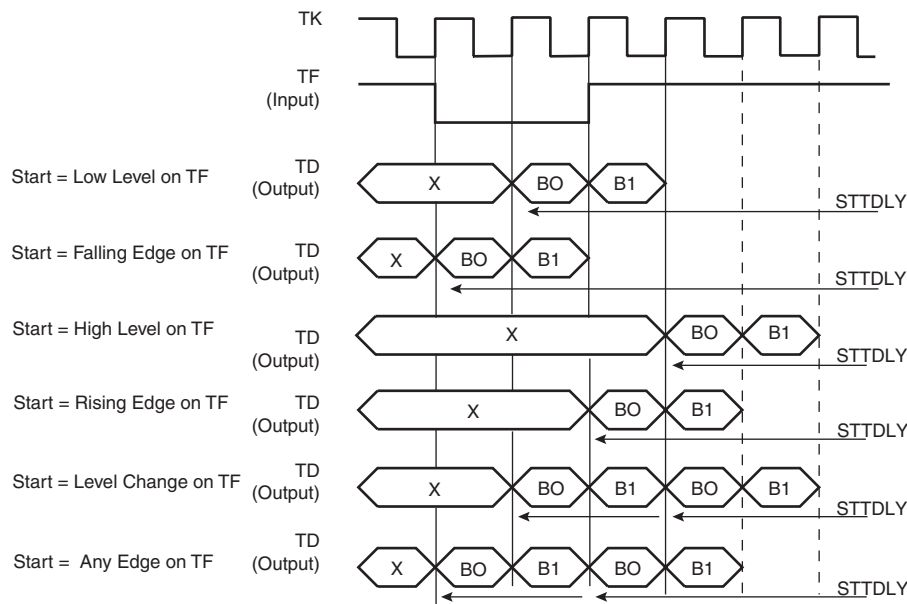
- Continuous. In this case, the transmission starts as soon as a word is written in SSC\_THR and the reception starts as soon as the Receiver is enabled.
- Synchronously with the transmitter/receiver
- On detection of a falling/rising edge on TF/RF
- On detection of a low level/high level on TF/RF
- On detection of a level change or an edge on TF/RF

A start can be programmed in the same manner on either side of the Transmit/Receive Clock Register (RCMR/TCMR). Thus, the start could be on TF (Transmit) or RF (Receive).

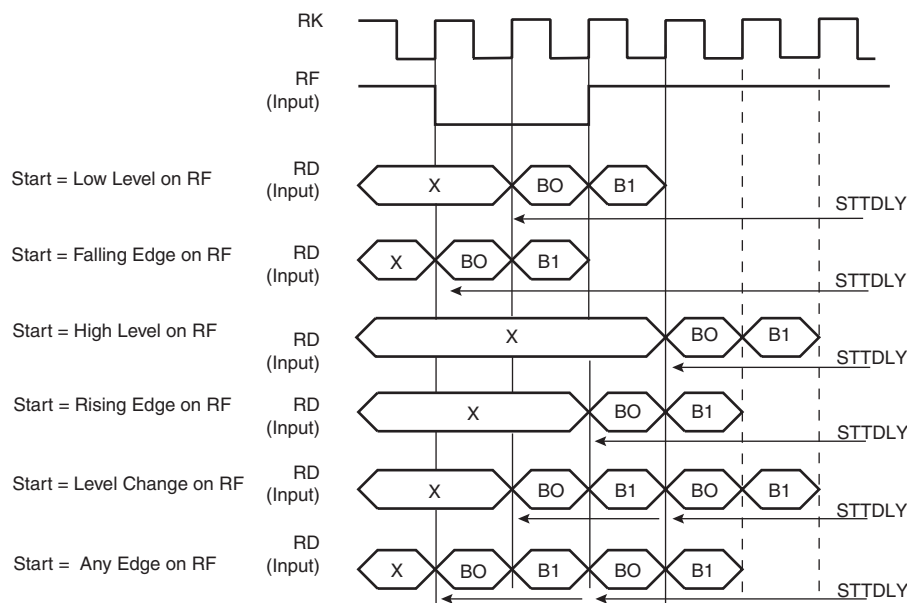
Moreover, the Receiver can start when data is detected in the bit stream with the Compare Functions.

Detection on TF/RF input/output is done by the field FSOS of the Transmit/Receive Frame Mode Register (TFMR/RFMR).

**Figure 35-10. Transmit Start Mode**



**Figure 35-11. Receive Pulse/Edge Start Modes**



### 35.6.5 Frame Sync

The Transmitter and Receiver Frame Sync pins, TF and RF, can be programmed to generate different kinds of frame synchronization signals. The Frame Sync Output Selection (FSOS) field in the Receive Frame Mode Register (SSC\_RFMR) and in the Transmit Frame Mode Register (SSC\_TFMR) are used to select the required waveform.

- Programmable low or high levels during data transfer are supported.
- Programmable high levels before the start of data transfers or toggling are also supported.

If a pulse waveform is selected, the Frame Sync Length (FSLEN) field in SSC\_RFMR and SSC\_TFMR programs the length of the pulse, from 1 bit time up to 256 bit time.

The periodicity of the Receive and Transmit Frame Sync pulse output can be programmed through the Period Divider Selection (PERIOD) field in SSC\_RCMR and SSC\_TCMR.

#### 35.6.5.1 Frame Sync Data

Frame Sync Data transmits or receives a specific tag during the Frame Sync signal.

During the Frame Sync signal, the Receiver can sample the RD line and store the data in the Receive Sync Holding Register and the transmitter can transfer Transmit Sync Holding Register in the Shifter Register. The data length to be sampled/shifted out during the Frame Sync signal is programmed by the FSLEN field in SSC\_RFMR/SSC\_TFMR and has a maximum value of 16.

Concerning the Receive Frame Sync Data operation, if the Frame Sync Length is equal to or lower than the delay between the start event and the actual data reception, the data sampling operation is performed in the Receive Sync Holding Register through the Receive Shift Register.

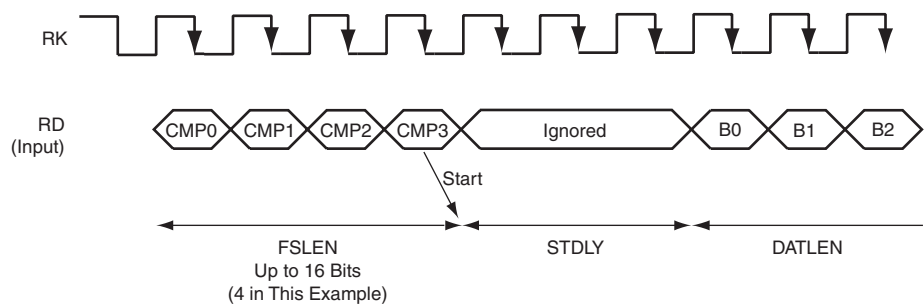
The Transmit Frame Sync Operation is performed by the transmitter only if the bit Frame Sync Data Enable (FSDEN) in SSC\_TFMR is set. If the Frame Sync length is equal to or lower than the delay between the start event and the actual data transmission, the normal transmission has priority and the data contained in the Transmit Sync Holding Register is transferred in the Transmit Register, then shifted out.

#### 35.6.5.2 Frame Sync Edge Detection

The Frame Sync Edge detection is programmed by the FSEDGE field in SSC\_RFMR/SSC\_TFMR. This sets the corresponding flags RXSYN/TXSYN in the SSC Status Register (SSC\_SR) on frame synchro edge detection (signals RF/TF).

### 35.6.6 Receive Compare Modes

**Figure 35-12.** Receive Compare Modes





## 35.6.6.1 Compare Functions

Length of the comparison patterns (Compare 0, Compare 1) and thus the number of bits they are compared to is defined by FSLEN, but with a maximum value of 16 bits. Comparison is always done by comparing the last bits received with the comparison pattern. Compare 0 can be one start event of the Receiver. In this case, the receiver compares at each new sample the last bits received at the Compare 0 pattern contained in the Compare 0 Register (SSC\_RC0R). When this start event is selected, the user can program the Receiver to start a new data transfer either by writing a new Compare 0, or by receiving continuously until Compare 1 occurs. This selection is done with the bit (STOP) in SSC\_RCMR.

## 35.6.7 Data Format

The data framing format of both the transmitter and the receiver are programmable through the Transmitter Frame Mode Register (SSC\_TFMR) and the Receiver Frame Mode Register (SSC\_RFMR). In either case, the user can independently select:

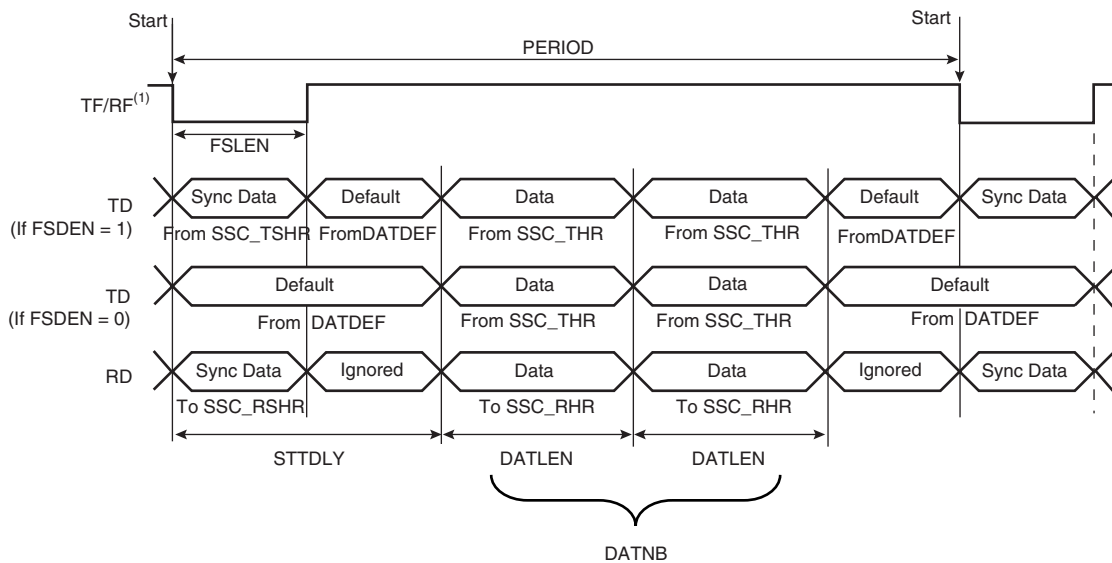
- the event that starts the data transfer (START)
- the delay in number of bit periods between the start event and the first data bit (STTDLY)
- the length of the data (DATLEN)
- the number of data to be transferred for each start event (DATNB).
- the length of synchronization transferred for each start event (FSLEN)
- the bit sense: most or lowest significant bit first (MSBF)

Additionally, the transmitter can be used to transfer synchronization and select the level driven on the TD pin while not in data transfer operation. This is done respectively by the Frame Sync Data Enable (FSDEN) and by the Data Default Value (DATDEF) bits in SSC\_TFMR.

**Table 35-3.** Data Frame Registers

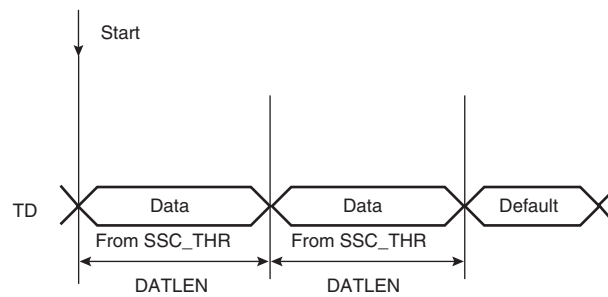
Transmitter	Receiver	Field	Length	Comment
SSC_TFMR	SSC_RFMR	DATLEN	Up to 32	Size of word
SSC_TFMR	SSC_RFMR	DATNB	Up to 16	Number of words transmitted in frame
SSC_TFMR	SSC_RFMR	MSBF		Most significant bit first
SSC_TFMR	SSC_RFMR	FSLEN	Up to 16	Size of Synchro data register
SSC_TFMR		DATDEF	0 or 1	Data default value ended
SSC_TFMR		FSDEN		Enable send SSC_TSHR
SSC_TCMR	SSC_RCMR	PERIOD	Up to 512	Frame size
SSC_TCMR	SSC_RCMR	STTDLY	Up to 255	Size of transmit start delay

**Figure 35-13.** Transmit and Receive Frame Format in Edge/Pulse Start Modes



Note: 1. Example of input on falling edge of TF/RF.

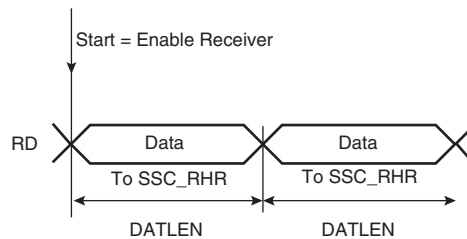
**Figure 35-14. Transmit Frame Format in Continuous Mode**



Start: 1. TXEMPTY set to 1  
2. Write into the SSC\_THR

Note: 1. STTDLY is set to 0. In this example, SSC\_THR is loaded twice. FSDEN value has no effect on the transmission. SyncData cannot be output in continuous mode.

**Figure 35-15. Receive Frame Format in Continuous Mode**



Note: 1. STTDLY is set to 0.

### 35.6.8 Loop Mode

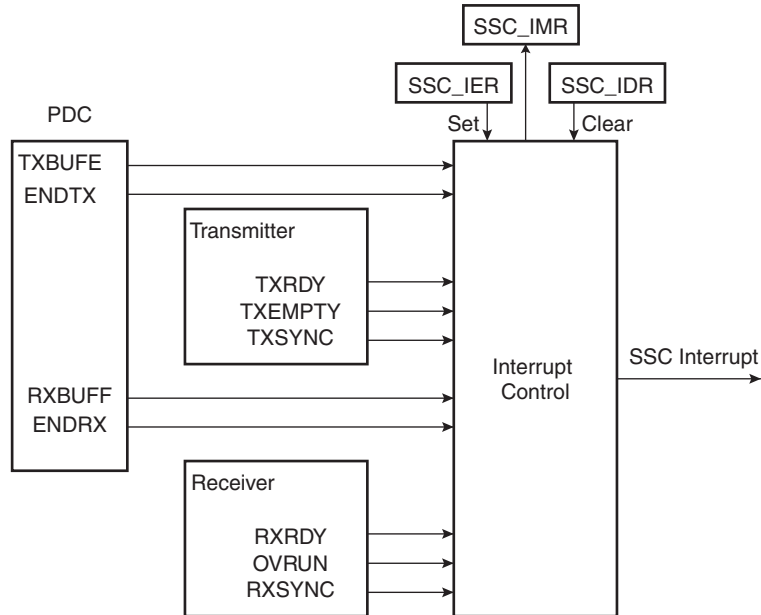
The receiver can be programmed to receive transmissions from the transmitter. This is done by setting the Loop Mode (LOOP) bit in SSC\_RFMR. In this case, RD is connected to TD, RF is connected to TF and RK is connected to TK.

### 35.6.9 Interrupt

Most bits in SSC\_SR have a corresponding bit in interrupt management registers.

The SSC can be programmed to generate an interrupt when it detects an event. The interrupt is controlled by writing SSC\_IER (Interrupt Enable Register) and SSC\_IDR (Interrupt Disable Register) These registers enable and disable, respectively, the corresponding interrupt by setting and clearing the corresponding bit in SSC\_IMR (Interrupt Mask Register), which controls the generation of interrupts by asserting the SSC interrupt line connected to the AIC.

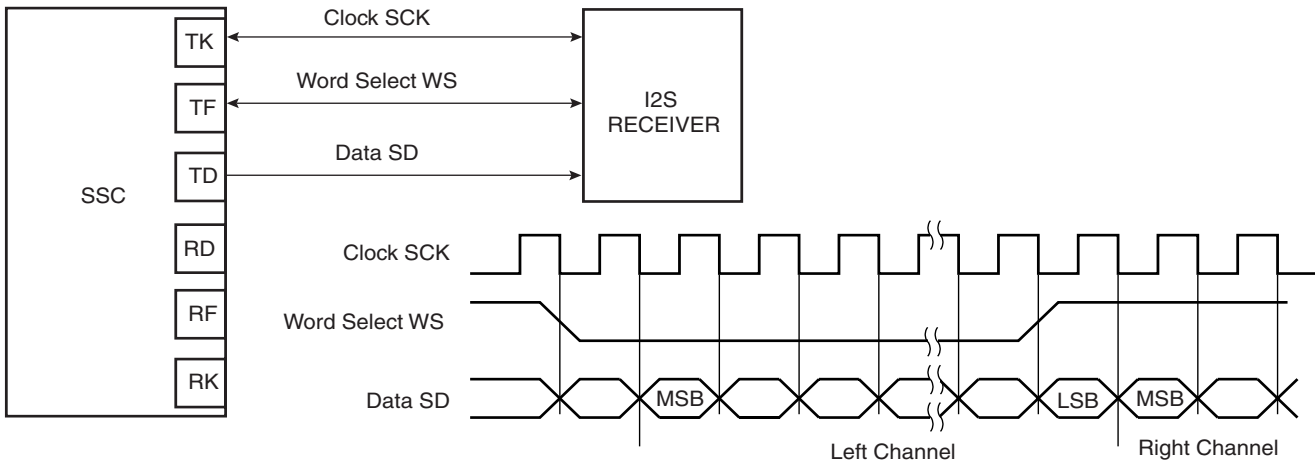
**Figure 35-16. Interrupt Block Diagram**



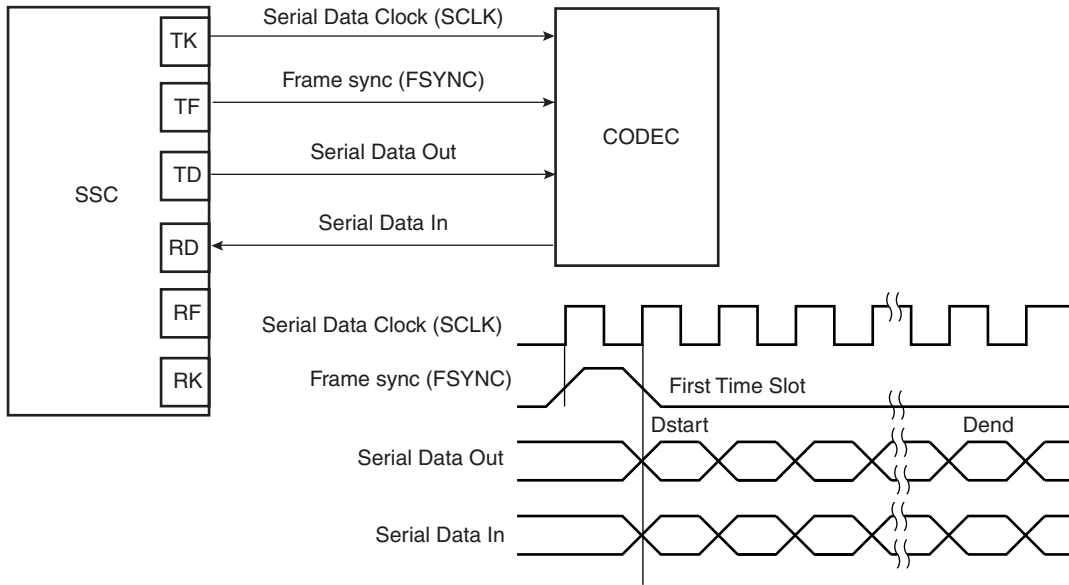
### 35.7 SSC Application Examples

The SSC can support several serial communication modes used in audio or high speed serial links. Some standard applications are shown in the following figures. All serial link applications supported by the SSC are not listed here.

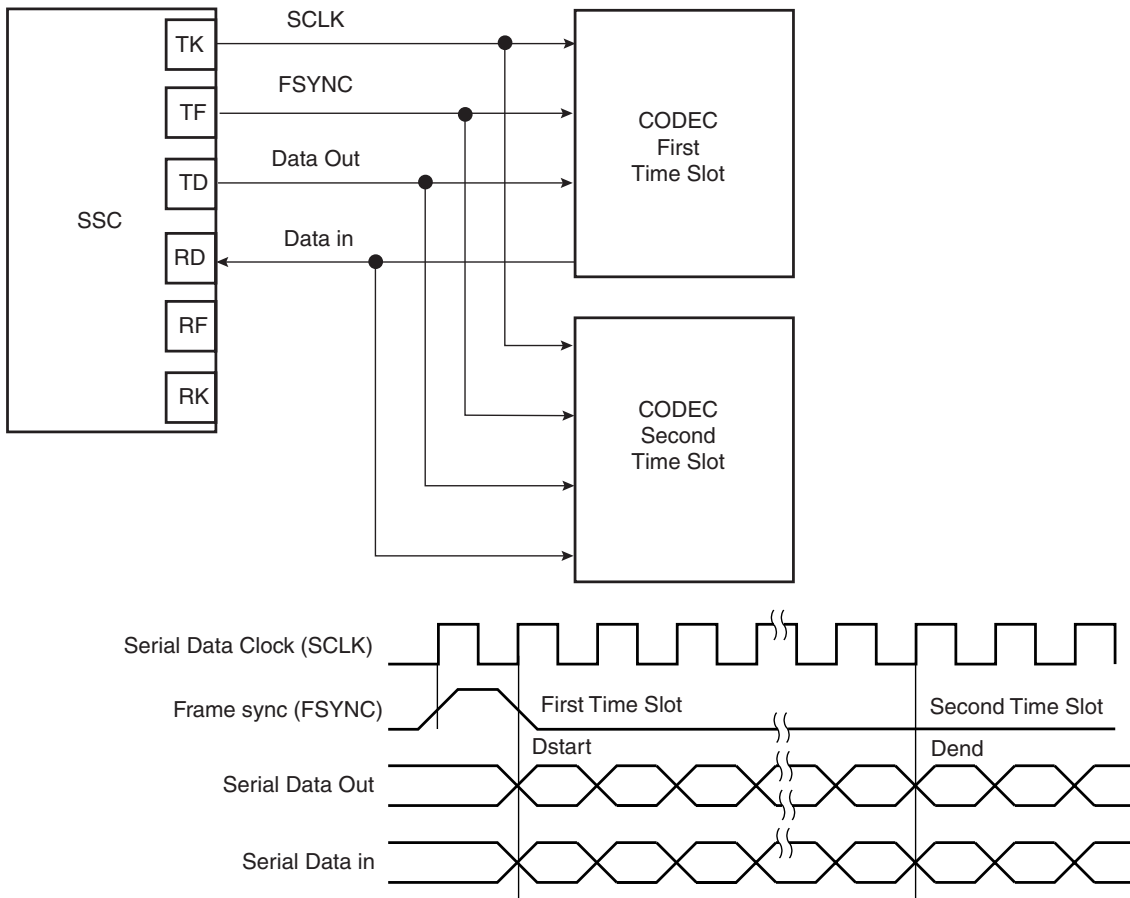
**Figure 35-17. Audio Application Block Diagram**



**Figure 35-18. Codec Application Block Diagram**



**Figure 35-19. Time Slot Application Block Diagram**



## 35.8 Synchronous Serial Controller (SSC) User Interface

**Table 35-4.** Register Mapping

Offset	Register	Register Name	Access	Reset
0x0	Control Register	SSC_CR	Write	–
0x4	Clock Mode Register	SSC_CMR	Read/Write	0x0
0x8	Reserved	–	–	–
0xC	Reserved	–	–	–
0x10	Receive Clock Mode Register	SSC_RCMR	Read/Write	0x0
0x14	Receive Frame Mode Register	SSC_RFMR	Read/Write	0x0
0x18	Transmit Clock Mode Register	SSC_TCMR	Read/Write	0x0
0x1C	Transmit Frame Mode Register	SSC_TFMR	Read/Write	0x0
0x20	Receive Holding Register	SSC_RHR	Read	0x0
0x24	Transmit Holding Register	SSC_THR	Write	–
0x28	Reserved	–	–	–
0x2C	Reserved	–	–	–
0x30	Receive Sync. Holding Register	SSC_RSHR	Read	0x0
0x34	Transmit Sync. Holding Register	SSC_TSHR	Read/Write	0x0
0x38	Receive Compare 0 Register	SSC_RC0R	Read/Write	0x0
0x3C	Receive Compare 1 Register	SSC_RC1R	Read/Write	0x0
0x40	Status Register	SSC_SR	Read	0x000000CC
0x44	Interrupt Enable Register	SSC_IER	Write	–
0x48	Interrupt Disable Register	SSC_IDR	Write	–
0x4C	Interrupt Mask Register	SSC_IMR	Read	0x0
0x50-0xFC	Reserved	–	–	–
0x100- 0x124	Reserved for Peripheral Data Controller (PDC)	–	–	–

## 35.8.1 SSC Control Register

**Name:** SSC\_CR

**Access Type:** Write-only

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

- **RXEN: Receive Enable**

0: No effect.

1: Enables Receive if RXDIS is not set.

- **RXDIS: Receive Disable**

0: No effect.

1: Disables Receive. If a character is currently being received, disables at end of current character reception.

- **TXEN: Transmit Enable**

0: No effect.

1: Enables Transmit if TXDIS is not set.

- **TXDIS: Transmit Disable**

0: No effect.

1: Disables Transmit. If a character is currently being transmitted, disables at end of current character transmission.

- **SWRST: Software Reset**

0: No effect.

1: Performs a software reset. Has priority on any other bit in SSC\_CR.



### 35.8.2 SSC Clock Mode Register

Name: SSC\_CMCR

Access Type: Read/Write

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

- **DIV: Clock Divider**

0: The Clock Divider is not active.

Any Other Value: The Divided Clock equals the Master Clock divided by 2 times DIV. The maximum bit rate is  $MCK/2$ . The minimum bit rate is  $MCK/2 \times 4095 = MCK/8190$ .



## 35.8.3 SSC Receive Clock Mode Register

Name: SSC\_RCMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
PERIOD							
23	22	21	20	19	18	17	16
STDDLY							
15	14	13	12	11	10	9	8
-	-	-	STOP	START			
7	6	5	4	3	2	1	0
CKG		CKI	CKO			CKS	

### • CKS: Receive Clock Selection

CKS	Selected Receive Clock
0x0	Divided Clock
0x1	TK Clock signal
0x2	RK pin
0x3	Reserved

### • CKO: Receive Clock Output Mode Selection

CKO	Receive Clock Output Mode	RK Pin
0x0	None	Input-only
0x1	Continuous Receive Clock	Output
0x2	Receive Clock only during data transfers	Output
0x3-0x7	Reserved	

### • CKI: Receive Clock Inversion

0: The data inputs (Data and Frame Sync signals) are sampled on Receive Clock falling edge. The Frame Sync signal output is shifted out on Receive Clock rising edge.

1: The data inputs (Data and Frame Sync signals) are sampled on Receive Clock rising edge. The Frame Sync signal output is shifted out on Receive Clock falling edge.

CKI affects only the Receive Clock and not the output clock signal.

• **CKG: Receive Clock Gating Selection**

CKG	Receive Clock Gating
0x0	None, continuous clock
0x1	Receive Clock enabled only if RF Low
0x2	Receive Clock enabled only if RF High
0x3	Reserved

• **START: Receive Start Selection**

START	Receive Start
0x0	Continuous, as soon as the receiver is enabled, and immediately after the end of transfer of the previous data.
0x1	Transmit start
0x2	Detection of a low level on RF signal
0x3	Detection of a high level on RF signal
0x4	Detection of a falling edge on RF signal
0x5	Detection of a rising edge on RF signal
0x6	Detection of any level change on RF signal
0x7	Detection of any edge on RF signal
0x8	Compare 0
0x9-0xF	Reserved

• **STOP: Receive Stop Selection**

0: After completion of a data transfer when starting with a Compare 0, the receiver stops the data transfer and waits for a new compare 0.

1: After starting a receive with a Compare 0, the receiver operates in a continuous mode until a Compare 1 is detected.

• **STTDLY: Receive Start Delay**

If STTDLY is not 0, a delay of STTDLY clock cycles is inserted between the start event and the actual start of reception. When the Receiver is programmed to start synchronously with the Transmitter, the delay is also applied.

Note: It is very important that STTDLY be set carefully. If STTDLY must be set, it should be done in relation to TAG (Receive Sync Data) reception.

• **PERIOD: Receive Period Divider Selection**

This field selects the divider to apply to the selected Receive Clock in order to generate a new Frame Sync Signal. If 0, no PERIOD signal is generated. If not 0, a PERIOD signal is generated each 2 x (PERIOD+1) Receive Clock.

## 35.8.4 SSC Receive Frame Mode Register

**Name:** SSC\_RFMR

**Access Type:** Read/Write

31	30	29	28	27	26	25	24	
FSLEN_EXT	FSLEN_EXT	FSLEN_EXT	FSLEN_EXT	–	–	–	FSEDGE	
23	22	21	20	19	18	17	16	
–	FSOS			FSLEN				
15	14	13	12	11	10	9	8	
–	–	–	–	DATNB				
7	6	5	4	3	2	1	0	
MSBF	–	LOOP	DATLEN					

- **DATLEN: Data Length**

0: Forbidden value (1-bit data length not supported).

Any other value: The bit stream contains DATLEN + 1 data bits. Moreover, it defines the transfer size performed by the PDC2 assigned to the Receiver. If DATLEN is lower or equal to 7, data transfers are in bytes. If DATLEN is between 8 and 15 (included), half-words are transferred, and for any other value, 32-bit words are transferred.

- **LOOP: Loop Mode**

0: Normal operating mode.

1: RD is driven by TD, RF is driven by TF and TK drives RK.

- **MSBF: Most Significant Bit First**

0: The lowest significant bit of the data register is sampled first in the bit stream.

1: The most significant bit of the data register is sampled first in the bit stream.

- **DATNB: Data Number per Frame**

This field defines the number of data words to be received after each transfer start, which is equal to (DATNB + 1).

- **FSLEN: Receive Frame Sync Length**

This field defines the number of bits sampled and stored in the Receive Sync Data Register. When this mode is selected by the START field in the Receive Clock Mode Register, it also determines the length of the sampled data to be compared to the Compare 0 or Compare 1 register.

This field is used with FSLEN\_EXT to determine the pulse length of the Receive Frame Sync signal.

Pulse length is equal to FSLEN + (FSLEN\_EXT \* 16) + 1 Receive Clock periods.

- **FSOS: Receive Frame Sync Output Selection**

FSOS	Selected Receive Frame Sync Signal	RF Pin
0x0	None	Input-only
0x1	Negative Pulse	Output
0x2	Positive Pulse	Output
0x3	Driven Low during data transfer	Output
0x4	Driven High during data transfer	Output
0x5	Toggling at each start of data transfer	Output
0x6-0x7	Reserved	Undefined

- **FSEDGE: Frame Sync Edge Detection**

Determines which edge on Frame Sync will generate the interrupt RXSYN in the SSC Status Register.

FSEDGE	Frame Sync Edge Detection
0x0	Positive Edge Detection
0x1	Negative Edge Detection

- **FSLEN\_EXT: FSLEN Field Extension**

Extends FSLEN field. For details, refer to FSLEN bit description on [page 523](#).

## 35.8.5 SSC Transmit Clock Mode Register

Name: SSC\_TCMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
PERIOD							
23	22	21	20	19	18	17	16
STTDLY							
15	14	13	12	11	10	9	8
-	-	-	-	START			
7	6	5	4	3	2	1	0
CKG		CKI	CKO			CKS	

### • CKS: Transmit Clock Selection

CKS	Selected Transmit Clock
0x0	Divided Clock
0x1	RK Clock signal
0x2	TK Pin
0x3	Reserved

### • CKO: Transmit Clock Output Mode Selection

CKO	Transmit Clock Output Mode	TK pin
0x0	None	Input-only
0x1	Continuous Transmit Clock	Output
0x2	Transmit Clock only during data transfers	Output
0x3-0x7	Reserved	

### • CKI: Transmit Clock Inversion

0: The data outputs (Data and Frame Sync signals) are shifted out on Transmit Clock falling edge. The Frame sync signal input is sampled on Transmit clock rising edge.

1: The data outputs (Data and Frame Sync signals) are shifted out on Transmit Clock rising edge. The Frame sync signal input is sampled on Transmit clock falling edge.

CKI affects only the Transmit Clock and not the output clock signal.

- **CKG: Transmit Clock Gating Selection**

CKG	Transmit Clock Gating
0x0	None, continuous clock
0x1	Transmit Clock enabled only if TF Low
0x2	Transmit Clock enabled only if TF High
0x3	Reserved

- **START: Transmit Start Selection**

START	Transmit Start
0x0	Continuous, as soon as a word is written in the SSC_THR Register (if Transmit is enabled), and immediately after the end of transfer of the previous data.
0x1	Receive start
0x2	Detection of a low level on TF signal
0x3	Detection of a high level on TF signal
0x4	Detection of a falling edge on TF signal
0x5	Detection of a rising edge on TF signal
0x6	Detection of any level change on TF signal
0x7	Detection of any edge on TF signal
0x8 - 0xF	Reserved

- **STTDLY: Transmit Start Delay**

If STTDLY is not 0, a delay of STTDLY clock cycles is inserted between the start event and the actual start of transmission of data. When the Transmitter is programmed to start synchronously with the Receiver, the delay is also applied.

Note: STTDLY must be set carefully. If STTDLY is too short in respect to TAG (Transmit Sync Data) emission, data is emitted instead of the end of TAG.

- **PERIOD: Transmit Period Divider Selection**

This field selects the divider to apply to the selected Transmit Clock to generate a new Frame Sync Signal. If 0, no period signal is generated. If not 0, a period signal is generated at each  $2 \times (\text{PERIOD} + 1)$  Transmit Clock.

## 35.8.6 SSC Transmit Frame Mode Register

**Name:** SSC\_TFMR

**Access Type:** Read/Write

31	30	29	28	27	26	25	24
FSLEN_EXT	FSLEN_EXT	FSLEN_EXT	FSLEN_EXT	–	–	–	FSEDGE
23	22	21	20	19	18	17	16
FSDEN	FSOS			FSLEN			
15	14	13	12	11	10	9	8
–	–	–	–	DATNB			
7	6	5	4	3	2	1	0
MSBF	–	DATDEF	DATLEN				

- **DATLEN: Data Length**

0: Forbidden value (1-bit data length not supported).

Any other value: The bit stream contains DATLEN + 1 data bits. Moreover, it defines the transfer size performed by the PDC2 assigned to the Transmit. If DATLEN is lower or equal to 7, data transfers are bytes, if DATLEN is between 8 and 15 (included), half-words are transferred, and for any other value, 32-bit words are transferred.

- **DATDEF: Data Default Value**

This bit defines the level driven on the TD pin while out of transmission. Note that if the pin is defined as multi-drive by the PIO Controller, the pin is enabled only if the SCC TD output is 1.

- **MSBF: Most Significant Bit First**

0: The lowest significant bit of the data register is shifted out first in the bit stream.

1: The most significant bit of the data register is shifted out first in the bit stream.

- **DATNB: Data Number per frame**

This field defines the number of data words to be transferred after each transfer start, which is equal to (DATNB + 1).

- **FSLEN: Transmit Frame Sync Length**

This field defines the length of the Transmit Frame Sync signal and the number of bits shifted out from the Transmit Sync Data Register if FSDEN is 1.

This field is used with FSLEN\_EXT to determine the pulse length of the Transmit Frame Sync signal.

Pulse length is equal to  $FSLEN + (FSLEN\_EXT * 16) + 1$  Transmit Clock periods.

- **FSOS: Transmit Frame Sync Output Selection**

FSOS	Selected Transmit Frame Sync Signal	TF Pin
0x0	None	Input-only
0x1	Negative Pulse	Output
0x2	Positive Pulse	Output
0x3	Driven Low during data transfer	Output
0x4	Driven High during data transfer	Output
0x5	Toggling at each start of data transfer	Output
0x6-0x7	Reserved	Undefined

- **FSDEN: Frame Sync Data Enable**

0: The TD line is driven with the default value during the Transmit Frame Sync signal.

1: SSC\_TSHR value is shifted out during the transmission of the Transmit Frame Sync signal.

- **FSEDGE: Frame Sync Edge Detection**

Determines which edge on frame sync will generate the interrupt TXSYN (Status Register).

FSEDGE	Frame Sync Edge Detection
0x0	Positive Edge Detection
0x1	Negative Edge Detection

- **FSLEN\_EXT: FSLEN Field Extension**

Extends FSLEN field. For details, refer to FSLEN bit description on [page 527](#).



## 35.8.7 SSC Receive Holding Register

Name: SSC\_RHR

Access Type: Read-only

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

- **RDAT: Receive Data**

Right aligned regardless of the number of data bits defined by DATLEN in SSC\_RFMR.

## 35.8.8 SSC Transmit Holding Register

Name: SSC\_THR

Access Type: Write-only

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

- **TDAT: Transmit Data**

Right aligned regardless of the number of data bits defined by DATLEN in SSC\_TFMR.



### 35.8.9 SSC Receive Synchronization Holding Register

Name: SSC\_RSHR

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
RSDAT							
7	6	5	4	3	2	1	0
RSDAT							

- RSDAT: Receive Synchronization Data

### 35.8.10 SSC Transmit Synchronization Holding Register

Name: SSC\_TSHR

Access Type: Read/Write

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

- TSDAT: Transmit Synchronization Data



## 35.8.11 SSC Receive Compare 0 Register

**Name:** SSC\_RC0R

**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
CP0							
7	6	5	4	3	2	1	0
CP0							

- CP0: Receive Compare Data 0



### 35.8.12 SSC Receive Compare 1 Register

Name: SSC\_RC1R

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
CP1							
7	6	5	4	3	2	1	0
CP1							

- CP1: Receive Compare Data 1



## 35.8.13 SSC Status Register

**Name:** SSC\_SR

**Access Type:** Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	RXEN	TXEN
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready**

0: Data has been loaded in SSC\_THR and is waiting to be loaded in the Transmit Shift Register (TSR).

1: SSC\_THR is empty.

- **TXEMPTY: Transmit Empty**

0: Data remains in SSC\_THR or is currently transmitted from TSR.

1: Last data written in SSC\_THR has been loaded in TSR and last data loaded in TSR has been transmitted.

- **ENDTX: End of Transmission**

0: The register SSC\_TCR has not reached 0 since the last write in SSC\_TCR or SSC\_TNCR.

1: The register SSC\_TCR has reached 0 since the last write in SSC\_TCR or SSC\_TNCR.

- **TXBUFE: Transmit Buffer Empty**

0: SSC\_TCR or SSC\_TNCR have a value other than 0.

1: Both SSC\_TCR and SSC\_TNCR have a value of 0.

- **RXRDY: Receive Ready**

0: SSC\_RHR is empty.

1: Data has been received and loaded in SSC\_RHR.

- **OVRUN: Receive Overrun**

0: No data has been loaded in SSC\_RHR while previous data has not been read since the last read of the Status Register.

1: Data has been loaded in SSC\_RHR while previous data has not yet been read since the last read of the Status Register.

- **ENDRX: End of Reception**

0: Data is written on the Receive Counter Register or Receive Next Counter Register.

1: End of PDC transfer when Receive Counter Register has arrived at zero.

- **RXBUFF: Receive Buffer Full**

0: SSC\_RCR or SSC\_RNCR have a value other than 0.

1: Both SSC\_RCR and SSC\_RNCR have a value of 0.

- **CP0: Compare 0**

0: A compare 0 has not occurred since the last read of the Status Register.

1: A compare 0 has occurred since the last read of the Status Register.

- **CP1: Compare 1**

0: A compare 1 has not occurred since the last read of the Status Register.

1: A compare 1 has occurred since the last read of the Status Register.

- **TXSYN: Transmit Sync**

0: A Tx Sync has not occurred since the last read of the Status Register.

1: A Tx Sync has occurred since the last read of the Status Register.

- **RXSYN: Receive Sync**

0: An Rx Sync has not occurred since the last read of the Status Register.

1: An Rx Sync has occurred since the last read of the Status Register.

- **TXEN: Transmit Enable**

0: Transmit is disabled.

1: Transmit is enabled.

- **RXEN: Receive Enable**

0: Receive is disabled.

1: Receive is enabled.

## 35.8.14 SSC Interrupt Enable Register

Name: SSC\_IER

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Enable**

0: No effect.

1: Enables the Transmit Ready Interrupt.

- **TXEMPTY: Transmit Empty Interrupt Enable**

0: No effect.

1: Enables the Transmit Empty Interrupt.

- **ENDTX: End of Transmission Interrupt Enable**

0: No effect.

1: Enables the End of Transmission Interrupt.

- **TXBUFE: Transmit Buffer Empty Interrupt Enable**

0: No effect.

1: Enables the Transmit Buffer Empty Interrupt

- **RXRDY: Receive Ready Interrupt Enable**

0: No effect.

1: Enables the Receive Ready Interrupt.

- **OVRUN: Receive Overrun Interrupt Enable**

0: No effect.

1: Enables the Receive Overrun Interrupt.

- **ENDRX: End of Reception Interrupt Enable**

0: No effect.

1: Enables the End of Reception Interrupt.

- **RXBUFF: Receive Buffer Full Interrupt Enable**

0: No effect.



1: Enables the Receive Buffer Full Interrupt.

- **CP0: Compare 0 Interrupt Enable**

0: No effect.

1: Enables the Compare 0 Interrupt.

- **CP1: Compare 1 Interrupt Enable**

0: No effect.

1: Enables the Compare 1 Interrupt.

- **TXSYN: Tx Sync Interrupt Enable**

0: No effect.

1: Enables the Tx Sync Interrupt.

- **RXSYN: Rx Sync Interrupt Enable**

0: No effect.

1: Enables the Rx Sync Interrupt.



## 35.8.15 SSC Interrupt Disable Register

Name: SSC\_IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Disable**

0: No effect.

1: Disables the Transmit Ready Interrupt.

- **TXEMPTY: Transmit Empty Interrupt Disable**

0: No effect.

1: Disables the Transmit Empty Interrupt.

- **ENDTX: End of Transmission Interrupt Disable**

0: No effect.

1: Disables the End of Transmission Interrupt.

- **TXBUFE: Transmit Buffer Empty Interrupt Disable**

0: No effect.

1: Disables the Transmit Buffer Empty Interrupt.

- **RXRDY: Receive Ready Interrupt Disable**

0: No effect.

1: Disables the Receive Ready Interrupt.

- **OVRUN: Receive Overrun Interrupt Disable**

0: No effect.

1: Disables the Receive Overrun Interrupt.

- **ENDRX: End of Reception Interrupt Disable**

0: No effect.

1: Disables the End of Reception Interrupt.

- **RXBUFF: Receive Buffer Full Interrupt Disable**

0: No effect.

1: Disables the Receive Buffer Full Interrupt.

- **CP0: Compare 0 Interrupt Disable**

0: No effect.

1: Disables the Compare 0 Interrupt.

- **CP1: Compare 1 Interrupt Disable**

0: No effect.

1: Disables the Compare 1 Interrupt.

- **TXSYN: Tx Sync Interrupt Enable**

0: No effect.

1: Disables the Tx Sync Interrupt.

- **RXSYN: Rx Sync Interrupt Enable**

0: No effect.

1: Disables the Rx Sync Interrupt.

## 35.8.16 SSC Interrupt Mask Register

Name: SSC\_IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
RXBUF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Mask**

0: The Transmit Ready Interrupt is disabled.

1: The Transmit Ready Interrupt is enabled.

- **TXEMPTY: Transmit Empty Interrupt Mask**

0: The Transmit Empty Interrupt is disabled.

1: The Transmit Empty Interrupt is enabled.

- **ENDTX: End of Transmission Interrupt Mask**

0: The End of Transmission Interrupt is disabled.

1: The End of Transmission Interrupt is enabled.

- **TXBUFE: Transmit Buffer Empty Interrupt Mask**

0: The Transmit Buffer Empty Interrupt is disabled.

1: The Transmit Buffer Empty Interrupt is enabled.

- **RXRDY: Receive Ready Interrupt Mask**

0: The Receive Ready Interrupt is disabled.

1: The Receive Ready Interrupt is enabled.

- **OVRUN: Receive Overrun Interrupt Mask**

0: The Receive Overrun Interrupt is disabled.

1: The Receive Overrun Interrupt is enabled.

- **ENDRX: End of Reception Interrupt Mask**

0: The End of Reception Interrupt is disabled.

1: The End of Reception Interrupt is enabled.

- **RXBUFF: Receive Buffer Full Interrupt Mask**

0: The Receive Buffer Full Interrupt is disabled.

1: The Receive Buffer Full Interrupt is enabled.

- **CP0: Compare 0 Interrupt Mask**

0: The Compare 0 Interrupt is disabled.

1: The Compare 0 Interrupt is enabled.

- **CP1: Compare 1 Interrupt Mask**

0: The Compare 1 Interrupt is disabled.

1: The Compare 1 Interrupt is enabled.

- **TXSYN: Tx Sync Interrupt Mask**

0: The Tx Sync Interrupt is disabled.

1: The Tx Sync Interrupt is enabled.

- **RXSYN: Rx Sync Interrupt Mask**

0: The Rx Sync Interrupt is disabled.

1: The Rx Sync Interrupt is enabled.

## 36. Timer Counter (TC)

### 36.1 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 36-1](#) gives the assignment of the device Timer Counter clock inputs common to Timer Counter 0 to 2

**Table 36-1.** Timer Counter Clock Assignment

Name	Definition
TIMER_CLOCK1	MCK/2
TIMER_CLOCK2	MCK/8
TIMER_CLOCK3	MCK/32
TIMER_CLOCK4	MCK/128
TIMER_CLOCK5	SLCK

## 36.2 Block Diagram

Figure 36-1. Timer Counter Block Diagram

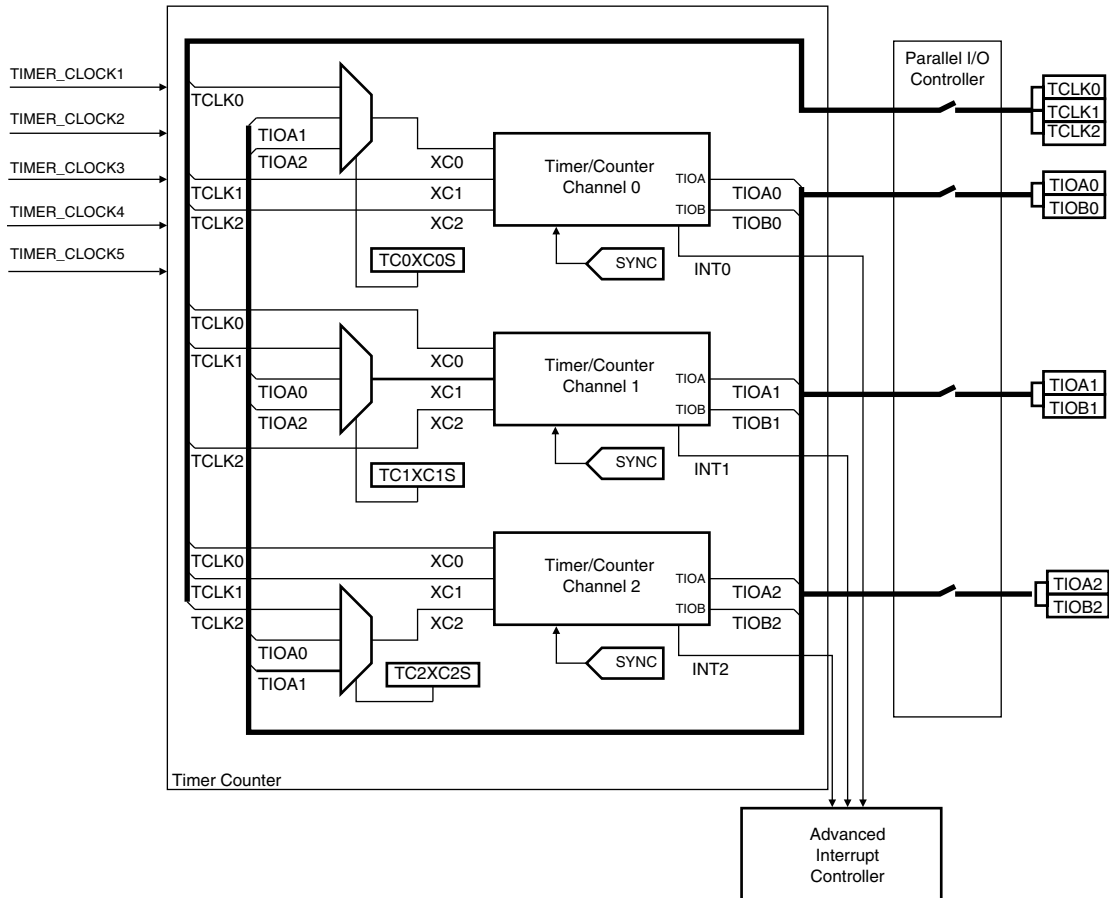


Table 36-2. Signal Name Description

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

## 36.3 Pin Name List

**Table 36-3.** 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

## 36.4 Product Dependencies

### 36.4.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the TC pins to their peripheral functions.

### 36.4.2 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.

### 36.4.3 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.

## 36.5 Functional Description

### 36.5.1 TC Description

The three channels of the Timer Counter are independent and identical in operation. The registers for channel programming are listed in [Table 36-5 on page 557](#).

### 36.5.2 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.

### 36.5.3 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 36-2 on page 545](#).

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.

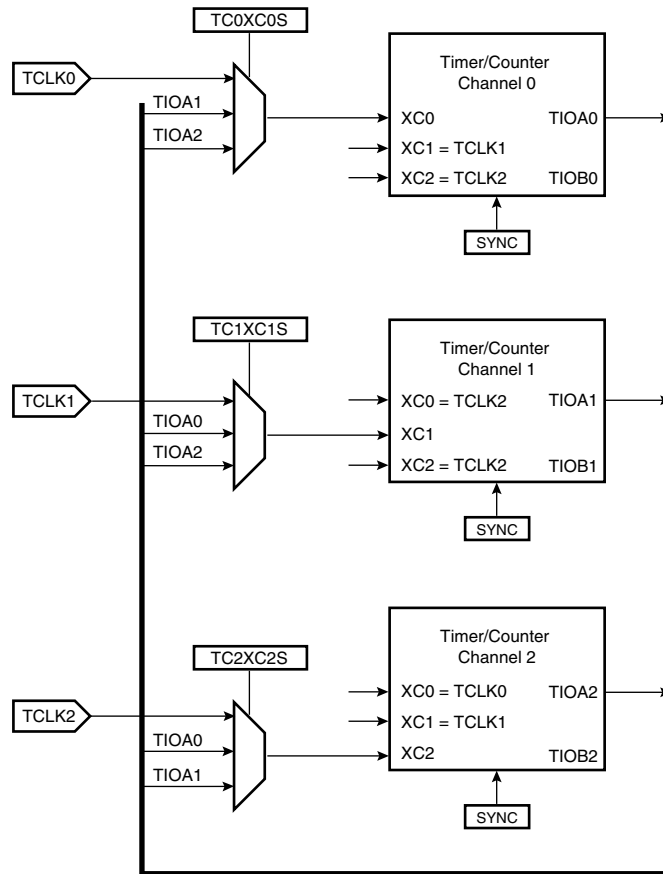
The selected clock can be inverted with the CLKI bit in TC\_CMR. This allows counting on the opposite edges of the clock.

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 36-3 on page 545](#)

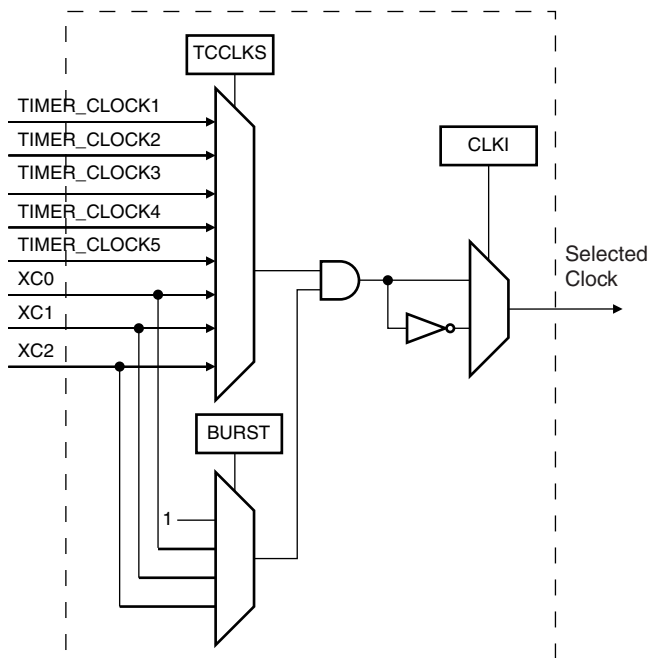
**Note:** In all cases, if an external clock is used, the duration of each of its levels must be longer than the master clock period. The external clock frequency must be at least 2.5 times lower than the master clock



**Figure 36-2.** Clock Chaining Selection



**Figure 36-3.** Clock Selection

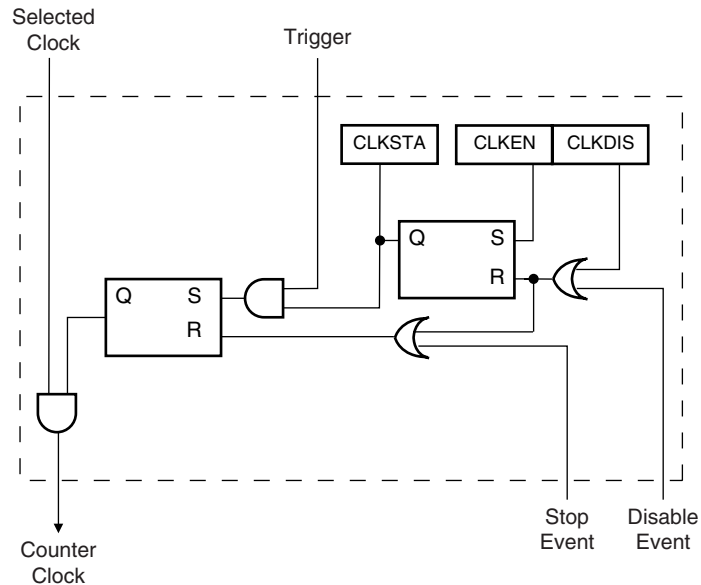


### 36.5.4 Clock Control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See [Figure 36-4](#).

- 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 36-4.** Clock Control



### 36.5.5 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.

### 36.5.6 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 ENETRIG in TC\_CMR.

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

Regardless of the trigger used, it will be taken into account at the following active edge of the selected clock. This means that the counter value can be read differently from zero just after a trigger, especially when a low frequency signal is selected as the clock.

### 36.5.7 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 36-5 shows the configuration of the TC channel when programmed in Capture Mode.

### 36.5.8 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.

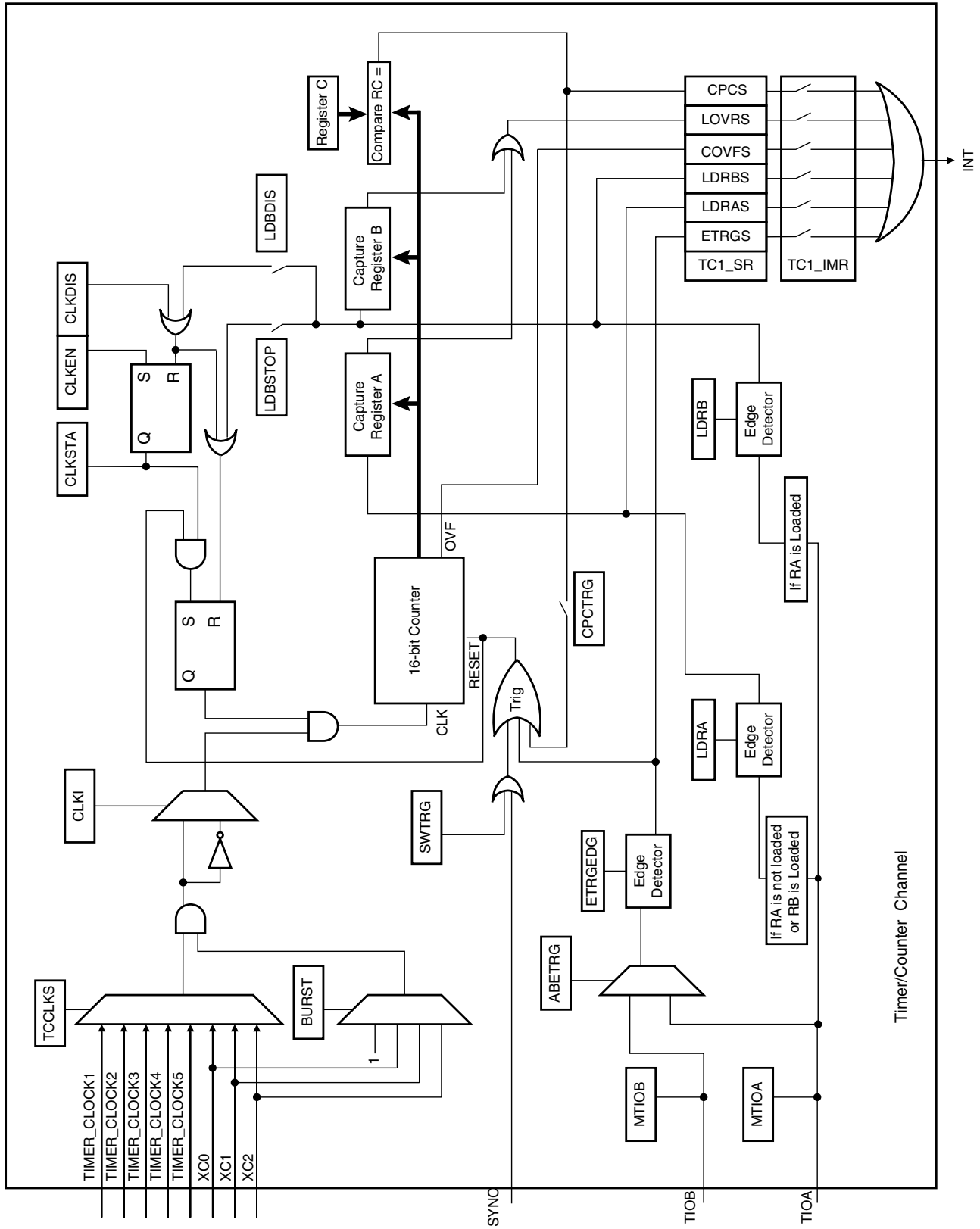
Loading RA or RB before the read of the last value loaded sets the Overrun Error Flag (LOVRS) in TC\_SR (Status Register). In this case, the old value is overwritten.

### 36.5.9 Trigger Conditions

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

The ABETRIG 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 36-5. Capture Mode



## 36.5.10 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 36-6 shows the configuration of the TC channel when programmed in Waveform Operating Mode.

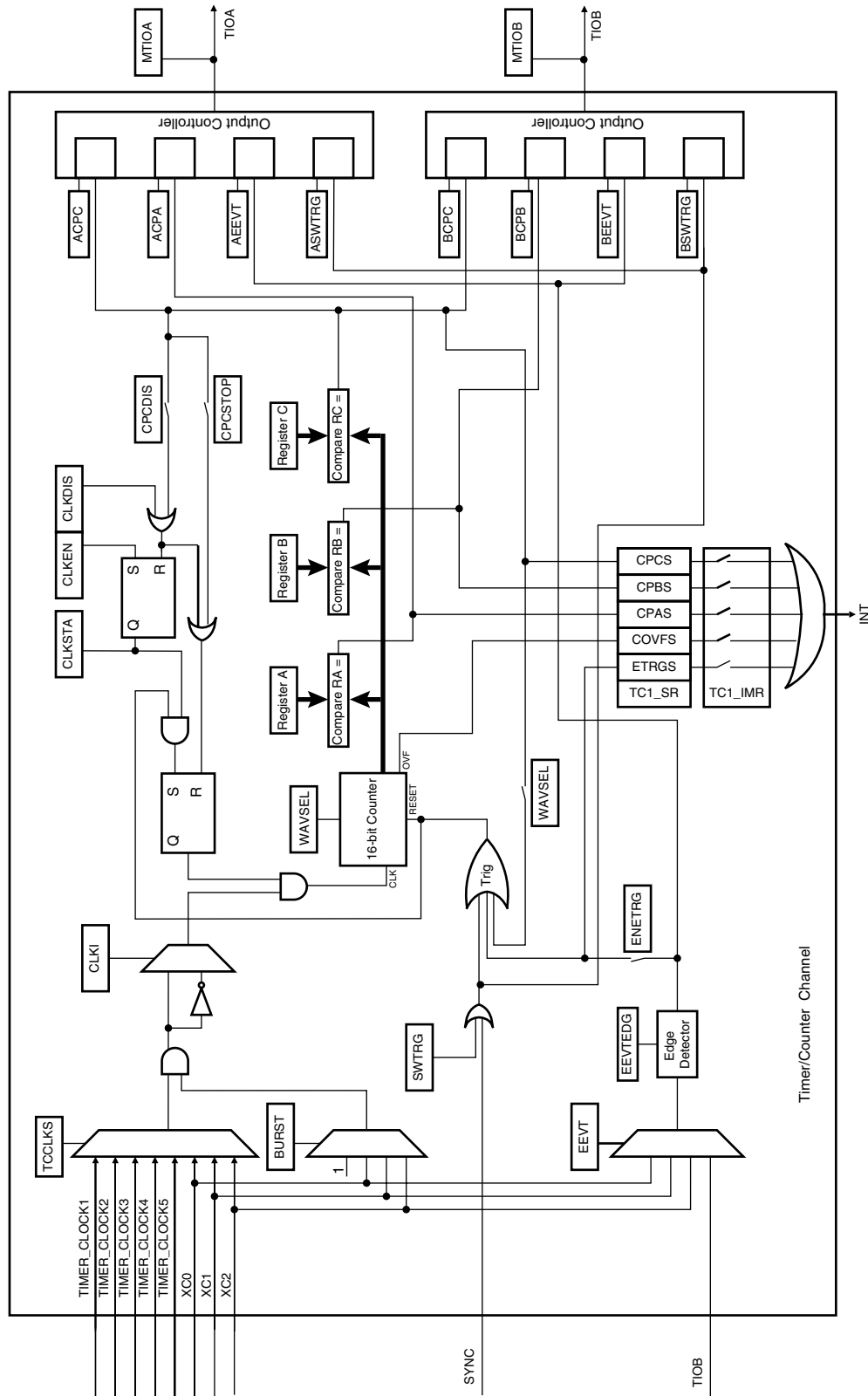
## 36.5.11 Waveform Selection

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 36-6. Waveform Mode



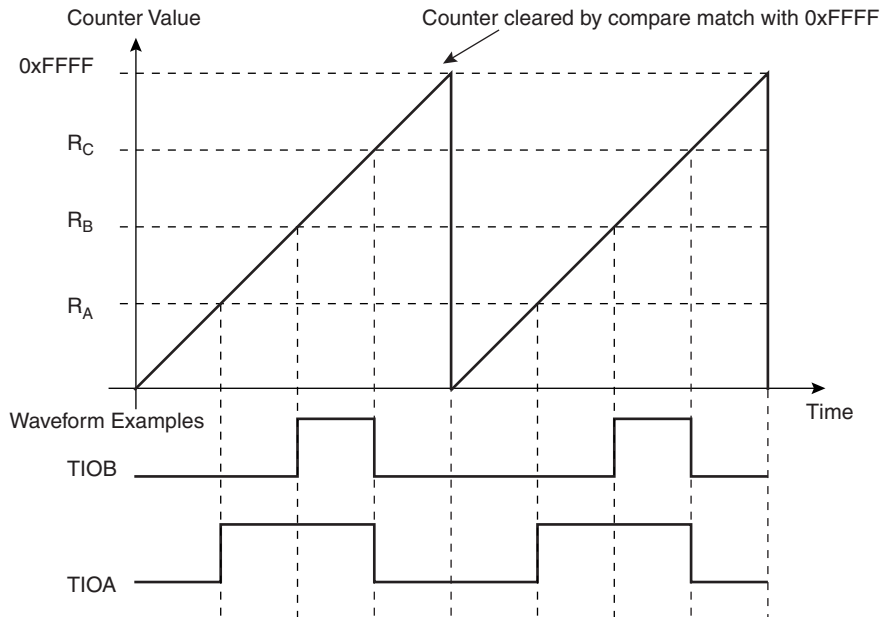
## 36.5.11.1 WAVSEL = 00

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 36-7](#).

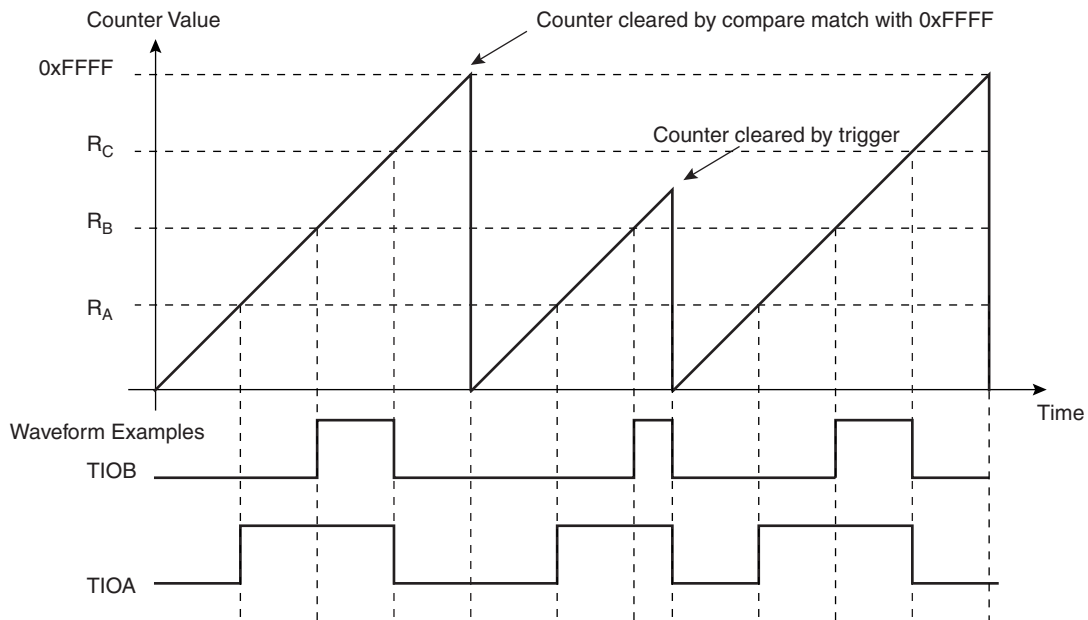
An external event trigger or a software trigger can reset the value of TC\_CV. It is important to note that the trigger may occur at any time. See [Figure 36-8](#).

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 36-7.** WAVSEL= 00 without trigger



**Figure 36-8.** WAVSEL= 00 with trigger



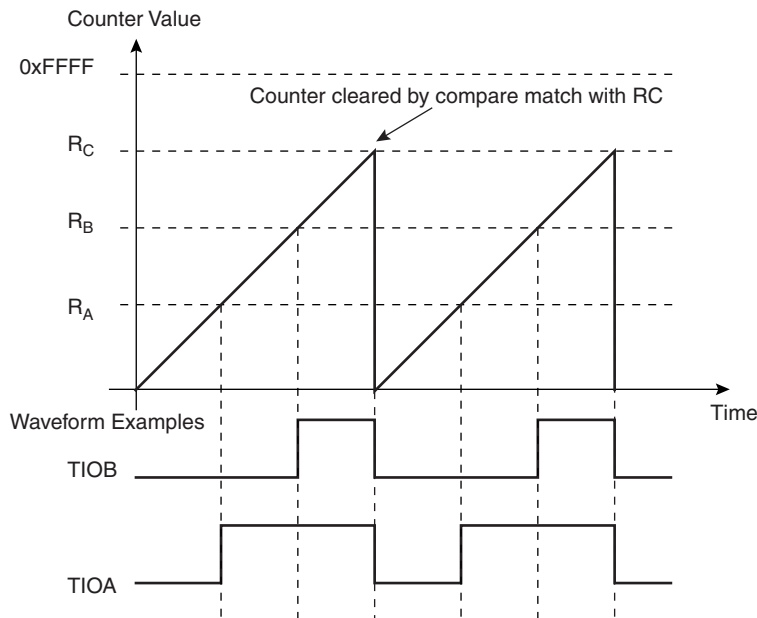
**36.5.11.2 WAVSEL = 10**

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 36-9](#).

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 36-10](#).

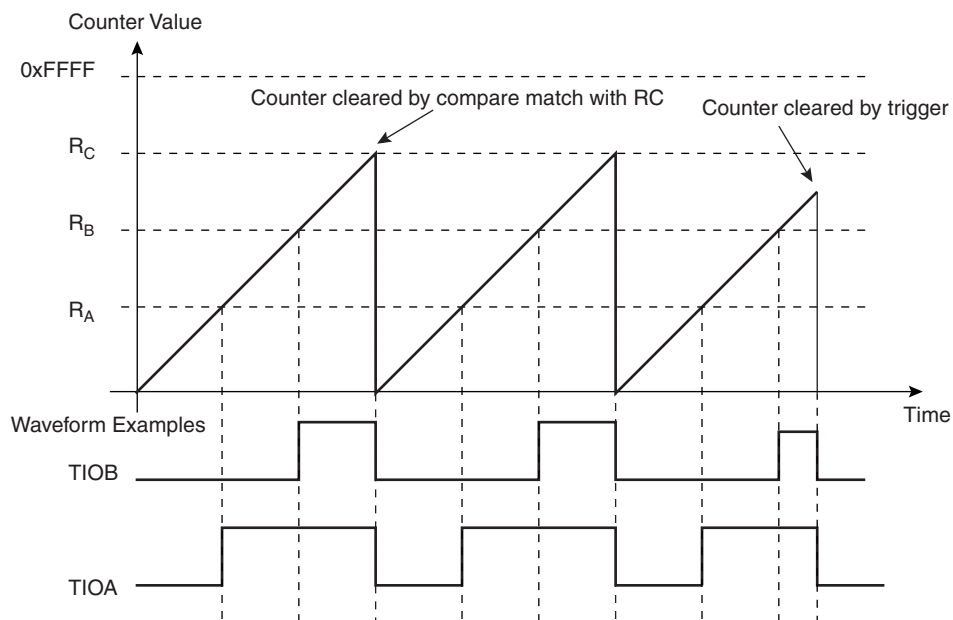
In addition, RC Compare can stop the counter clock (CPCSTOP = 1 in TC\_CMR) and/or disable the counter clock (CPCDIS = 1 in TC\_CMR).

**Figure 36-9.** WAVSEL = 10 Without Trigger





**Figure 36-10. WAVSEL = 10 With Trigger**



### 36.5.11.3 WAVSEL = 01

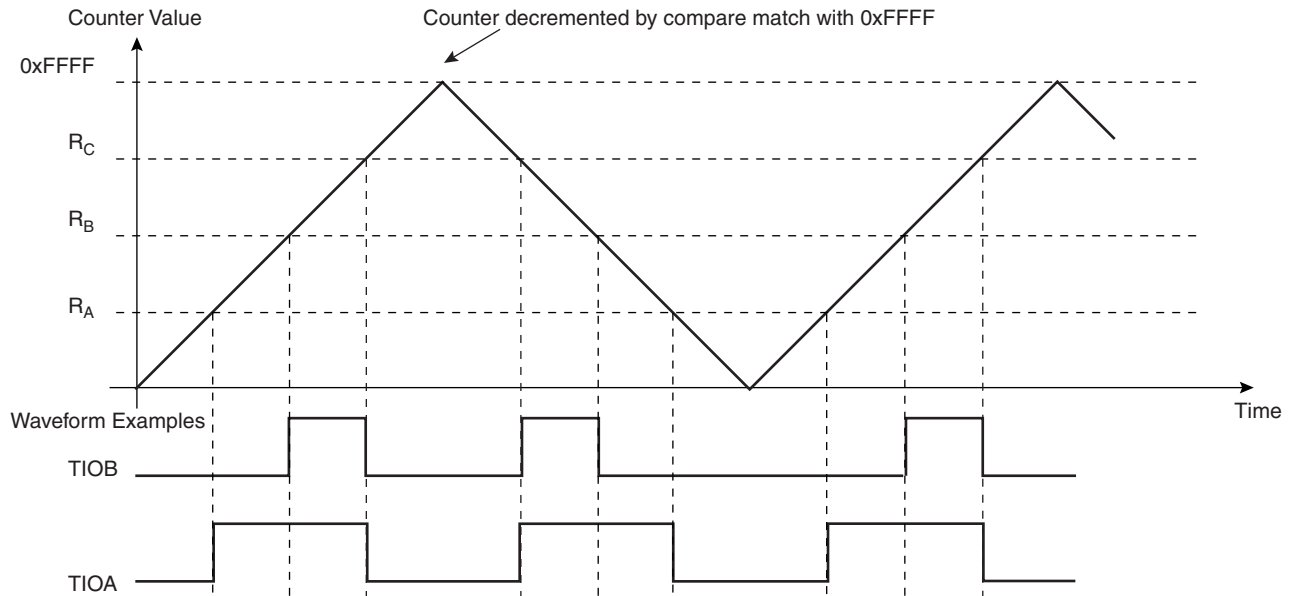
When WAVSEL = 01, the value of TC\_CV is incremented from 0 to 0xFFFF. Once 0xFFFF is reached, the value of TC\_CV is decremented to 0, then re-incremented to 0xFFFF and so on. See [Figure 36-11](#).

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 36-12](#).

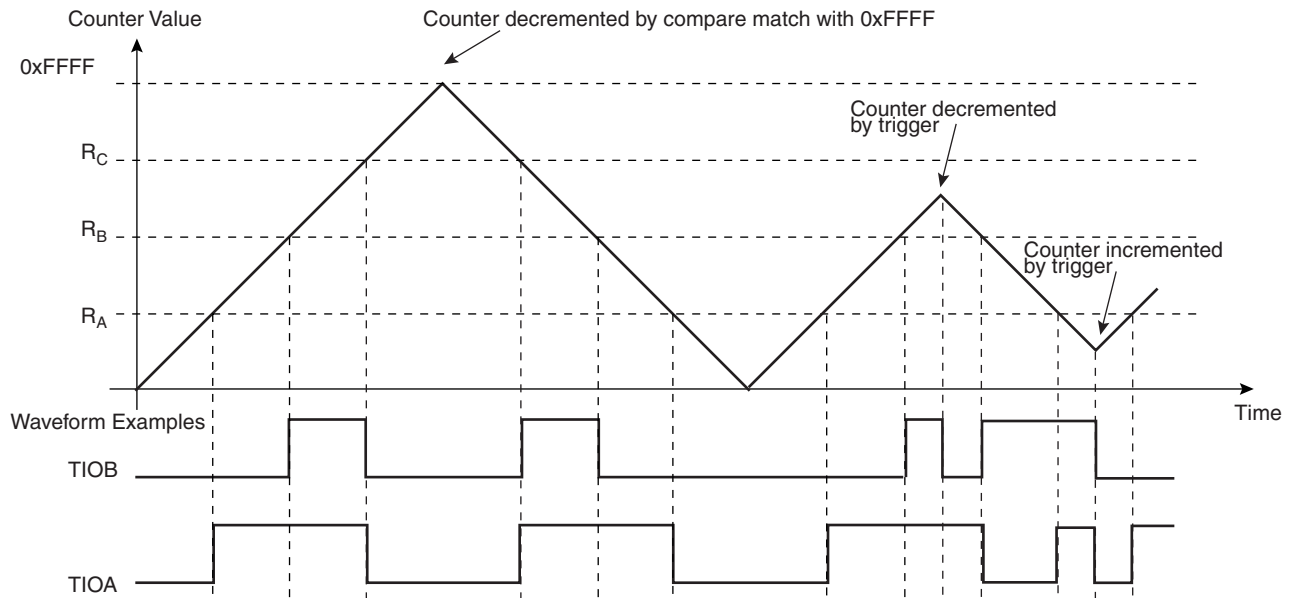
RC Compare cannot be programmed to generate a trigger in this configuration.

At the same time, RC Compare can stop the counter clock (CPCSTOP = 1) and/or disable the counter clock (CPCDIS = 1).

**Figure 36-11. WAVSEL = 01 Without Trigger**



**Figure 36-12. WAVSEL = 01 With Trigger**



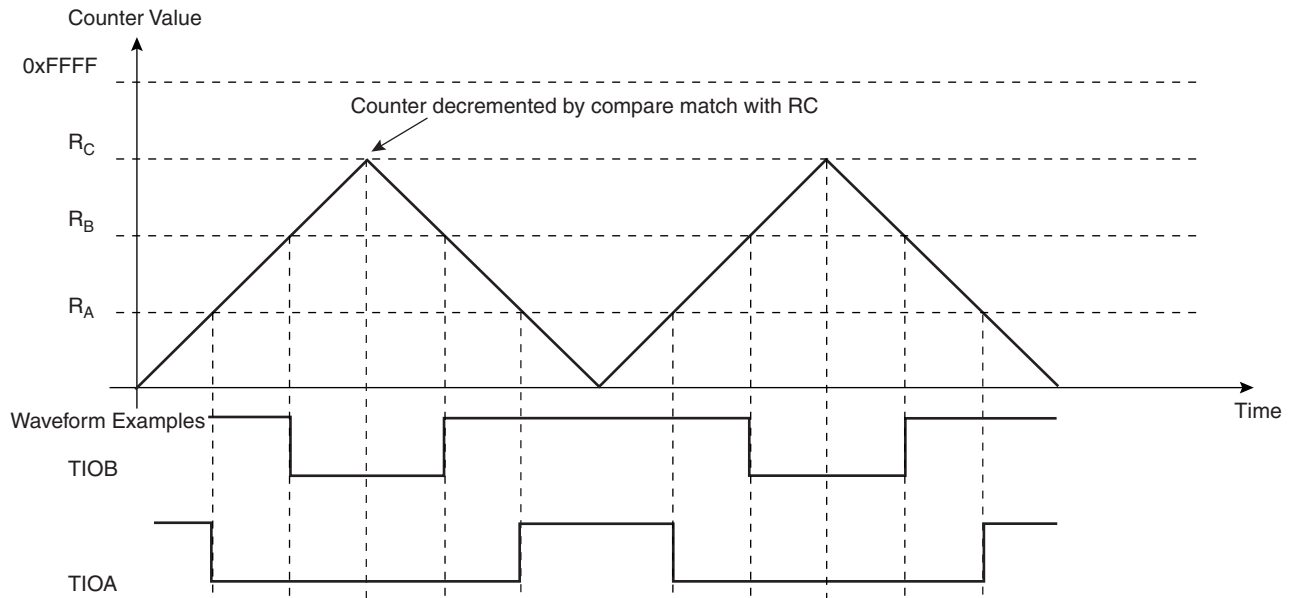
**36.5.11.4 WAVSEL = 11**

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 36-13](#).

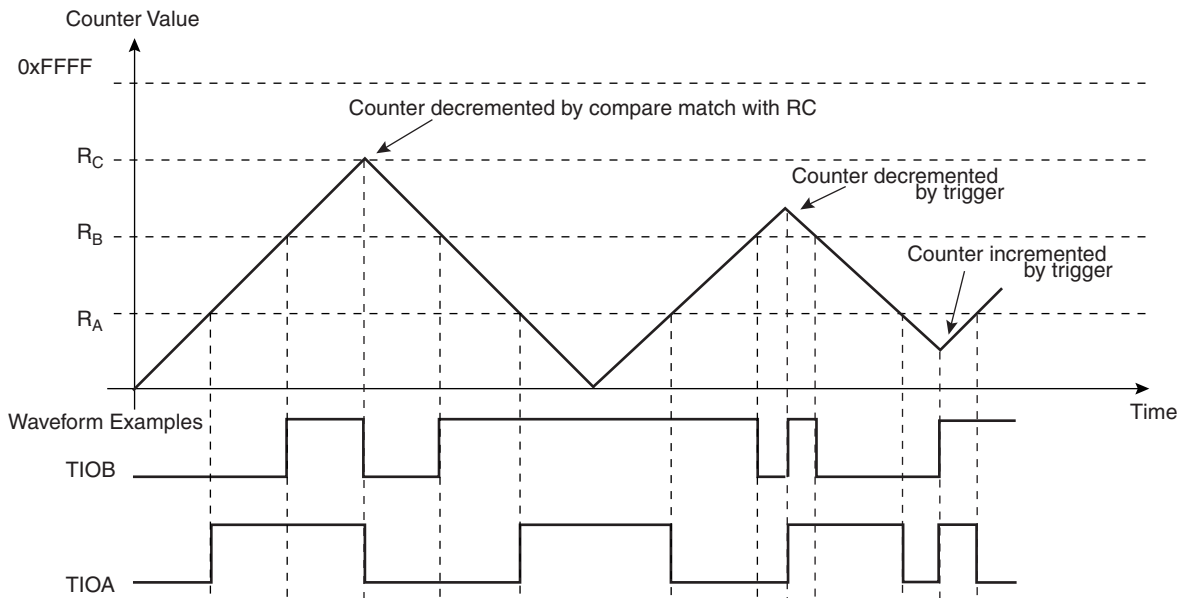
A trigger such as an external event or a software trigger can modify TC\_CV at any time. If a trigger occurs while TC\_CV is incrementing, TC\_CV then decrements. If a trigger is received while TC\_CV is decrementing, TC\_CV then increments. See [Figure 36-14](#).

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

**Figure 36-13. WAVSEL = 11 Without Trigger**



**Figure 36-14. WAVSEL = 11 With Trigger**



### 36.5.12 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 ENETR 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.

### 36.5.13 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.

## 36.6 Timer Counter (TC) User Interface

**Table 36-4.** TC Global Memory Map

Offset	Channel/Register	Name	Access	Reset Value
0x00	TC Channel 0		See <a href="#">Table 36-5</a>	
0x40	TC Channel 1		See <a href="#">Table 36-5</a>	
0x80	TC Channel 2		See <a href="#">Table 36-5</a>	
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 36-5](#). The offset of each of the channel registers in [Table 36-5](#) is in relation to the offset of the corresponding channel as mentioned in [Table 36-5](#).

**Table 36-5.** 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 <sup>(1)</sup>	0
0x18	Register B	TC_RB	Read/Write <sup>(1)</sup>	0
0x1C	Register C	TC_RC	Read/Write	0
0x20	Status Register	TC_SR	Read-only	0
0x24	Interrupt Enable Register	TC_IER	Write-only	–
0x28	Interrupt Disable Register	TC_IDR	Write-only	–
0x2C	Interrupt Mask Register	TC_IMR	Read-only	0
0xFC	Reserved	–	–	–

Notes: 1. Read-only if WAVE = 0

### 36.6.1 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.

## 36.6.2 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

### 36.6.3 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.



## 36.6.4 TC Channel Mode Register: Capture Mode

Register Name: TC\_CMCR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	LDRB		LDRA	
15	14	13	12	11	10	9	8
WAVE = 0	CPCTRG	–	–	–	ABETRG	ETRGDGD	
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

## 36.6.5 TC Channel Mode Register: Waveform Mode

Register Name: TC\_CMCR

Access Type: Read/Write

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

- **TCCLKS: Clock Selection**

TCCLKS			Clock Selected
0	0	0	TIMER_CLOCK1
0	0	1	TIMER_CLOCK2
0	1	0	TIMER_CLOCK3
0	1	1	TIMER_CLOCK4
1	0	0	TIMER_CLOCK5
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI: Clock Invert**

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

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

- **BURST: Burst Signal Selection**

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

- **CPCSTOP: Counter Clock Stopped with RC Compare**

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

1 = Counter clock is stopped when counter reaches RC.

- **CPCDIS: Counter Clock Disable with RC Compare**

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

1 = Counter clock is disabled when counter reaches RC.

- **EEVTEG: External Event Edge Selection**

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

- **EEVT: External Event Selection**

EEVT		Signal selected as external event	TIOB Direction
0	0	TIOB	input <sup>(1)</sup>
0	1	XC0	output
1	0	XC1	output
1	1	XC2	output

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

- **ENETR: External Event Trigger Enable**

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

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

- **WAVSEL: Waveform Selection**

WAVSEL		Effect
0	0	UP mode without automatic trigger on RC Compare
1	0	UP mode with automatic trigger on RC Compare
0	1	UPDOWN mode without automatic trigger on RC Compare
1	1	UPDOWN mode with automatic trigger on RC Compare

- **WAVE = 1**

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

1 = Waveform Mode is enabled.

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

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

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

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

- **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

## 36.6.6 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.

## 36.6.7 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.

### 36.6.8 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.

### 36.6.9 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.



## 36.6.10 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.

## 36.6.11 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.

### 36.6.12 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.

## 36.6.13 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.



## 37. DMA Controller (DMAC)

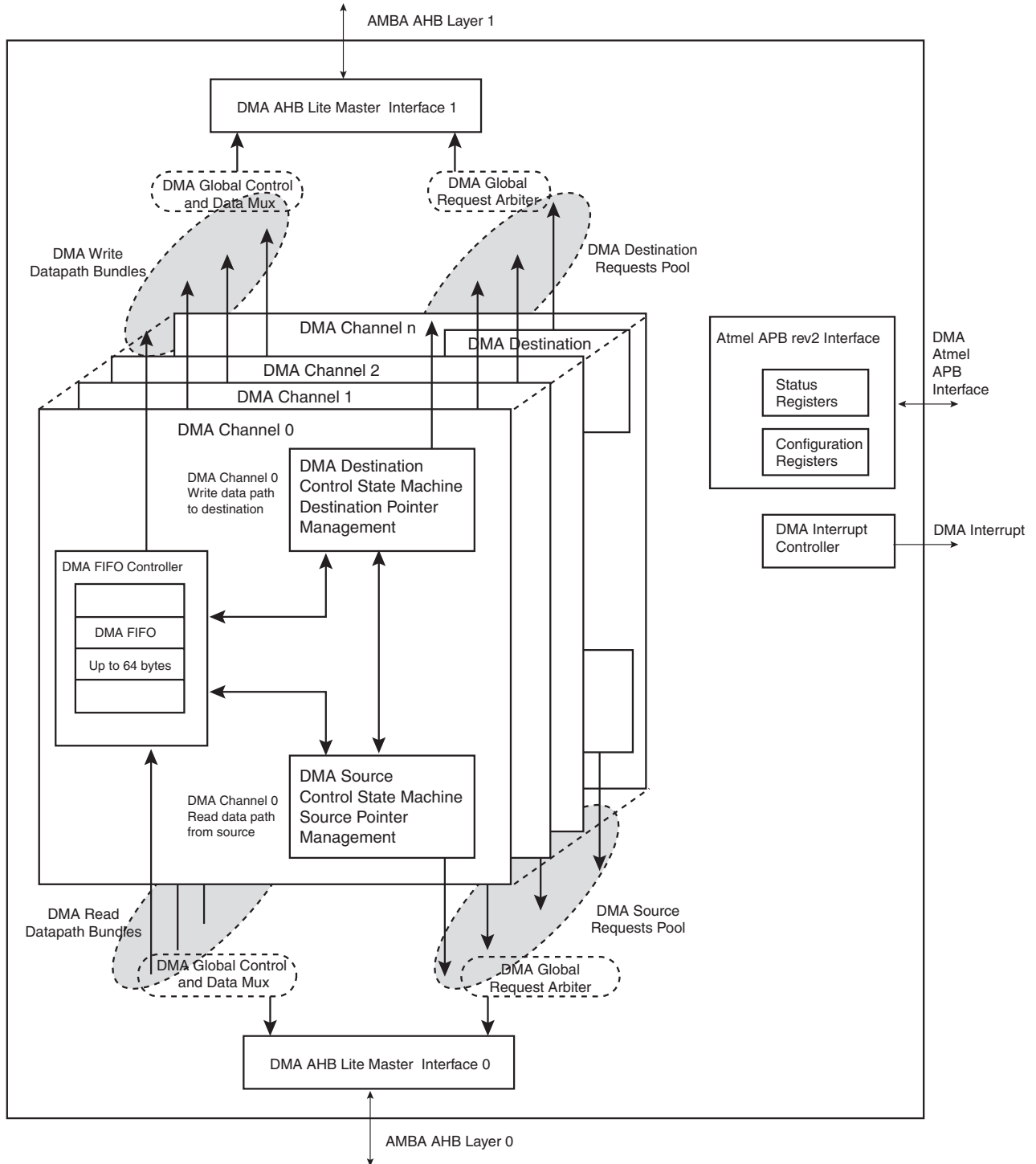
### 37.1 Description

The DMA Controller (DMAC) is an AHB-central DMA controller core that transfers data from a source peripheral to a destination peripheral over one or more AMBA buses. One channel is required for each source/destination pair. In the most basic configuration, the DMAC has one master interface and one channel. The master interface reads the data from a source and writes it to a destination. Two AMBA transfers are required for each DMAC data transfer. This is also known as a dual-access transfer.

The DMAC is programmed via the APB interface.

## 37.2 Block Diagram

Figure 37-1. DMA Controller (DMAC) Block Diagram





## 37.3 Functional Description

### 37.3.1 Basic Definitions

**Source peripheral:** Device on an AMBA layer from where the DMAC reads data, which is then stored in the channel FIFO. The source peripheral teams up with a destination peripheral to form a channel.

**Destination peripheral:** Device to which the DMAC writes the stored data from the FIFO (previously read from the source peripheral).

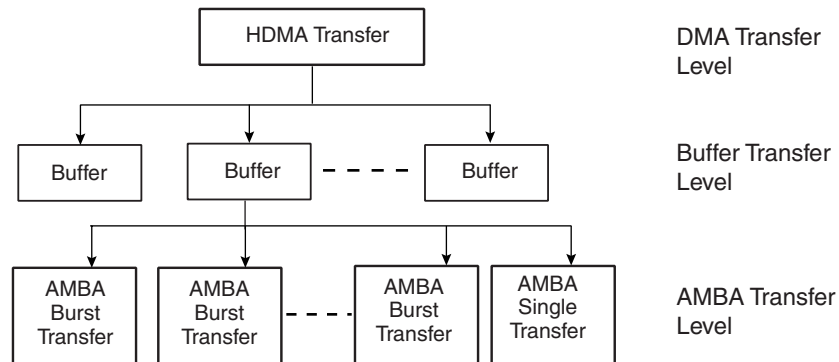
**Memory:** Source or destination that is always “ready” for a DMAC transfer and does not require a handshaking interface to interact with the DMAC.

**Channel:** Read/write datapath between a source peripheral on one configured AMBA layer and a destination peripheral on the same or different AMBA layer that occurs through the channel FIFO.

**Master interface:** DMAC is a master on the AHB bus reading data from the source and writing it to the destination over the AHB bus.

**Slave interface:** The APB interface over which the DMAC is programmed. The slave interface in practice could be on the same layer as any of the master interfaces or on a separate layer.

**Figure 37-2.** DMAC Transfer Hierarchy for Memory



**Buffer:** A buffer of DMAC data. The amount of data (length) is determined by the flow controller. For transfers between the DMAC and memory, a buffer is broken directly into a sequence of AMBA bursts and AMBA single transfers.

**DMAC transfer:** Software controls the number of buffers in a DMAC transfer. Once the DMAC transfer has completed, then hardware within the DMAC disables the channel and can generate an interrupt to signal the completion of the DMAC transfer. You can then re-program the channel for a new DMAC transfer.

**Single-buffer DMAC transfer:** Consists of a single buffer.

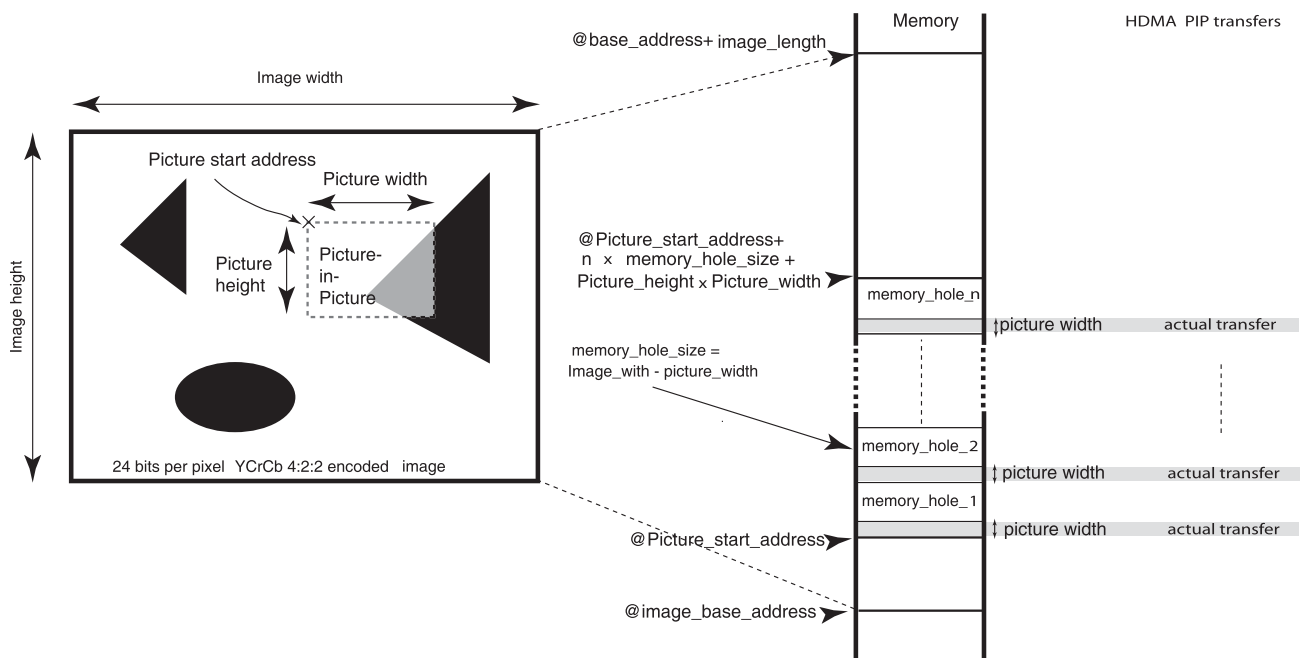
**Multi-buffer DMAC transfer:** A DMAC transfer may consist of multiple DMAC buffers. Multi-buffer DMAC transfers are supported through buffer chaining (linked list pointers), auto-reloading of channel registers, and contiguous buffers. The source and destination can independently select which method to use.

- **Linked lists (buffer chaining)** – A descriptor pointer (DSCR) points to the location in system memory where the next linked list item (LLI) exists. The LLI is a set of registers that describe the next buffer (buffer descriptor) and a descriptor pointer register. The DMAC fetches the LLI at the beginning of every buffer when buffer chaining is enabled.
- **Replay** – The DMAC automatically reloads the channel registers at the end of each buffers to the value when the channel was first enabled.
- **Contiguous buffers** – Where the address of the next buffer is selected to be a continuation from the end of the previous buffer.

Picture-in-Picture Mode: DMAC contains a picture-in-picture mode support. When this mode is enabled, addresses are automatically incremented by a programmable value when the DMAC channel transfer count reaches a user defined boundary.

Figure 37-3 on page 578 illustrates a memory mapped image 4:2:2 encoded located at `image_base_address` in memory. A user defined start address is defined at `Picture_start_address`. The incremented value is set to `memory_hole_size = image_width - picture_width`, and the boundary is set to `picture_width`.

**Figure 37-3.** Picture-In-Picture Mode Support



**Bus locking:** Software can program a channel to maintain control of the AMBA bus by asserting `hmastlock` for the duration of a DMAC transfer. Channel locking is asserted for the duration of bus locking at a minimum.

### 37.3.2 Memory Peripherals

Figure 37-2 on page 577 shows the DMAC transfer hierarchy of the DMAC for a memory peripheral. There is no handshaking interface with the DMAC, and therefore the memory peripheral can never be a flow controller. Once the channel is enabled, the transfer proceeds immediately without waiting for a transaction request. The alternative to not having a transaction-level hand-

shaking interface is to allow the DMAC to attempt AMBA transfers to the peripheral once the channel is enabled. If the peripheral slave cannot accept these AMBA transfers, it inserts wait states onto the bus until it is ready; it is not recommended that more than 16 wait states be inserted onto the bus.

### 37.3.3 DMAC Transfer Types

A DMAC transfer may consist of single or multi-buffers transfers. On successive buffers of a multi-buffer transfer, the DMAC\_SADDRx/DMAC\_DADDRx registers in the DMAC are re-programmed using either of the following methods:

- Buffer chaining using linked lists
- Replay mode
- Contiguous address between buffers

On successive buffers of a multi-buffer transfer, the DMAC\_CTRLAx and DMAC\_CTRLBx registers in the DMAC are re-programmed using either of the following methods:

- Buffer chaining using linked lists
- Replay mode

When buffer chaining, using linked lists is the multi-buffer method of choice, and on successive buffers, the DMAC\_DSCRx register in the DMAC is re-programmed using the following method:

- Buffer chaining using linked lists

A buffer descriptor (LLI) consists of following registers, DMAC\_SADDRx, DMAC\_DADDRx, DMAC\_DSCRx, DMAC\_CTRLAx, DMAC\_CTRLBx. These registers, along with the DMAC\_CFGx register, are used by the DMAC to set up and describe the buffer transfer.

#### 37.3.3.1 Multi-buffer Transfers

#### 37.3.3.2 Buffer Chaining Using Linked Lists

In this case, the DMAC re-programs the channel registers prior to the start of each buffer by fetching the buffer descriptor for that buffer from system memory. This is known as an LLI update.

DMAC buffer chaining is supported by using a Descriptor Pointer register (DMAC\_DSCRx) that stores the address in memory of the next buffer descriptor. Each buffer descriptor contains the corresponding buffer descriptor (DMAC\_SADDRx, DMAC\_DADDRx, DMAC\_DSCRx, DMAC\_CTRLAx, DMAC\_CTRLBx).

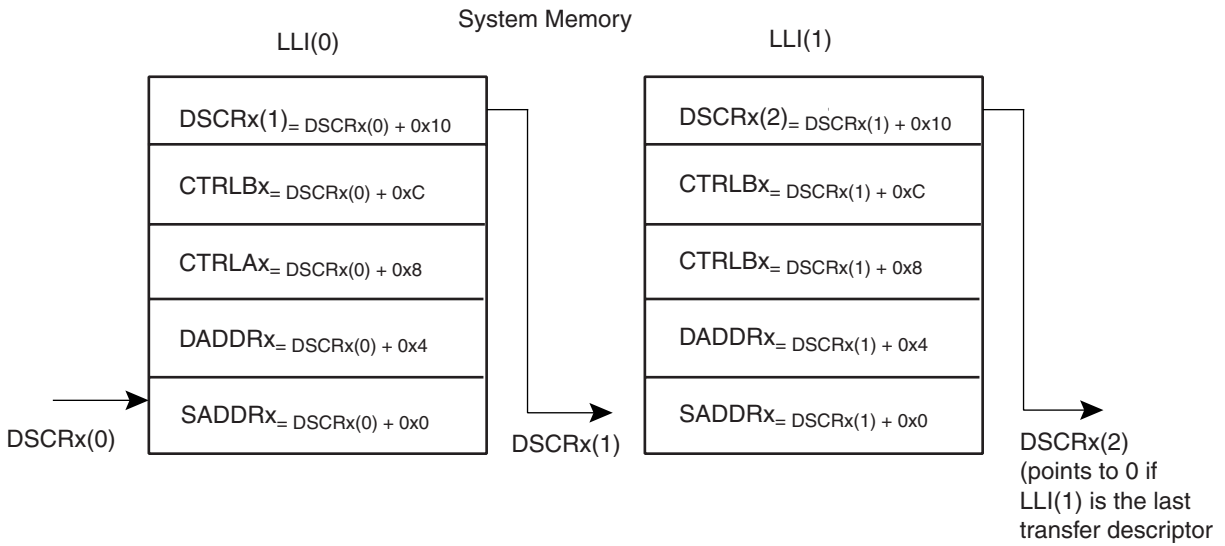
To set up buffer chaining, a sequence of linked lists must be programmed in memory.

The DMAC\_SADDRx, DMAC\_DADDRx, DMAC\_DSCRx, DMAC\_CTRLAx and DMAC\_CTRLBx registers are fetched from system memory on an LLI update. The updated content of the DMAC\_CTRLAx register is written back to memory on buffer completion. [Figure 37-4 on page 580](#) shows how to use chained linked lists in memory to define multi-buffer transfers using buffer chaining.

The Linked List multi-buffer transfer is initiated by programming DMAC\_DSCRx with DSCRx(0) (LLI(0) base address) and DMAC\_CTRLBx register with both SRC\_DSCR and DST\_DSCR set to 0. Other fields and registers are ignored and overwritten when the descriptor is retrieved from memory.

The last transfer descriptor must be written to memory with its next descriptor address set to 0.

**Figure 37-4.** Multi Buffer Transfer Using Linked List



### 37.3.3.3 Programming DMAC for Multiple Buffer Transfers

**Table 37-1.** Multiple Buffers Transfer Management Table

Transfer Type	AUTO	SRC_REP	DST_REP	SRC_DSCR	DST_DSCR	BTSIZE	SADDR	DADDR	Other Fields
1) Single Buffer or Last buffer of a multiple buffer transfer	0	–	–	1	1	USR	USR	USR	USR
2) Multi Buffer transfer with contiguous DADDR	0	–	0	0	1	LLI	LLI	CONT	LLI
3) Multi Buffer transfer with contiguous SADDR	0	0	–	1	0	LLI	CONT	LLI	LLI
4) Multi Buffer transfer with LLI support	0	–	–	0	0	LLI	LLI	LLI	LLI
5) Multi Buffer transfer with DADDR reloaded	0	–	1	0	1	LLI	LLI	REP	LLI
6) Multi Buffer transfer with SADDR reloaded	0	1	–	1	0	LLI	REP	LLI	LLI
7) Multi Buffer transfer with BTSIZE reloaded and contiguous DADDR	1	–	0	0	1	REP	LLI	CONT	LLI
8) Multi Buffer transfer with BTSIZE reloaded and contiguous SADDR	1	0	–	1	0	REP	CONT	LLI	LLI
9) Automatic mode channel is stalling BTsize is reloaded	1	0	0	1	1	REP	CONT	CONT	REP
10) Automatic mode BTSIZE, SADDR and DADDR reloaded	1	1	1	1	1	REP	REP	REP	REP
11) Automatic mode BTSIZE, SADDR reloaded and DADDR contiguous	1	1	0	1	1	REP	REP	CONT	REP

- Notes:
1. USR means that the register field is manually programmed by the user.
  2. CONT means that address are contiguous.
  3. REP means that the register field is updated with its previous value. If the transfer is the first one, then the user must manually program the value.
  4. Channel stalled is true if the relevant BTC interrupt is not masked.
  5. LLI means that the register field is updated with the content of the linked list item.

### 37.3.3.4 Replay Mode of Channel Registers

During automatic replay mode, the channel registers are reloaded with their initial values at the completion of each buffer and the new values used for the new buffer. Depending on the row number in [Table 37-1 on page 581](#), some or all of the DMAC\_SADDRx, DMAC\_DADDRx, DMAC\_CTRLAx and DMAC\_CTRLBx channel registers are reloaded from their initial value at the start of a buffer transfer.

### 37.3.3.5 Contiguous Address Between Buffers

In this case, the address between successive buffers is selected to be a continuation from the end of the previous buffer. Enabling the source or destination address to be contiguous between

buffers is a function of DMAC\_CTRLAx.SRC\_DSCR, DMAC\_CFGx.SRC\_REP, DMAC\_CTRLAx.DST\_DSCR and DMAC\_CFGx.DST\_REP registers.

### 37.3.3.6 Suspension of Transfers Between buffers

At the end of every buffer transfer, an end of buffer interrupt is asserted if:

- the channel buffer interrupt is unmasked, DMAC\_EBCIMR.BTC[n] = '1', where n is the channel number.

Note: The buffer complete interrupt is generated at the completion of the buffer transfer to the destination.

At the end of a chain of multiple buffers, an end of linked list interrupt is asserted if:

- the channel end of chained buffer interrupt is unmasked, DMAC\_EBCIMR.CBTC[n] = '1', when n is the channel number.

### 37.3.3.7 Ending Multi-buffer Transfers

All multi-buffer transfers must end as shown in Row 1 of [Table 37-1 on page 581](#). At the end of every buffer transfer, the DMAC samples the row number, and if the DMAC is in Row 1 state, then the previous buffer transferred was the last buffer and the DMAC transfer is terminated.

For rows 9, 10 and 11 of [Table 37-1 on page 581](#), (DMAC\_DSCRx = 0 and DMAC\_CTRLBx.AUTO is set), multi-buffer DMAC transfers continue until the automatic mode is disabled by writing a '1' in DMAC\_CTRLBx.AUTO bit. This bit should be programmed to zero in the end of buffer interrupt service routine that services the next-to-last buffer transfer. This puts the DMAC into Row 1 state.

For rows 2, 3, 4, 5, and 6 (DMAC\_CTRLBx.AUTO cleared) the user must setup the last buffer descriptor in memory such that both LLI.DMAC\_CTRLBx.SRC\_DSCR and LLI.DMAC\_CTRLBx.DST\_DSCR are one and LLI.DMAC\_DSCRx is set to 0.

## 37.3.4 Programming a Channel

Four registers, the DMAC\_DSCR<sub>x</sub>, the DMAC\_CTRLA<sub>x</sub>, the DMAC\_CTRLB<sub>x</sub> and DMAC\_CFG<sub>x</sub>, need to be programmed to set up whether single or multi-buffer transfers take place, and which type of multi-buffer transfer is used. The different transfer types are shown in [Table 37-1 on page 581](#).

The “BTSIZE, SADDR and DADDR” columns indicate where the values of DMAC\_SAR<sub>x</sub>, DMAC\_DAR<sub>x</sub>, DMAC\_CTL<sub>x</sub>, and DMAC\_LLP<sub>x</sub> are obtained for the next buffer transfer when multi-buffer DMAC transfers are enabled.

### 37.3.4.1 Programming Examples

#### 37.3.4.2 Single-buffer Transfer (Row 1)

1. Read the Channel Handler Status Register DMAC\_CHSR.ENABLE Field to choose a free (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMAC transfer by reading the interrupt status register, DMAC\_EBCISR.
3. Program the following channel registers:
  - a. Write the starting source address in the DMAC\_SADDR<sub>x</sub> register for channel x.
  - b. Write the starting destination address in the DMAC\_DADDR<sub>x</sub> register for channel x.
  - c. Program DMAC\_CTRLA<sub>x</sub>, DMAC\_CTRLB<sub>x</sub> and DMAC\_CFG<sub>x</sub> according to Row 1 as shown in [Table 37-1 on page 581](#). Program the DMAC\_CTRLB<sub>x</sub> register with both DST\_DSCR and SRC\_DSCR fields set to one and AUTO field set to 0.
  - d. Write the control information for the DMAC transfer in the DMAC\_CTRLA<sub>x</sub> and DMAC\_CTRLB<sub>x</sub> registers for channel x. For example, in the register, you can program the following:
    - Set up the transfer characteristics, such as:
      - Transfer width for the source in the SRC\_WIDTH field.
      - Transfer width for the destination in the DST\_WIDTH field.
      - Source AHB Master interface layer in the SIF field where source resides.
      - Destination AHB Master Interface layer in the DIF field where destination resides.
      - Incrementing/decrementing or fixed address for source in SRC\_INC field.
      - Incrementing/decrementing or fixed address for destination in DST\_INC field.
  - e. Write the channel configuration information into the DMAC\_CFG<sub>x</sub> register for channel x.
  - f. If source picture-in-picture mode is enabled (DMAC\_CTRLB<sub>x</sub>.SRC\_PIP is enabled), program the DMAC\_SPIP<sub>x</sub> register for channel x.
  - g. If destination picture-in-picture mode is enabled (DMAC\_CTRLB<sub>x</sub>.DST\_PIP is enabled), program the DMAC\_DPIP<sub>x</sub> register for channel x.
4. After the DMAC selected channel has been programmed, enable the channel by writing a ‘1’ to the DMAC\_CHER.ENABLE[n] bit, where n is the channel number. Make sure that bit 0 of DMAC\_EN.ENABLE register is enabled.
5. Once the transfer completes, hardware sets the interrupts and disables the channel. At this time you can either respond to the buffer Complete or Transfer Complete interrupts, or poll for the Channel Handler Status Register (DMAC\_CHSR.ENABLE[n]) bit until it is cleared by hardware, to detect when the transfer is complete.

## 37.3.4.3 Multi-buffer Transfer with Linked List for Source and Linked List for Destination (Row 4)

1. Read the Channel Enable register to choose a free (disabled) channel.
2. Set up the chain of Linked List Items (otherwise known as buffer descriptors) in memory. Write the control information in the LLI.DMAC\_CTRLAx and LLI.DMAC\_CTRLBx registers location of the buffer descriptor for each LLI in memory (see [Figure 37-5 on page 585](#)) for channel x. For example, in the register, you can program the following:
  - h. Set up the transfer characteristics, such as:
    - i. Transfer width for the source in the SRC\_WIDTH field.
    - ii. Transfer width for the destination in the DST\_WIDTH field.
    - iii. Source AHB master interface layer in the SIF field where source resides.
    - iv. Destination AHB master interface layer in the DIF field where destination resides.
    - v. Incrementing/decrementing or fixed address for source in SRC\_INCR field.
    - vi. Incrementing/decrementing or fixed address for destination DST\_INCR field.
3. Write the channel configuration information into the DMAC\_CFGx register for channel x.
4. Make sure that the LLI.DMAC\_CTRLBx register locations of all LLI entries in memory (except the last) are set as shown in Row 4 of [Table 37-1 on page 581](#). The LLI.DMAC\_CTRLBx register of the last Linked List Item must be set as described in Row 1 of [Table 37-1](#). [Figure 37-4 on page 580](#) shows a Linked List example with two list items.
5. Make sure that the LLI.DMAC\_DSCRx register locations of all LLI entries in memory (except the last) are non-zero and point to the base address of the next Linked List Item.
6. Make sure that the LLI.DMAC\_SADDRx/LLI.DMAC\_DADDRx register locations of all LLI entries in memory point to the start source/destination buffer address preceding that LLI fetch.
7. Make sure that the LLI.DMAC\_CTRLAx.DONE field of the LLI.DMAC\_CTRLAx register locations of all LLI entries in memory are cleared.
8. If source picture-picture mode is enabled (DMAC\_CTRLBx.SRC\_PIP is enabled), program the DMAC\_SPIPx register for channel x.
9. If destination picture-in-picture is enabled (DMAC\_CTRLBx.DST\_PIP is enabled), program the DMAC\_DPIPx register for channel x.
10. Clear any pending interrupts on the channel from the previous DMAC transfer by reading the status register: DMAC\_EBCISR.
11. Program the DMAC\_CTRLBx, DMAC\_CFGx registers according to Row 4 as shown in [Table 37-1 on page 581](#).
12. Program the DMAC\_DSCRx register with DMAC\_DSCRx(0), the pointer to the first Linked List item.
13. Finally, enable the channel by writing a '1' to the DMAC\_CHER.ENABLE[n] bit, where n is the channel number. The transfer is performed.
14. The DMAC fetches the first LLI from the location pointed to by DMAC\_DSCRx(0).

Note: The LLI.DMAC\_SADDRx, LLI.DMAC\_DADDRx, LLI.DMAC\_DSCRx, LLI.DMAC\_CTRLAx and LLI.DMAC\_CTRLBx registers are fetched. The DMAC automatically reprograms the DMAC\_SADDRx, DMAC\_DADDRx, DMAC\_DSCRx, DMAC\_CTRLBx and DMAC\_CTRLAx channel registers from the DMAC\_DSCRx(0).

15. Once the buffer of data is transferred, the DMAC\_CTRLAx register is written out to system memory at the same location and on the same layer (DMAC\_DSCRx.DSCR\_IF) where it was originally fetched, that is, the location of the DMAC\_CTRLAx register of

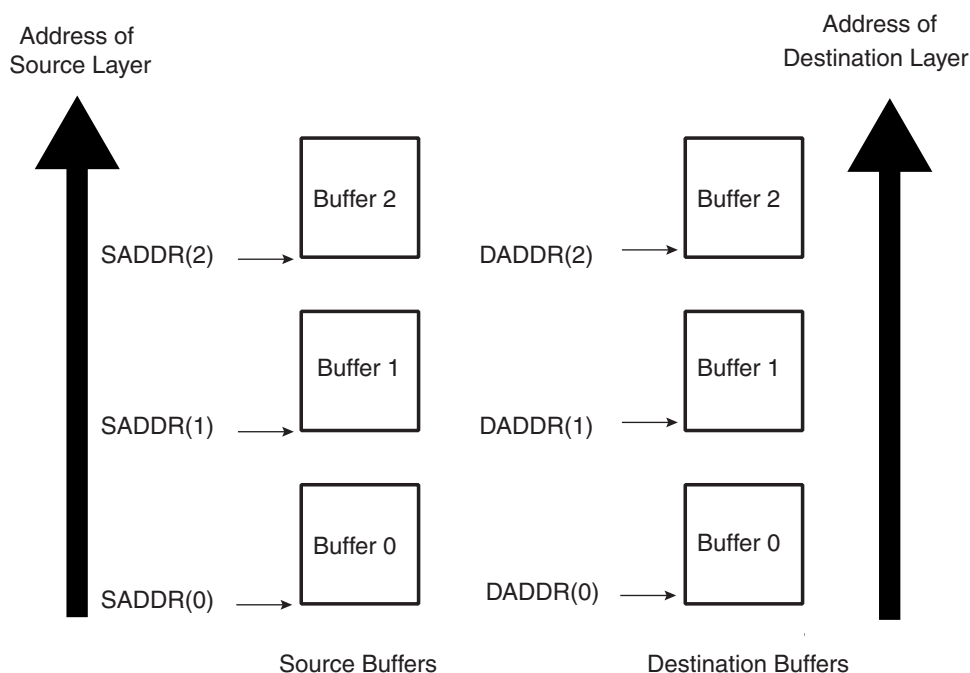


the linked list item fetched prior to the start of the buffer transfer. Only DMAC\_CTRLAx register is written out because only the DMAC\_CTRLAx.BTSIZE and DMAC\_CTRLAx.DONE bits have been updated by DMAC hardware. Additionally, the DMAC\_CTRLAx.DONE bit is asserted when the buffer transfer has completed.

Note: Do not poll the DMAC\_CTRLAx.DONE bit in the DMAC memory map. Instead, poll the LLI.DMAC\_CTRLAx.DONE bit in the LLI for that buffer. If the poll LLI.DMAC\_CTRLAx.DONE bit is asserted, then this buffer transfer has completed. This LLI.DMAC\_CTRLAx.DONE bit was cleared at the start of the transfer.

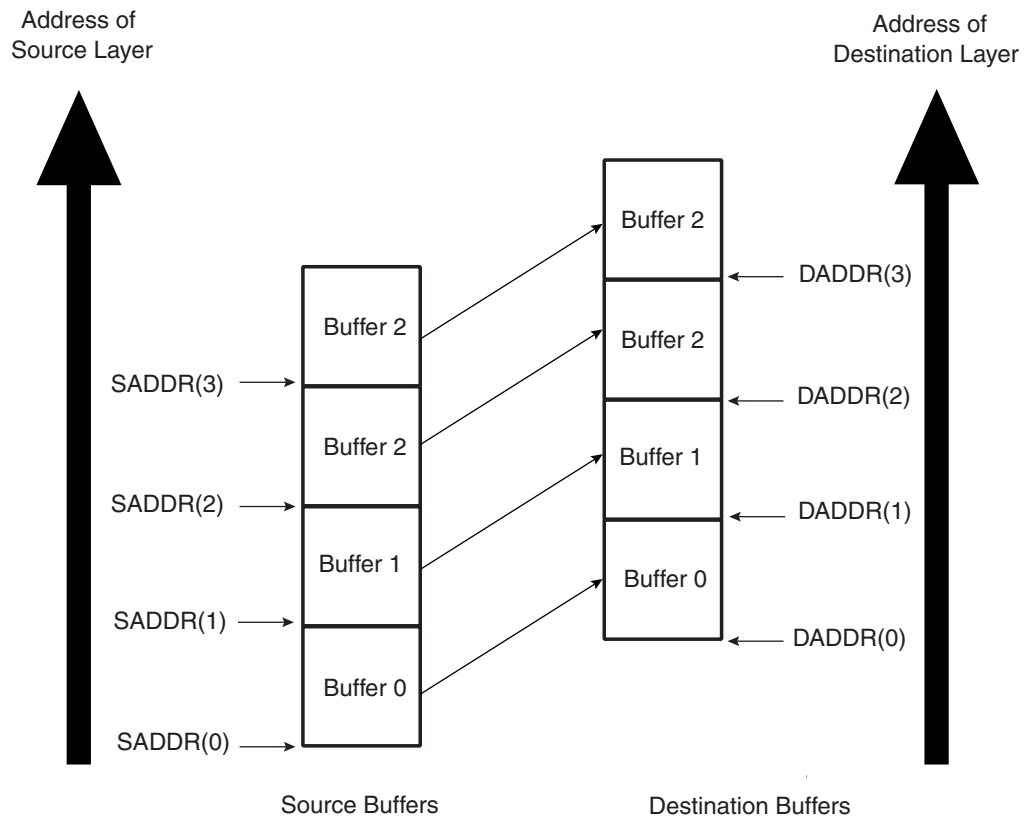
- The DMAC does not wait for the buffer interrupt to be cleared, but continues fetching the next LLI from the memory location pointed to by current DMAC\_DSCRx register and automatically reprograms the DMAC\_SADDRx, DMAC\_DADDRx, DMAC\_DSCRx, DMAC\_CTRLAx and DMAC\_CTRLBx channel registers. The DMAC transfer continues until the DMAC determines that the DMAC\_CTRLBx and DMAC\_DSCRx registers at the end of a buffer transfer match described in Row 1 of [Table 37-1 on page 581](#). The DMAC then knows that the previous buffer transferred was the last buffer in the DMAC transfer. The DMAC transfer might look like that shown in [Figure 37-5 on page 585](#).

**Figure 37-5.** Multi-buffer with Linked List Address for Source and Destination



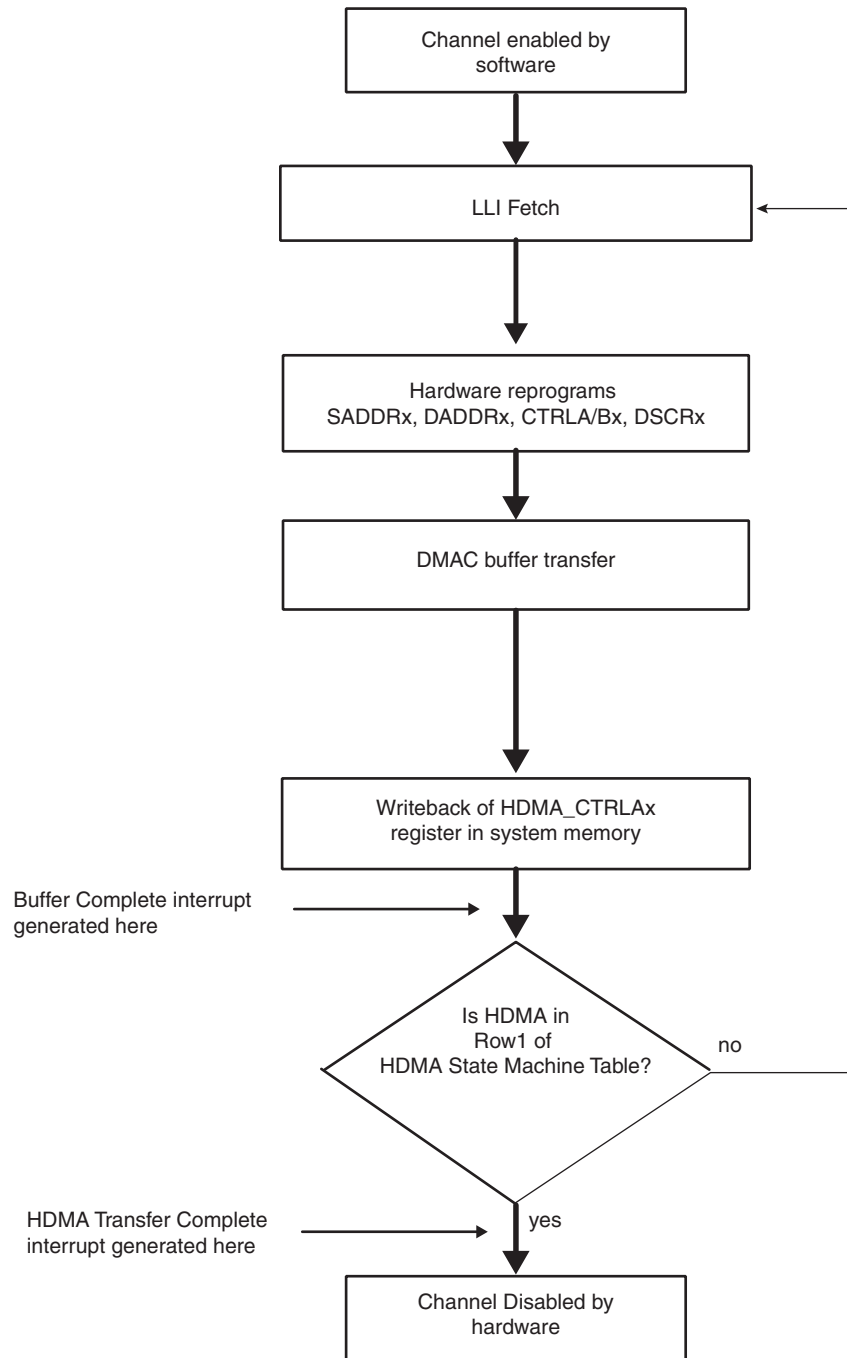
If the user needs to execute a DMAC transfer where the source and destination address are contiguous but the amount of data to be transferred is greater than the maximum buffer size DMAC\_CTRLAx.BTSIZE, then this can be achieved using the type of multi-buffer transfer as shown in [Figure 37-6 on page 586](#).

**Figure 37-6.** Multi-buffer with Linked Address for Source and Destination Buffers are Contiguous



The DMAC transfer flow is shown in [Figure 37-7 on page 587](#).

**Figure 37-7.** DMAC Transfer Flow for Source and Destination Linked List Address



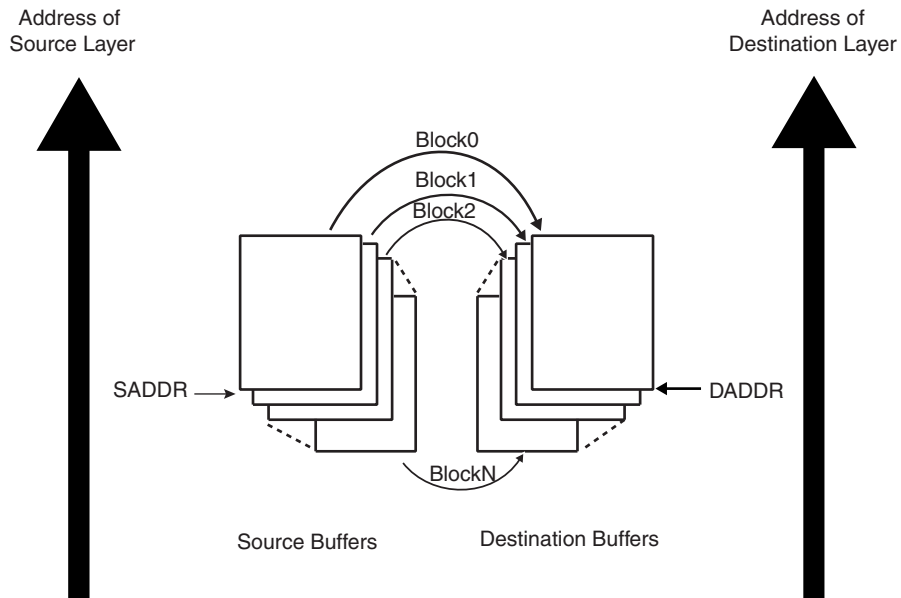
#### 37.3.4.4 Multi-buffer Transfer with Source Address Auto-reloaded and Destination Address Auto-reloaded (Row 10)

1. Read the Channel Enable register to choose an available (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMAC transfer by reading the interrupt status register. Program the following channel registers:

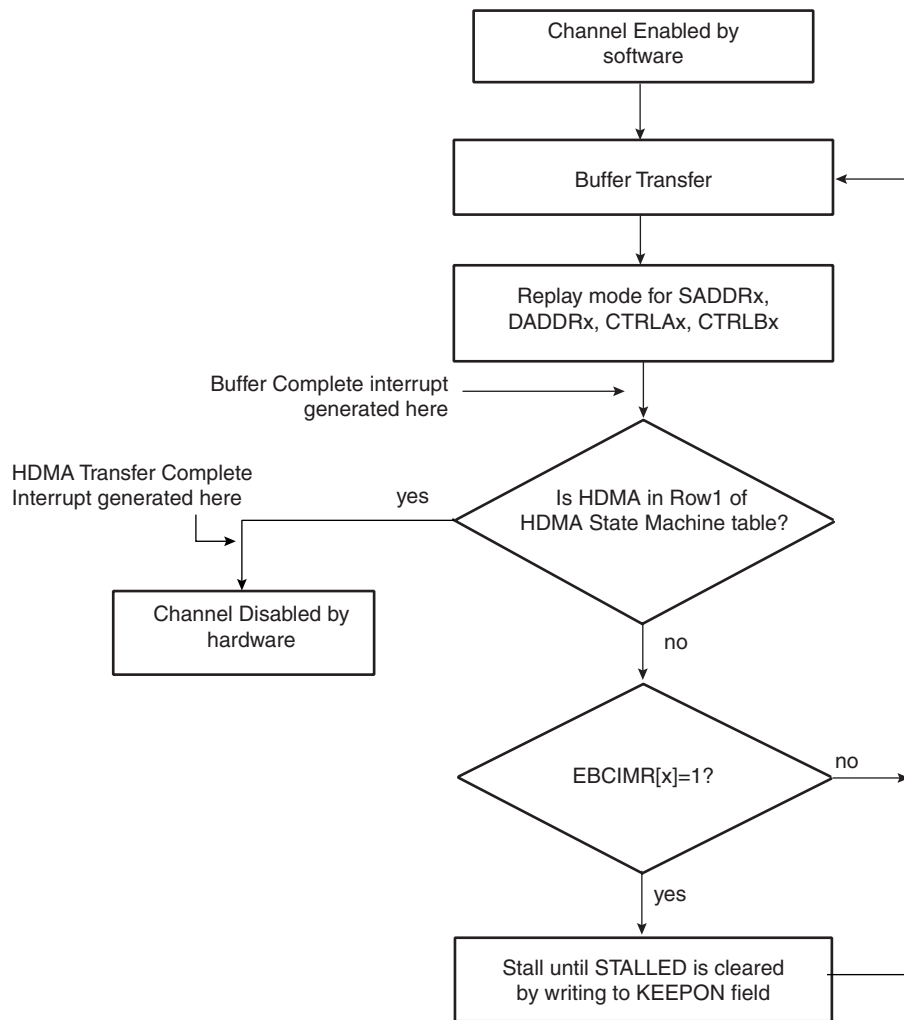
- a. Write the starting source address in the DMAC\_SADDRx register for channel x.
  - b. Write the starting destination address in the DMAC\_DADDRx register for channel x.
  - c. Program DMAC\_CTRLAx, DMAC\_CTRLBx and DMAC\_CFGx according to Row 10 as shown in [Table 37-1 on page 581](#). Program the DMAC\_DSCRx register with '0'.
  - d. Write the control information for the DMAC transfer in the DMAC\_CTRLAx and DMAC\_CTRLBx register for channel x. For example, in the register, you can program the following:
    - Set up the transfer characteristics, such as:
    - Transfer width for the source in the SRC\_WIDTH field.
    - Transfer width for the destination in the DST\_WIDTH field.
    - Source AHB master interface layer in the SIF field where source resides.
    - Destination AHB master interface layer in the DIF field where destination resides.
    - Incrementing/decrementing or fixed address for source in SRC\_INCR field.
    - Incrementing/decrementing or fixed address for destination in DST\_INCR field.
  - e. If source picture-in-picture mode is enabled (DMAC\_CTRLBx.SPIP is enabled), program the DMAC\_SPIPx register for channel x.
  - f. If destination picture-in-picture is enabled (DMAC\_CTRLBx.DPIP), program the DMAC\_DPIPx register for channel x.
  - g. Write the channel configuration information into the DMAC\_CFGx register for channel x. Ensure that the reload bits, DMAC\_CFGx.SRC\_REP, DMAC\_CFGx.DST\_REP and DMAC\_CTRLBx.AUTO are enabled.
3. After the DMAC selected channel has been programmed, enable the channel by writing a '1' to the DMAC\_CHER.ENABLE[n] bit where n is the channel number. Make sure that bit 0 of the DMAC\_EN register is enabled.
  4. When the buffer transfer has completed, the DMAC reloads the DMAC\_SADDRx, DMAC\_DADDRx and DMAC\_CTRLAx registers. Hardware sets the buffer Complete interrupt. The DMAC then samples the row number as shown in [Table 37-1 on page 581](#). If the DMAC is in Row 1, then the DMAC transfer has completed. Hardware sets the transfer complete interrupt and disables the channel. So you can either respond to the Buffer Complete or Chained buffer transfer Complete interrupts, or poll for the Channel Enable in the Channel Status Register (DMAC\_CHSR.ENABLE[n]) until it is disabled, to detect when the transfer is complete. If the DMAC is not in Row 1, the next step is performed.
  5. The DMAC transfer proceeds as follows:
    - a. If interrupts is un-masked (DMAC\_EBCIMR.BTC[x] = '1', where x is the channel number) hardware sets the buffer complete interrupt when the buffer transfer has completed. It then stalls until the STALLED[n] bit of DMAC\_CHSR register is cleared by software, writing '1' to DMAC\_CHER.KEEPON[n] bit where n is the channel number. If the next buffer is to be the last buffer in the DMAC transfer, then the buffer complete ISR (interrupt service routine) should clear the automatic mode bit in the DMAC\_CTRLBx.AUTO bit. This put the DMAC into Row 1 as shown in [Table 37-1 on page 581](#). If the next buffer is not the last buffer in the DMAC transfer, then the reload bits should remain enabled to keep the DMAC in Row 4.
    - b. If the buffer complete interrupt is masked (DMAC\_EBCIMR.BTC[x] = '1', where x is the channel number), then hardware does not stall until it detects a write to the buffer complete interrupt enable register DMAC\_EBCIER register but starts the next buffer transfer immediately. In this case software must clear the automatic

mode bit in the DMAC\_CTRLB to put the DMAC into ROW 1 of [Table 37-1 on page 581](#) before the last buffer of the DMAC transfer has completed. The transfer is similar to that shown in [Figure 37-8 on page 589](#). The DMAC transfer flow is shown in [Figure 37-9 on page 590](#).

**Figure 37-8.** Multi-buffer DMAC Transfer with Source and Destination Address Auto-reloaded



**Figure 37-9.** DMAC Transfer Flow for Source and Destination Address Auto-reloaded



### 37.3.4.5 Multi-buffer Transfer with Source Address Auto-reloaded and Linked List Destination Address (Row 6)

1. Read the Channel Enable register to choose a free (disabled) channel.
2. Set up the chain of linked list items (otherwise known as buffer descriptors) in memory. Write the control information in the LLI.DMAC\_CTRLAx and DMAC\_CTRLBx registers location of the buffer descriptor for each LLI in memory for channel x. For example, in the register you can program the following:
  - c. Set up the transfer characteristics, such as:
    - i. Transfer width for the source in the SRC\_WIDTH field.
    - ii. Transfer width for the destination in the DST\_WIDTH field.
    - iii. Source AHB master interface layer in the SIF field where source resides.
    - iv. Destination AHB master interface layer in the DIF field where destination resides.
    - v. Incrementing/decrementing or fixed address for source in SRC\_INCR field.
    - vi. Incrementing/decrementing or fixed address for destination DST\_INCR field.
3. Write the starting source address in the DMAC\_SADDRx register for channel x.

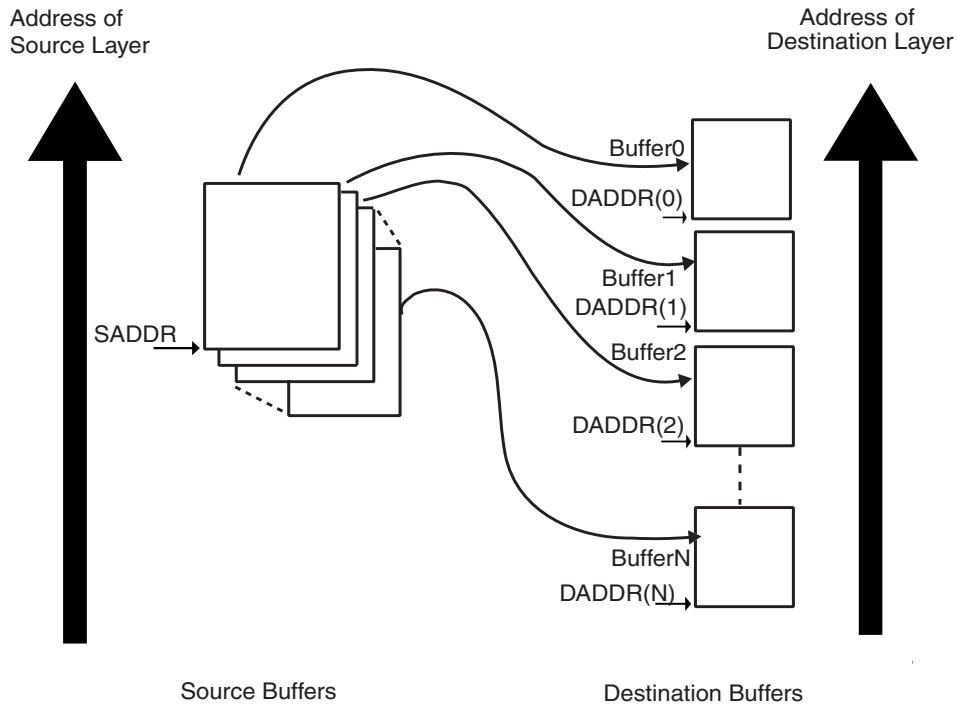
Note: The values in the LLI.DMAC\_SADDRx register locations of each of the Linked List Items (LLIs) setup up in memory, although fetched during a LLI fetch, are not used.

4. Write the channel configuration information into the DMAC\_CFGx register for channel x.
  5. Make sure that the LLI.DMAC\_CTRLBx register locations of all LLIs in memory (except the last) are set as shown in Row 6 of [Table 37-1 on page 581](#) while the LLI.DMAC\_CTRLBx register of the last Linked List item must be set as described in Row 1 of [Table 37-1](#). [Figure 37-4 on page 580](#) shows a Linked List example with two list items.
  6. Make sure that the LLI.DMAC\_DSCRx register locations of all LLIs in memory (except the last) are non-zero and point to the next Linked List Item.
  7. Make sure that the LLI.DMAC\_DADDRx register location of all LLIs in memory point to the start destination buffer address proceeding that LLI fetch.
  8. Make sure that the LLI.DMAC\_CTLx.DONE field of the LLI.DMAC\_CTRLA register locations of all LLIs in memory is cleared.
  9. If source picture-in-picture is enabled (DMAC\_CTRLBx.SPIP is enabled), program the DMAC\_SPIPx register for channel x.
  10. If destination picture-in-picture is enabled (DMAC\_CTRLBx.DPIP is enabled), program the DMAC\_DPIPx register for channel x.
  11. Clear any pending interrupts on the channel from the previous DMAC transfer by reading to the DMAC\_EBCISR register.
  12. Program the DMAC\_CTLx, DMAC\_CFGx registers according to Row 6 as shown in [Table 37-1 on page 581](#).
  13. Program the DMAC\_DSCRx register with DMAC\_DSCRx(0), the pointer to the first Linked List item.
  14. Finally, enable the channel by writing a '1' to the DMAC\_CHER.ENABLE[n] bit where n is the channel number. The transfer is performed. Make sure that bit 0 of the DMAC\_EN register is enabled.
  15. The DMAC fetches the first LLI from the location pointed to by DMAC\_DSCRx(0).
- Note: The LLI.DMAC\_SADDRx, LLI.DMAC\_DADDRx, LLI.DMAC\_LL Px LLI.DMAC\_CTRLAx and LLI.DMAC\_CTRLBx registers are fetched. The LLI.DMAC\_SADDRx register although fetched is not used.
16. The DMAC\_CTRLAx register is written out to system memory. The DMAC\_CTRLAx register is written out to the same location on the same layer (DMAC\_DSCRx.DSCR\_IF) where it was originally fetched, that is the location of the DMAC\_CTRLAx register of the linked list item fetched prior to the start of the buffer transfer. Only DMAC\_CTRLAx register is written out, because only the DMAC\_CTRLAx.BTSIZE and DMAC\_CTRLAx.DONE fields have been updated by hardware within the DMAC. The LLI.DMAC\_CTRLAx.DONE bit is asserted to indicate buffer completion. Therefore, software can poll the LLI.DMAC\_CTRLAx.DONE field of the DMAC\_CTRLAx register in the LLI to ascertain when a buffer transfer has completed.
- Note: Do not poll the DMAC\_CTRLAx.DONE bit in the DMAC memory map. Instead poll the LLI.DMAC\_CTRLAx.DONE bit in the LLI for that buffer. If the polled LLI.DMAC\_CTRLAx.DONE bit is asserted, then this buffer transfer has completed. This LLI.DMAC\_CTRLA.DONE bit was cleared at the start of the transfer.
17. The DMAC reloads the DMAC\_SADDRx register from the initial value. Hardware sets the buffer complete interrupt. The DMAC samples the row number as shown in [Table 37-1 on page 581](#). If the DMAC is in Row 1, then the DMAC transfer has completed. Hardware sets the transfer complete interrupt and disables the channel. You can either respond to the Buffer Complete or Chained buffer Transfer Complete interrupts, or poll for the Channel Enable (DMAC\_CHSR.ENABLE) bit until it is cleared by hardware, to

detect when the transfer is complete. If the DMAC is not in Row 1 as shown in [Table 37-1 on page 581](#), the following step is performed.

18. The DMAC fetches the next LLI from memory location pointed to by the current DMAC\_DSCRx register, and automatically reprograms the DMAC\_DADDRx, DMAC\_CTRLAx, DMAC\_CTRLBx and DMAC\_DSCRx channel registers. Note that the DMAC\_SADDRx is not re-programmed as the reloaded value is used for the next DMAC buffer transfer. If the next buffer is the last buffer of the DMAC transfer then the DMAC\_CTRLBx and DMAC\_DSCRx registers just fetched from the LLI should match Row 1 of [Table 37-1 on page 581](#). The DMAC transfer might look like that shown in [Figure 37-10 on page 592](#).

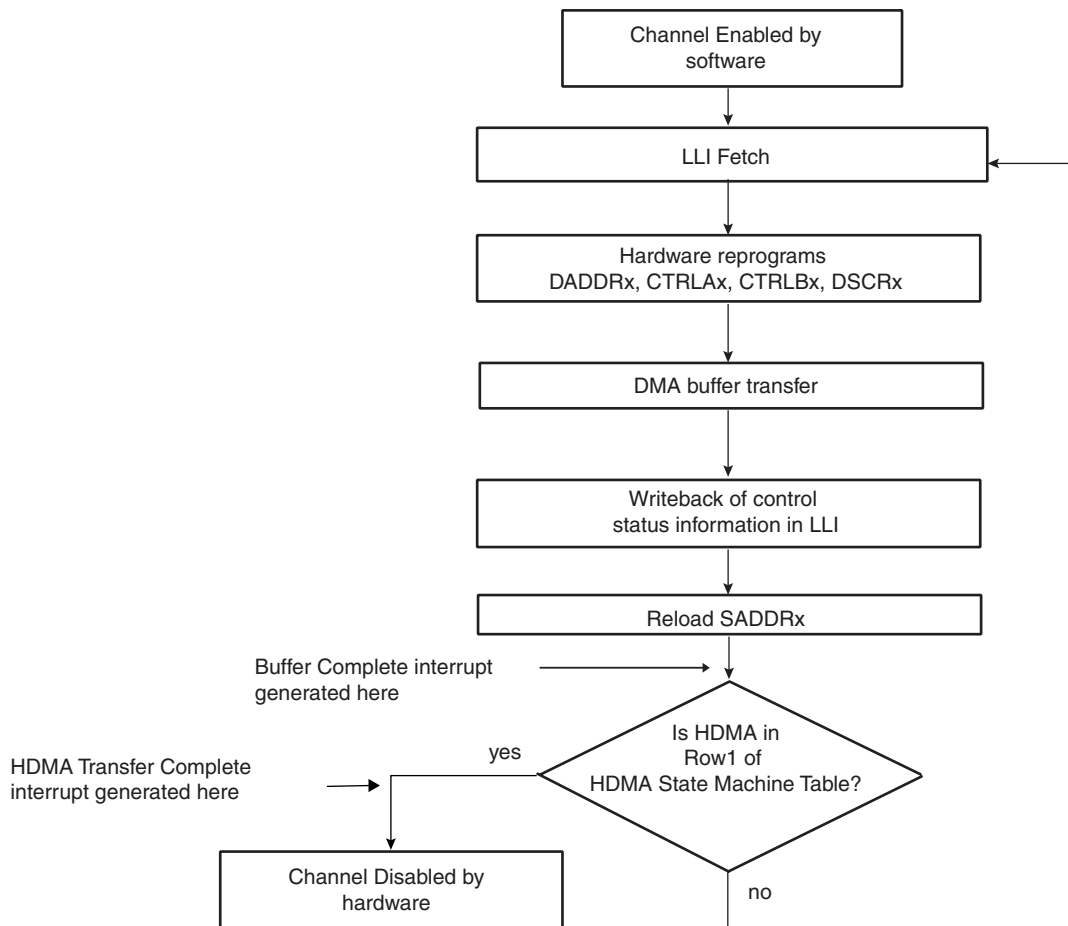
**Figure 37-10.** Multi-buffer DMAC Transfer with Source Address Auto-reloaded and Linked List Destination Address



The DMAC Transfer flow is shown in [Figure 37-11 on page 593](#).



**Figure 37-11.** DMAC Transfer Flow for Replay Mode at Source and Linked List Destination Address



### 37.3.4.6 Multi-buffer Transfer with Source Address Auto-reloaded and Contiguous Destination Address (Row 11)

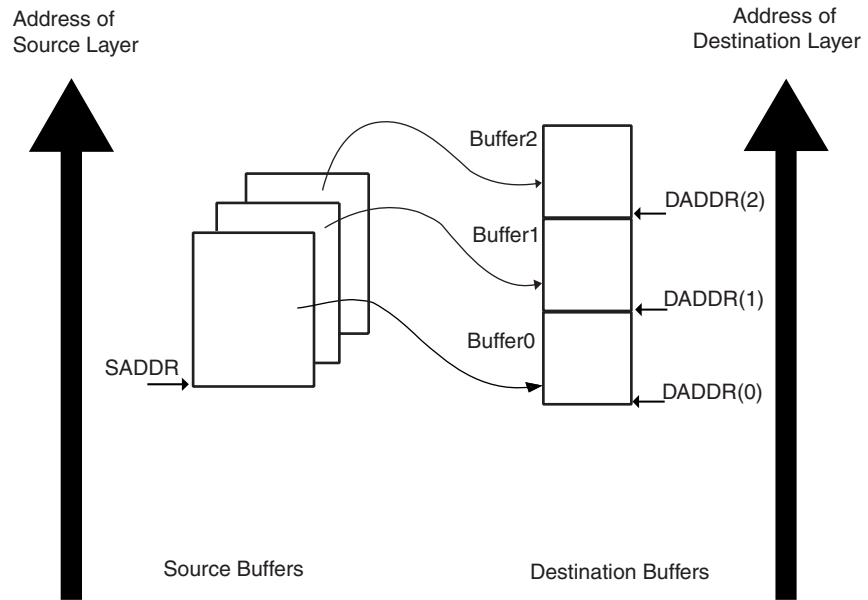
1. Read the Channel Enable register to choose a free (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMAC transfer by reading to the Interrupt Status Register.
3. Program the following channel registers:
  - a. Write the starting source address in the DMAC\_SADDRx register for channel x.
  - b. Write the starting destination address in the DMAC\_DADDRx register for channel x.
  - c. Program DMAC\_CTRLAx, DMAC\_CTRLBx and DMAC\_CFGx according to Row 11 as shown in [Table 37-1 on page 581](#). Program the DMAC\_DSCRx register with '0'. DMAC\_CTRLBx.AUTO field is set to '1' to enable automatic mode support.
  - d. Write the control information for the DMAC transfer in the DMAC\_CTRLBx and DMAC\_CTRLAx register for channel x. For example, in this register, you can program the following:
    - Set up the transfer characteristics, such as:
    - Transfer width for the source in the SRC\_WIDTH field.
    - Transfer width for the destination in the DST\_WIDTH field.

- Source AHB master interface layer in the SIF field where source resides.
  - Destination AHB master interface master layer in the DIF field where destination resides.
  - Incrementing/decrementing or fixed address for source in SRC\_INCR field.
  - Incrementing/decrementing or fixed address for destination in DST\_INCR field.
- e. If source picture-in-picture is enabled (DMAC\_CTRLBx.SPIP is enabled), program the DMAC\_SPIPx register for channel x.
  - f. If destination picture-in-picture is enabled (DMAC\_CTRLBx.DPIP), program the DMAC\_DPIPx register for channel x.
  - g. Write the channel configuration information into the DMAC\_CFGx register for channel x.
4. After the DMAC channel has been programmed, enable the channel by writing a '1' to the DMAC\_CHER.ENABLE[n] bit where n is the channel number. Make sure that bit 0 of the DMAC\_EN.ENABLE register is enabled.
  5. When the buffer transfer has completed, the DMAC reloads the DMAC\_SADDRx register. The DMAC\_DADDRx register remains unchanged. Hardware sets the buffer complete interrupt. The DMAC then samples the row number as shown in [Table 37-1 on page 581](#). If the DMAC is in Row 1, then the DMAC transfer has completed. Hardware sets the transfer complete interrupt and disables the channel. So you can either respond to the Buffer Complete or Transfer Complete interrupts, or poll for ENABLE field in the Channel Status Register (DMAC\_CHSR.ENABLE[n] bit) until it is cleared by hardware, to detect when the transfer is complete. If the DMAC is not in Row 1, the next step is performed.
  6. The DMAC transfer proceeds as follows:
    - a. If the buffer complete interrupt is un-masked (DMAC\_EBCIMR.BTC[x] = '1', where x is the channel number) hardware sets the buffer complete interrupt when the buffer transfer has completed. It then stalls until STALLED[n] bit of DMAC\_CHSR is cleared by writing in the KEEPON[n] field of DMAC\_CHER register where n is the channel number. If the next buffer is to be the last buffer in the DMAC transfer, then the buffer complete ISR (interrupt service routine) should clear the automatic mode bit, DMAC\_CTRLBx.AUTO. This puts the DMAC into Row 1 as shown in [Table 37-1 on page 581](#). If the next buffer is not the last buffer in the DMAC transfer then the automatic transfer mode bit should remain enabled to keep the DMAC in Row 11 as shown in [Table 37-1 on page 581](#).
    - b. If the buffer complete interrupt is masked (DMAC\_EBCIMR.BTC[x] = '1', where x is the channel number) then hardware does not stall until it detects a write to the buffer transfer completed interrupt enable register but starts the next buffer transfer immediately. In this case software must clear the automatic mode bit, DMAC\_CTRLBx.AUTO, to put the device into ROW 1 of [Table 37-1 on page 581](#) before the last buffer of the DMAC transfer has completed.

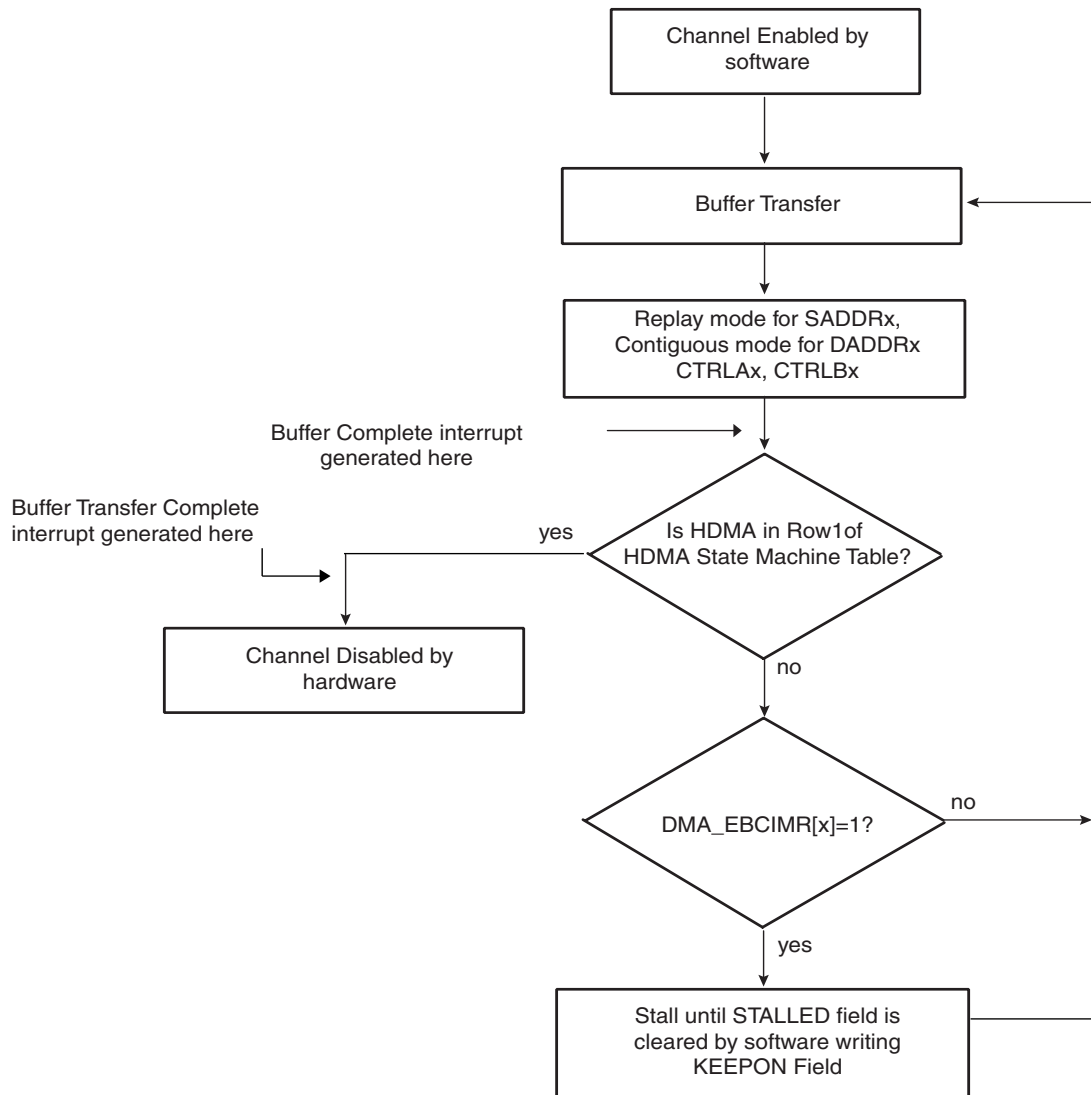
The transfer is similar to that shown in [Figure 37-12 on page 595](#).

The DMAC Transfer flow is shown in [Figure 37-13 on page 596](#).

**Figure 37-12.** Multi-buffer Transfer with Source Address Auto-reloaded and Contiguous Destination Address



**Figure 37-13.** DMAC Transfer Replay Mode is Enabled for the Source and Contiguous Destination Address



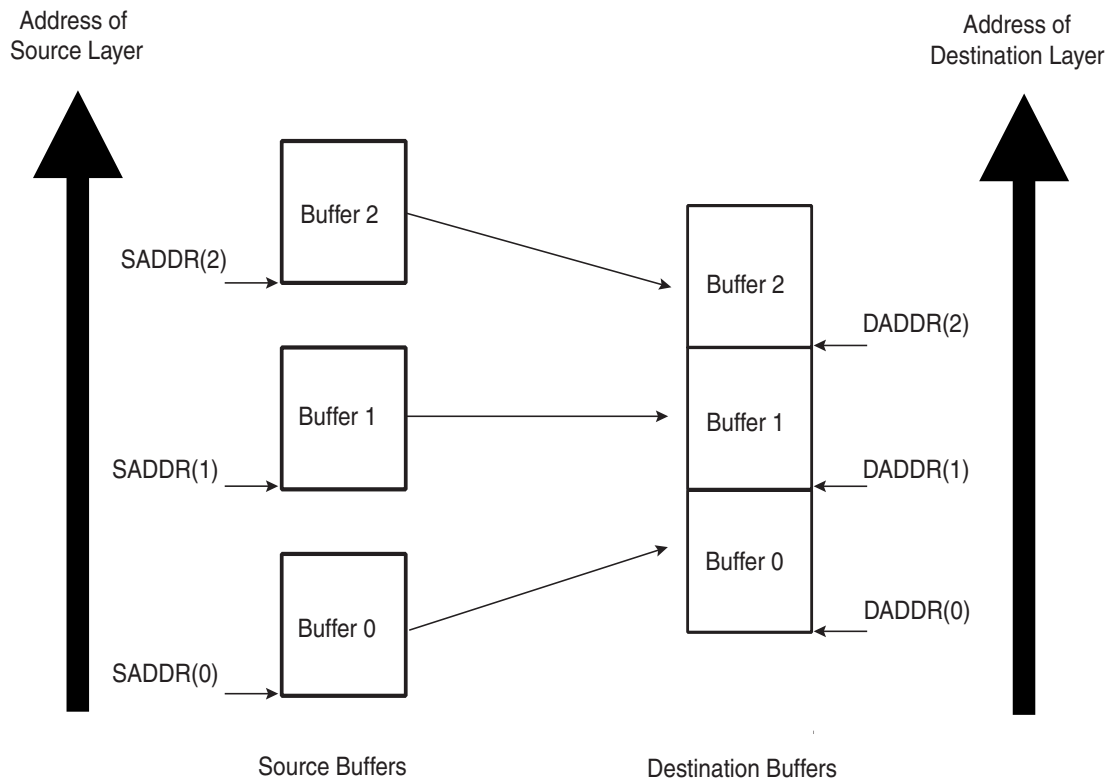
### 37.3.4.7 Multi-buffer DMAC Transfer with Linked List for Source and Contiguous Destination Address (Row 2)

1. Read the Channel Enable register to choose a free (disabled) channel.
2. Set up the linked list in memory. Write the control information in the LLI.DMAC\_CTRLAx and LLI.DMAC\_CTRLBx register location of the buffer descriptor for each LLI in memory for channel x. For example, in the register, you can program the following:
  - c. Set up the transfer characteristics, such as:
    - i. Transfer width for the source in the SRC\_WIDTH field.
    - ii. Transfer width for the destination in the DST\_WIDTH field.
    - iii. Source AHB master interface layer in the SIF field where source resides.
    - iv. Destination AHB master interface layer in the DIF field where destination resides.
    - v. Incrementing/decrementing or fixed address for source in SRC\_INCR field.
    - vi. Incrementing/decrementing or fixed address for destination DST\_INCR field.

3. Write the starting destination address in the DMAC\_DADDRx register for channel x.
- Note: The values in the LLI.DMAC\_DADDRx register location of each Linked List Item (LLI) in memory, although fetched during an LLI fetch, are not used.
4. Write the channel configuration information into the DMAC\_CFGx register for channel x.
  5. Make sure that all LLI.DMAC\_CTRLBx register locations of the LLI (except the last) are set as shown in Row 2 of [Table 37-1 on page 581](#), while the LLI.DMAC\_CTRLBx register of the last Linked List item must be set as described in Row 1 of [Table 37-1](#). [Figure 37-4 on page 580](#) shows a Linked List example with two list items.
  6. Make sure that the LLI.DMAC\_DSCRx register locations of all LLIs in memory (except the last) are non-zero and point to the next Linked List Item.
  7. Make sure that the LLI.DMAC\_SADDRx register location of all LLIs in memory point to the start source buffer address proceeding that LLI fetch.
  8. Make sure that the LLI.DMAC\_CTRLAx.DONE field of the LLI.DMAC\_CTRLAx register locations of all LLIs in memory is cleared.
  9. If source picture-in-picture is enabled (DMAC\_CTRLBx.SPIP is enabled), program the DMAC\_SPIPx register for channel x.
  10. If destination picture-in-picture is enabled (DMAC\_CTRLBx.DPIP is enabled), program the DMAC\_DPIPx register for channel x.
  11. Clear any pending interrupts on the channel from the previous DMAC transfer by reading the interrupt status register.
  12. Program the DMAC\_CTRLAx, DMAC\_CTRLBx and DMAC\_CFGx registers according to Row 2 as shown in [Table 37-1 on page 581](#)
  13. Program the DMAC\_DSCRx register with DMAC\_DSCRx(0), the pointer to the first Linked List item.
  14. Finally, enable the channel by writing a '1' to the DMAC\_CHER.ENABLE[n] bit. The transfer is performed. Make sure that bit 0 of the DMAC\_EN register is enabled.
  15. The DMAC fetches the first LLI from the location pointed to by DMAC\_DSCRx(0).
- Note: The LLI.DMAC\_SADDRx, LLI.DMAC\_DADDRx, LLI.DMAC\_DSCRx and LLI.DMAC\_CTRLA/Bx registers are fetched. The LLI.DMAC\_DADDRx register location of the LLI although fetched is not used. The DMAC\_DADDRx register in the DMAC remains unchanged.
16. Once the buffer of data is transferred, the DMAC\_CTRLAx register is written out to system memory at the same location and on the same layer (DMAC\_DSCRx.DSCR\_IF) where it was originally fetched, that is, the location of the DMAC\_CTRLAx register of the linked list item fetched prior to the start of the buffer transfer. Only DMAC\_CTRLAx register is written out because only the DMAC\_CTRLAx.BTSIZE and DMAC\_CTRLAx.DONE fields have been updated by DMAC hardware. Additionally, the DMAC\_CTRLAx.DONE bit is asserted when the buffer transfer has completed.
- Note: Do not poll the DMAC\_CTRLAx.DONE bit in the DMAC memory map. Instead, poll the LLI.DMAC\_CTRLAx.DONE bit in the LLI for that buffer. If the poll LLI.DMAC\_CTRLAx.DONE bit is asserted, then this buffer transfer has completed. This LLI.DMAC\_CTRLAx.DONE bit was cleared at the start of the transfer.
17. The DMAC does not wait for the buffer interrupt to be cleared, but continues and fetches the next LLI from the memory location pointed to by current DMAC\_DSCRx register and automatically reprograms the DMAC\_SADDRx, DMAC\_CTRLAx, DMAC\_CTRLBx and DMAC\_DSCRx channel registers. The DMAC\_DADDRx register is left unchanged. The DMAC transfer continues until the DMAC samples the DMAC\_CTRLAx, DMAC\_CTRLBx and DMAC\_DSCRx registers at the end of a buffer transfer match that described in Row 1 of [Table 37-1 on page 581](#). The DMAC then knows that the previous buffer transferred was the last buffer in the DMAC transfer.

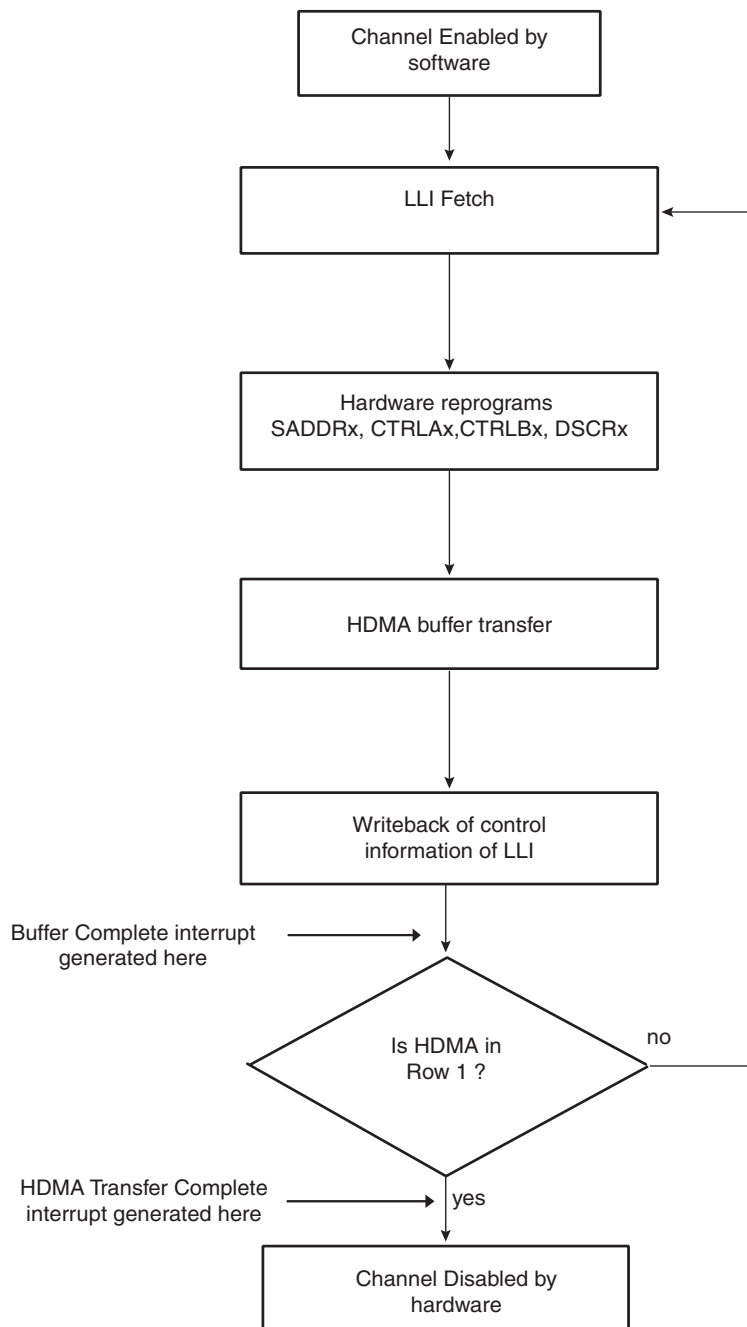
The DMAC transfer might look like that shown in [Figure 37-14 on page 598](#). Note that the destination address is decrementing.

**Figure 37-14.** DMAC Transfer with Linked List Source Address and Contiguous Destination Address



The DMAC transfer flow is shown in [Figure 37-15 on page 599](#).

**Figure 37-15.** DMAC Transfer Flow for Linked List Source Address and Contiguous Destination Address



### 37.3.5 Disabling a Channel Prior to Transfer Completion

Under normal operation, software enables a channel by writing a '1' to the Channel Handler Enable Register, DMAC\_CHER.ENABLE[n], and hardware disables a channel on transfer completion by clearing the DMAC\_CHSR.ENABLE[n] register bit.

The recommended way for software to disable a channel without losing data is to use the SUSPEND[n] bit in conjunction with the EMPTY[n] bit in the Channel Handler Status Register.

1. If software wishes to disable a channel *n* prior to the DMAC transfer completion, then it can set the DMAC\_CHER.SUSPEND[*n*] bit to tell the DMAC to halt all transfers from the source peripheral. Therefore, the channel FIFO receives no new data.
2. Software can now poll the DMAC\_CHSR.EMPTY[*n*] bit until it indicates that the channel *n* FIFO is empty, where *n* is the channel number.
3. The DMAC\_CHER.ENABLE[*n*] bit can then be cleared by software once the channel *n* FIFO is empty, where *n* is the channel number.

When DMAC\_CTRLAx.SRC\_WIDTH is less than DMAC\_CTRLAx.DST\_WIDTH and the DMAC\_CHSRx.SUSPEND[*n*] bit is high, the DMAC\_CHSRx.EMPTY[*n*] is asserted once the contents of the FIFO do not permit a single word of DMAC\_CTRLAx.DST\_WIDTH to be formed. However, there may still be data in the channel FIFO but not enough to form a single transfer of DMAC\_CTLx.DST\_WIDTH width. In this configuration, once the channel is disabled, the remaining data in the channel FIFO are not transferred to the destination peripheral. It is permitted to remove the channel from the suspension state by writing a '1' to the DMAC\_CHER.RESUME[*n*] field register. The DMAC transfer completes in the normal manner. *n* defines the channel number.

### 37.3.5.1 Abnormal Transfer Termination

A DMAC transfer may be terminated abruptly by software by clearing the channel enable bit, DMAC\_CHDR.ENABLE[*n*] where *n* is the channel number. This does not mean that the channel is disabled immediately after the DMAC\_CHSR.ENABLE[*n*] bit is cleared over the APB interface. Consider this as a request to disable the channel. The DMAC\_CHSR.ENABLE[*n*] must be polled and then it must be confirmed that the channel is disabled by reading back 0.

Software may terminate all channels abruptly by clearing the global enable bit in the DMAC Configuration Register (DMAC\_EN.ENABLE bit). Again, this does not mean that all channels are disabled immediately after the DMAC\_EN.ENABLE is cleared over the APB slave interface. Consider this as a request to disable all channels. The DMAC\_CHSR.ENABLE must be polled and then it must be confirmed that all channels are disabled by reading back '0'.

**Note:** If the channel enable bit is cleared while there is data in the channel FIFO, this data is not sent to the destination peripheral and is not present when the channel is re-enabled. For read sensitive source peripherals, such as a source FIFO, this data is therefore lost. When the source is not a read sensitive device (i.e., memory), disabling a channel without waiting for the channel FIFO to empty may be acceptable as the data is available from the source peripheral upon request and is not lost.

## 37.4 DMAC Software Requirements

- There must not be any write operation to Channel registers in an active channel after the channel enable is made HIGH. If any channel parameters must be reprogrammed, this can only be done after disabling the DMAC channel.
- You must program the DMAC\_SADDRx and DMAC\_DADDRx channel registers with a byte, half-word and word aligned address depending on the source width and destination width.
- After the software disables a channel by writing into the channel disable register, it must re-enable the channel only after it has polled a 0 in the corresponding channel enable status register. This is because the current AHB Burst must terminate properly.
- If you program the BTSIZE field in the DMAC\_CTRLA, as zero, and the DMAC is defined as the flow controller, then the channel is automatically disabled.
- When AUTO Field is set to TRUE, then the BTSIZE Field is automatically reloaded from its previous value. BTSIZE must be initialized to a non zero value if the first transfer is initiated



with AUTO field set to TRUE even if LLI mode is enabled because the LLI fetch operation will not update this field.

## 37.5 DMA Controller (DMAC) User Interface

**Table 37-2.** Register Mapping

Offset	Register	Name	Access	Reset State
0x000	DMAC Global Configuration Register	DMAC_GCFG	Read/Write	0x10
0x004	DMAC Enable Register	DMAC_EN	Read/Write	0x0
0x008	Reserved	–	–	–
0x00C	Reserved	–	–	–
0x010	Reserved	–	–	–
0x014	Reserved	–	–	–
0x018	DMAC Error, Chained Buffer transfer completed and Buffer transfer completed Interrupt Enable register.	DMAC_EBCIER	Write-only	–
0x01C	DMAC Error, Chained Buffer transfer completed and Buffer transfer completed Interrupt Disable register.	DMAC_EBCIDR	Write-only	–
0x020	DMAC Error, Chained Buffer transfer completed and Buffer transfer completed Mask Register.	DMAC_EBCIMR	Read-only	0x0
0x024	DMAC Error, Chained Buffer transfer completed and Buffer transfer completed Status Register.	DMAC_EBCISR	Read-only	0x0
0x028	DMAC Channel Handler Enable Register	DMAC_CHER	Write-only	–
0x02C	DMAC Channel Handler Disable Register	DMAC_CHDR	Write-only	–
0x030	DMAC Channel Handler Status Register	DMAC_CHSR	Read-only	0x00FF0000
0x034	Reserved	–	–	–
0x038	Reserved	–	–	–
0x03C	DMAC Channel 0 Source Address Register	DMAC_SADDR0	Read/Write	0x0
0x040	DMAC Channel 0 Destination Address Register	DMAC_DADDR0	Read/Write	0x0
0x044	DMAC Channel 0 Descriptor Address Register	DMAC_DSCR0	Read/Write	0x0
0x048	DMAC Channel 0 Control A Register	DMAC_CTRLA0	Read/Write	0x0
0x04C	DMAC Channel 0 Control B Register	DMAC_CTRLB0	Read/Write	0x0
0x050	DMAC Channel 0 Configuration Register	DMAC_CFG0	Read/Write	0x01000000
0x054	DMAC Channel 0 Source Picture in Picture Configuration Register	DMAC_SPIP0	Read/Write	0x0
0x058	DMAC Channel 0 Destination Picture in Picture Configuration Register	DMAC_DPIP0	Read/Write	0x0
0x05C	Reserved	–	–	–
0x060	Reserved	–	–	–
0x064	DMAC Channel 1 Source Address Register	DMAC_SADDR1	Read/Write	0x0
0x068	DMAC Channel 1 Destination Address Register	DMAC_DADDR1	Read/Write	0x0
0x06C	DMAC Channel 1 Descriptor Address Register	DMAC_DSCR1	Read/Write	0x0
0x070	DMAC Channel 1 Control A Register	DMAC_CTRLA1	Read/Write	0x0
0x074	DMAC Channel 1 Control B Register	DMAC_CTRLB1	Read/Write	0x0

**Table 37-2.** Register Mapping

Offset	Register	Name	Access	Reset State
0x078	DMAC Channel 1 Configuration Register	DMAC_CFG1	Read/Write	0x01000000
0x07C	DMAC Channel 1 Source Picture in Picture Configuration Register	DMAC_SPIP1	Read/Write	0x0
0x080	DMAC Channel 1 Destination Picture in Picture Configuration Register	DMAC_DPIP1	Read/Write	0x0

## 37.5.1 DMAC Global Configuration Register

Name: DMAC\_GCFG

Access: Read/Write

Reset Value: 0x00000010

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	ARB_CFG	–	–	–	IF0_BIGEND

- **IF0\_BIGEND**

0: AHB-Lite Interface 0 is little endian.

1: AHB-Lite Interface 0 is big endian.

- **ARB\_CFG**

0: Fixed priority arbiter.

1: Modified round robin arbiter.

## 37.5.2 DMAC Enable Register

Name: DMAC\_EN

Access: 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
-	-	-	-	-	-	-	ENABLE

- **ENABLE**

0: DMA Controller is disabled.

1: DMA Controller is enabled.

## 37.5.3 DMAC Error, Buffer Transfer and Chained Buffer Transfer Interrupt Enable Register

Name: DMAC\_EBCIER

Access: Write-only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	ERR1	ERR0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	CBTC1	CBTC0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	BTC1	BTC0

- **BTC[1:0]**

Buffer Transfer Completed Interrupt Enable Register. Set the relevant bit in the BTC field to enable the interrupt for channel i.

- **CBTC[1:0]**

Chained Buffer Transfer Completed Interrupt Enable Register. Set the relevant bit in the CBTC field to enable the interrupt for channel i.

- **ERR[1:0]**

Access Error Interrupt Enable Register. Set the relevant bit in the ERR field to enable the interrupt for channel i.

## 37.5.4 DMAC Error, Buffer Transfer and Chained Buffer Transfer Interrupt Disable Register

Name: DMAC\_EBCIDR

Access: Write-only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	ERR1	ERR0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	CBTC1	CBTC0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	BTC1	BTC0

- **BTC[1:0]**

Buffer transfer completed Disable Interrupt Register. When set, a bit of the BTC field disables the interrupt from the relevant DMAC channel.

- **CBTC[1:0]**

Chained Buffer transfer completed Disable Register. When set, a bit of the CBTC field disables the interrupt from the relevant DMAC channel.

- **ERR[1:0]**

Access Error Interrupt Disable Register. When set, a bit of the ERR field disables the interrupt from the relevant DMAC channel.

## 37.5.5 DMAC Error, Buffer Transfer and Chained Buffer Transfer Interrupt Mask Register

Name: DMAC\_EBCIMR

Access: Read-only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	ERR1	ERR0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	CBTC1	CBTC0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	BTC1	BTC0

- **BTC[1:0]**

0: Buffer Transfer completed interrupt is disabled for channel i.

1: Buffer Transfer completed interrupt is enabled for channel i.

- **CBTC[1:0]**

0: Chained Buffer Transfer interrupt is disabled for channel i.

1: Chained Buffer Transfer interrupt is enabled for channel i.

- **ERR[1:0]**

0: Transfer Error Interrupt is disabled for channel i.

1: Transfer Error Interrupt is enabled for channel i.



## 37.5.6 DMAC Error, Buffer Transfer and Chained Buffer Transfer Status Register

Name: DMAC\_EBCISR

Access: Read-only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	ERR1	ERR0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	CBTC1	CBTC0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	BTC1	BTC0

- **BTC[1:0]**

When BTC[*i*] is set, Channel *i* buffer transfer has terminated.

- **CBTC[1:0]**

When CBTC[*i*] is set, Channel *i* Chained buffer has terminated. LLI Fetch operation is disabled.

- **ERR[1:0]**

When ERR[*i*] is set, Channel *i* has detected an AHB Read or Write Error Access.

## 37.5.7 DMAC Channel Handler Enable Register

Name: DMAC\_CHER

Access: Write-only

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	KEEP1	KEEP0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	SUSP1	SUSP0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	ENA1	ENA0

- **ENA[1:0]**

When set, a bit of the ENA field enables the relevant channel.

- **SUSP[1:0]**

When set, a bit of the SUSP field freezes the relevant channel and its current context.

- **KEEP[1:0]**

When set, a bit of the KEEP field resumes the current channel from an automatic stall state.

## 37.5.8 DMAC Channel Handler Disable Register

Name: DMAC\_CHDR

Access: Write-only

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
–	–	–	–	–	–	RES1	RES0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	DIS1	DIS0

- **DIS[1:0]**

Write one to this field to disable the relevant DMAC Channel. The content of the FIFO is lost and the current AHB access is terminated. Software must poll DIS[1:0] field in the DMAC\_CHSR register to be sure that the channel is disabled.

- **RES[1:0]**

Write one to this field to resume the channel transfer restoring its context.

## 37.5.9 DMAC Channel Handler Status Register

Name: DMAC\_CHSR

Access: Read-only

Reset Value: 0x00FF0000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	STAL1	STAL0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	EMPT1	EMPT0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	SUSP1	SUSP0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	ENA1	ENA0

- **ENA[1:0]**

A one in any position of this field indicates that the relevant channel is enabled.

- **SUSP[1:0]**

A one in any position of this field indicates that the channel transfer is suspended.

- **EMPT[1:0]**

A one in any position of this field indicates that the relevant channel is empty.

- **STAL[1:0]**

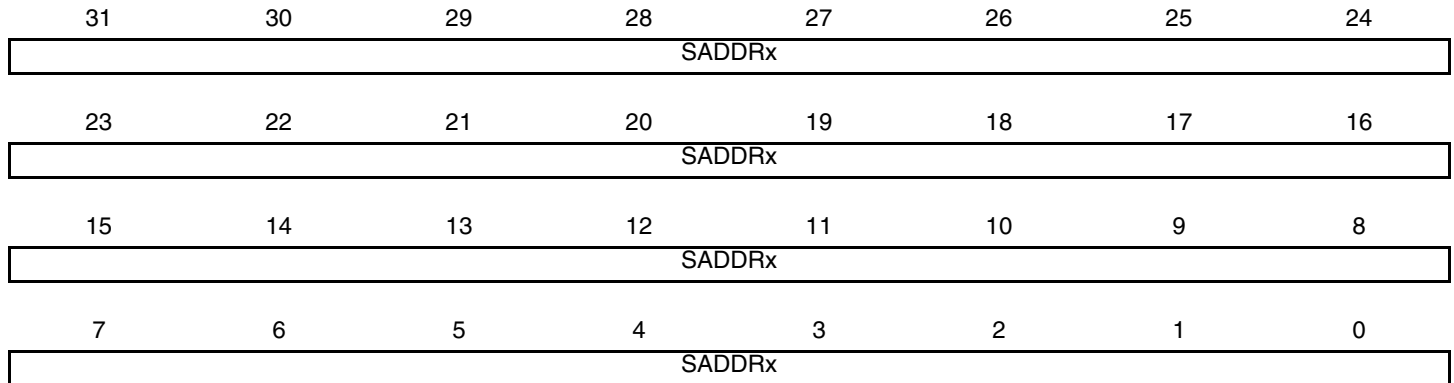
A one in any position of this field indicates that the relevant channel is stalling.

## 37.5.10 DMAC Channel x [x = 0..1] Source Address Register

Name: DMAC\_SADDRx [x = 0..1]

Access: Read/Write

Reset Value: 0x00000000



- **SADDRx**

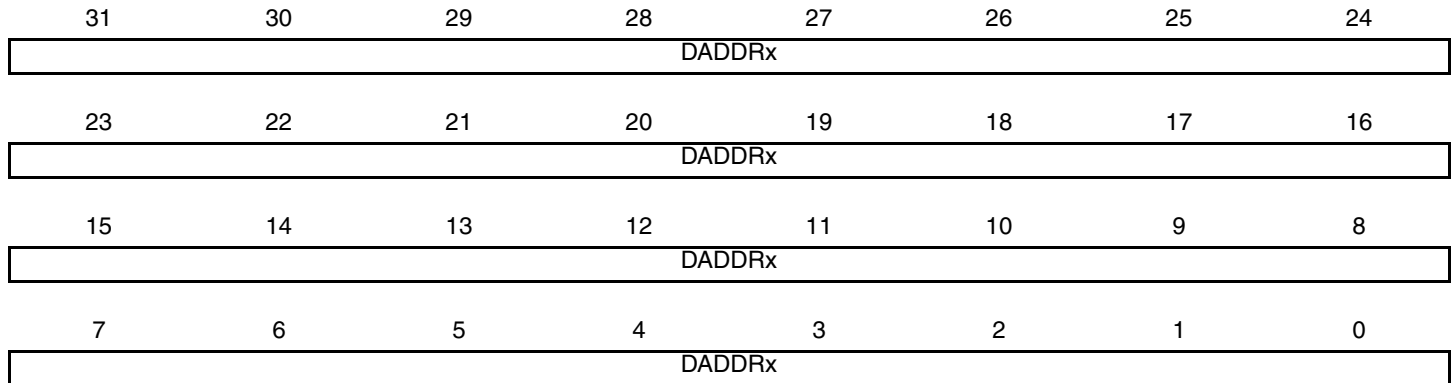
Channel x source address. This register must be aligned with the source transfer width.

## 37.5.11 DMAC Channel x [x = 0..1] Destination Address Register

Name: DMAC\_DADDRx [x = 0..1]

Access: Read/Write

Reset Value: 0x00000000



- **DADDRx**

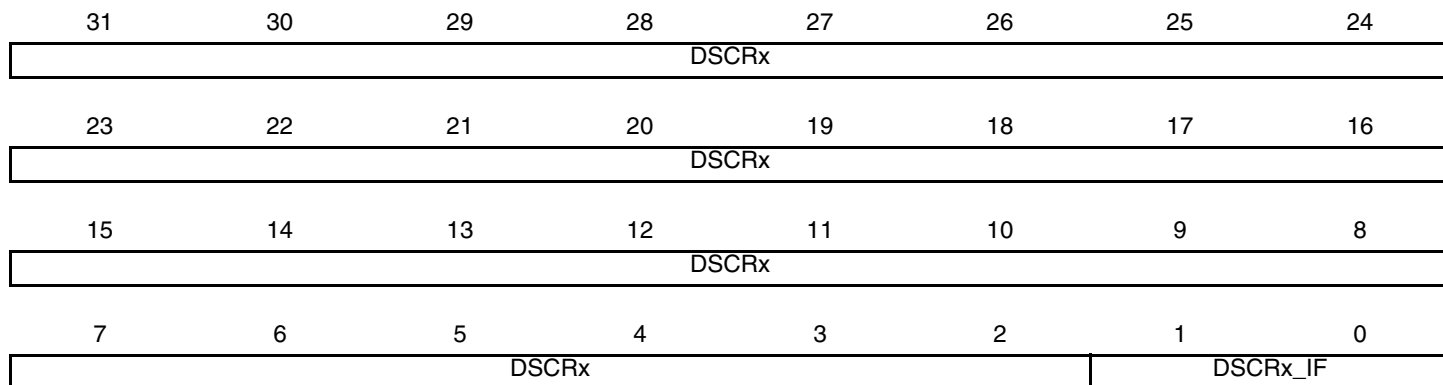
Channel x destination address. This register must be aligned with the destination transfer width.

## 37.5.12 DMAC Channel x [x = 0..1] Descriptor Address Register

Name: DMAC\_DSCRx [x = 0..1]

Access: Read/Write

Reset Value: 0x00000000



- **DSCRx\_IF**

00: The Buffer Transfer descriptor is fetched via AHB-Lite Interface 0.

01: Reserved.

10: Reserved.

11: Reserved.

- **DSCRx**

Buffer Transfer descriptor address. This address is word aligned.

## 37.5.13 DMAC Channel x [x = 0..1] Control A Register

Name: DMAC\_CTRLAx [x = 0..1]

Access: Read/Write

Reset Value: 0x00000000

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

- **BTSIZE**

Buffer Transfer Size. The transfer size relates to the number of transfers to be performed, that is, for writes it refers to the number of source width transfers to perform when DMAC is flow controller. For Reads, BTSIZE refers to the number of transfers completed on the Source Interface. When this field is set to 0, the DMAC module is automatically disabled when the relevant channel is enabled.

- **SRC\_WIDTH**

SRC_WIDTH	Single Transfer Size
00	BYTE
01	HALF-WORD
1X	WORD

- **DST\_WIDTH**

DST_WIDTH	Single Transfer Size
00	BYTE
01	HALF-WORD
1X	WORD

- **DONE**

0: The transfer is performed.

1: If SOD field of DMAC\_CFG register is set to true, then the DMAC is automatically disabled when an LLI updates the content of this register.

The DONE field is written back to memory at the end of the transfer.



## 37.5.14 DMAC Channel x [x = 0..1] Control B Register

Name: DMAC\_CTRLBx [x = 0..1]

Access: Read/Write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
AUTO	–	DST_INCR		–	–	SRC_INCR	
23	22	21	20	19	18	17	16
–	–	–	DST_DSCR	–	–	–	SRC_DSCR
15	14	13	12	11	10	9	8
–	–		DST_PIP	–	–	–	SRC_PIP
7	6	5	4	3	2	1	0
–	–	DIF		–	–	SIF	

- **SIF**

Source Interface Selection Field.

00: The source transfer is done via AHB-Lite Interface 0.

01: Reserved.

10: Reserved.

11: Reserved.

- **DIF**

Destination Interface Selection Field.

00: The destination transfer is done via AHB-Lite Interface 0.

01: Reserved.

10: Reserved.

11: Reserved.

- **SRC\_PIP**

0: Picture-in-Picture mode is disabled. The source data area is contiguous.

1: Picture-in-Picture mode is enabled. When the source PIP counter reaches the programmable boundary, the address is automatically increment of a user defined amount.

- **DST\_PIP**

0: Picture-in-Picture mode is disabled. The Destination data area is contiguous.

1: Picture-in-Picture mode is enabled. When the Destination PIP counter reaches the programmable boundary the address is automatically incremented by a user-defined amount.

- **SRC\_DSCR**

0: Source address is updated when the descriptor is fetched from the memory.

1: Buffer Descriptor Fetch operation is disabled for the source.

- **DST\_DSCR**

0: Destination address is updated when the descriptor is fetched from the memory.

1: Buffer Descriptor Fetch operation is disabled for the destination.

- **SRC\_INCR**

SRC_INCR	Type of addressing mode
00	INCREMENTING
01	DECREMENTING
10	FIXED

- **DST\_INCR**

DST_INCR	Type of addressing scheme
00	INCREMENTING
01	DECREMENTING
10	FIXED

- **AUTO**

Automatic multiple buffer transfer is enabled. When set, this bit enables replay mode or contiguous mode when several buffers are transferred.

## 37.5.15 DMAC Channel x [x = 0..1] Configuration Register

Name: DMAC\_CFGx [x = 0..1]

Access: Read/Write

Reset Value: 0x0100000000

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

- **SRC\_REP**

0: When automatic mode is activated, source address is contiguous between two buffers.

1: When automatic mode is activated, the source address and the control register are reloaded from previous transfer.

- **DST\_REP**

0: When automatic mode is activated, destination address is contiguous between two buffers.

1: When automatic mode is activated, the destination and the control register are reloaded from the previous transfer.

- **SOD**

0: STOP ON DONE disabled, the descriptor fetch operation ignores DONE Field of CTRLA register.

1: STOP ON DONE activated, the DMAC module is automatically disabled if DONE FIELD is set to 1.

## 37.5.16 DMAC Channel x [x = 0..1] Source Picture in Picture Configuration Register

Name: DMAC\_SPIPx [x = 0..1]

Access: Read/Write

Reset Value: 0x00000000

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

- **SPIP\_HOLE**

This field indicates the value to add to the address when the programmable boundary has been reached.

- **SPIP\_BOUNDARY**

This field indicates the number of source transfers to perform before the automatic address increment operation.

## 37.5.17 DMAC Channel x [x = 0..1] Destination Picture in Picture Configuration Register

Name: DMAC\_DPIP<sub>x</sub> [x = 0..1]

Access: Read/Write

Reset Value: 0x00000000

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

- **DPIP\_HOLE**

This field indicates the value to add to the address when the programmable boundary has been reached.

- **DPIP\_BOUNDARY**

This field indicates the number of source transfers to perform before the automatic address increment operation.



## 38. MultiMedia Card Interface (MCI)

### 38.1 Description

The MultiMedia Card Interface (MCI) supports the MultiMedia Card (MMC) Specification V3.11, the SDIO Specification V1.1 and the SD Memory Card Specification V1.0.

The MCI includes a command register, response registers, data registers, timeout counters and error detection logic that automatically handle the transmission of commands and, when required, the reception of the associated responses and data with a limited processor overhead.

The MCI supports stream, block and multi-block data read and write, and is compatible with the Peripheral DMA Controller (PDC) channels, minimizing processor intervention for large buffer transfers.

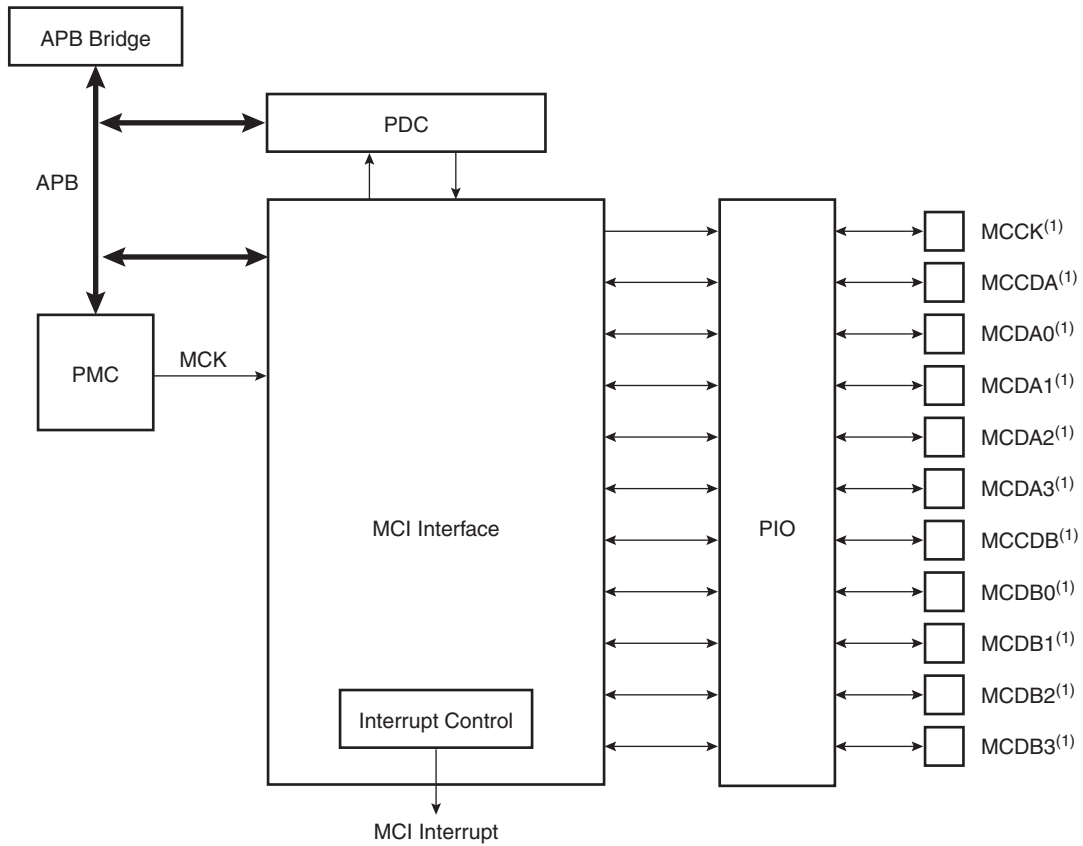
The MCI operates at a rate of up to Master Clock divided by 2 and supports the interfacing of 2 slot(s). Each slot may be used to interface with a MultiMediaCard bus (up to 30 Cards) or with a SD Memory Card. Only one slot can be selected at a time (slots are multiplexed). A bit field in the SD Card Register performs this selection.

The SD Memory Card communication is based on a 9-pin interface (clock, command, four data and three power lines) and the MultiMedia Card on a 7-pin interface (clock, command, one data, three power lines and one reserved for future use).

The SD Memory Card interface also supports MultiMedia Card operations. The main differences between SD and MultiMedia Cards are the initialization process and the bus topology.

## 38.2 Block Diagram

Figure 38-1. Block Diagram

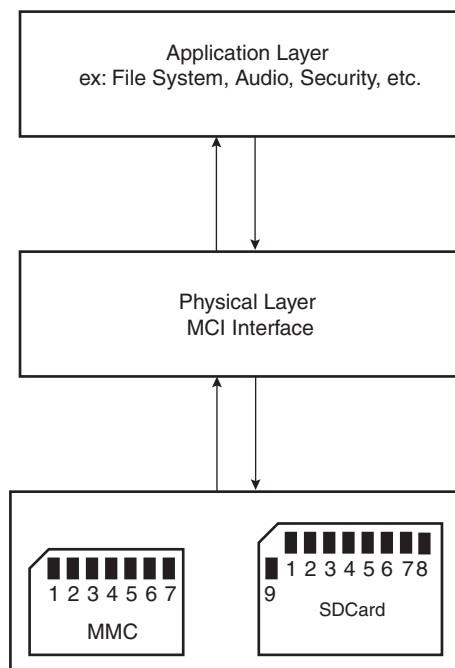


Note: 1. When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx\_CK, MCCDA to MCIx\_CDA, MCCDB to MCIx\_CDB, MCDAy to MCIx\_DAy, MCDBy to MCIx\_DBy.



## 38.3 Application Block Diagram

Figure 38-2. Application Block Diagram



## 38.4 Pin Name List

Table 38-1. I/O Lines Description

Pin Name <sup>(2)</sup>	Pin Description	Type <sup>(1)</sup>	Comments
MCCDA/MCCDB	Command/response	I/O/PP/OD	CMD of an MMC or SDCard/SDIO
MCCK	Clock	I/O	CLK of an MMC or SD Card/SDIO
MCDA0 - MCDA3	Data 0..3 of Slot A	I/O/PP	DAT0 of an MMC DAT[0..3] of an SD Card/SDIO
MCDB0 - MCDB3	Data 0..3 of Slot B	I/O/PP	DAT0 of an MMC DAT[0..3] of an SD Card/SDIO

Notes: 1. I: Input, O: Output, PP: Push/Pull, OD: Open Drain.

2. When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx\_CK, MCCDA to MCIx\_CDA, MCCDB to MCIx\_CDB, MCDAy to MCIx\_DAy, MCDBy to MCIx\_DBy.

## 38.5 Product Dependencies

### 38.5.1 I/O Lines

The pins used for interfacing the MultiMedia Cards or SD Cards may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the peripheral functions to MCI pins.

### 38.5.2 Power Management

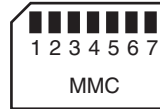
The MCI may be clocked through the Power Management Controller (PMC), so the programmer must first configure the PMC to enable the MCI clock.

### 38.5.3 Interrupt

The MCI interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling the MCI interrupt requires programming the AIC before configuring the MCI.

## 38.6 Bus Topology

**Figure 38-3.** Multimedia Memory Card Bus Topology



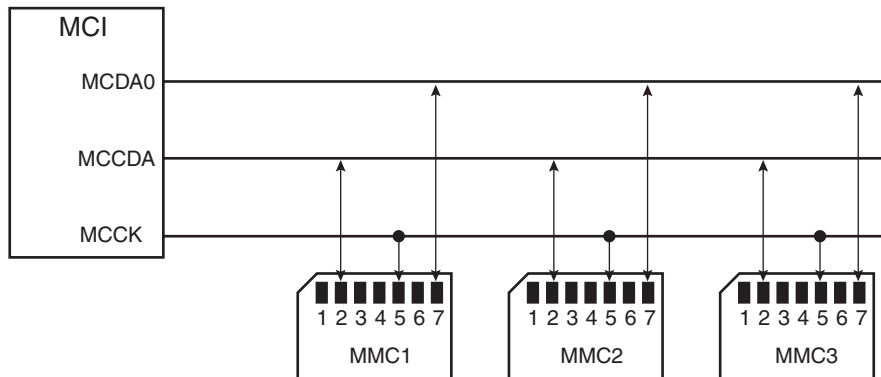
The MultiMedia Card communication is based on a 7-pin serial bus interface. It has three communication lines and four supply lines.

**Table 38-2.** Bus Topology

Pin Number	Name	Type <sup>(1)</sup>	Description	MCI Pin Name <sup>(2)</sup> (Slot z)
1	RSV	NC	Not connected	-
2	CMD	I/O/PP/OD	Command/response	MCCDz
3	VSS1	S	Supply voltage ground	VSS
4	VDD	S	Supply voltage	VDD
5	CLK	I/O	Clock	MCKK
6	VSS2	S	Supply voltage ground	VSS
7	DAT[0]	I/O/PP	Data 0	MCDz0

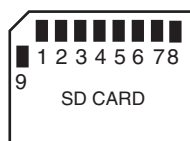
- Notes:
1. I: Input, O: Output, PP: Push/Pull, OD: Open Drain.
  2. When several MCI (x MCI) are embedded in a product, MCKK refers to MCIx\_CK, MCCDA to MCIx\_CDA, MCCDB to MCIx\_CDB, MCDAY to MCIx\_DAY, MCDBy to MCIx\_DBy.

**Figure 38-4.** MMC Bus Connections (One Slot)



Note: When several MCI (x MCI) are embedded in a product, MCKK refers to MCIx\_CK, MCCDA to MCIx\_CDA, MCDAY to MCIx\_DAY.

**Figure 38-5.** SD Memory Card Bus Topology



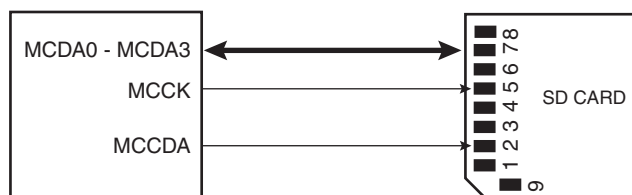
The SD Memory Card bus includes the signals listed in [Table 38-3](#).

**Table 38-3.** SD Memory Card Bus Signals

Pin Number	Name	Type <sup>(1)</sup>	Description	MCI Pin Name <sup>(2)</sup> (Slot z)
1	CD/DAT[3]	I/O/PP	Card detect/ Data line Bit 3	MCDz3
2	CMD	PP	Command/response	MCCDz
3	VSS1	S	Supply voltage ground	VSS
4	VDD	S	Supply voltage	VDD
5	CLK	I/O	Clock	MCCK
6	VSS2	S	Supply voltage ground	VSS
7	DAT[0]	I/O/PP	Data line Bit 0	MCDz0
8	DAT[1]	I/O/PP	Data line Bit 1 or Interrupt	MCDz1
9	DAT[2]	I/O/PP	Data line Bit 2	MCDz2

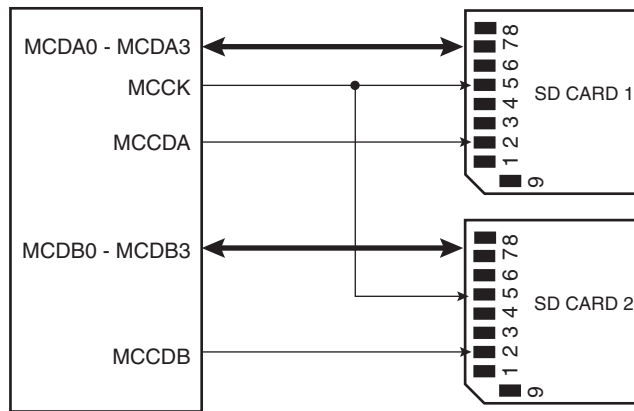
- Notes: 1. I: input, O: output, PP: Push Pull, OD: Open Drain.  
 2. When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx\_CK, MCCDA to MCIx\_CDA, MCCDB to MCIx\_CDB, MCDAy to MCIx\_DAy, MCDBy to MCIx\_DBy.

**Figure 38-6.** SD Card Bus Connections with One Slot



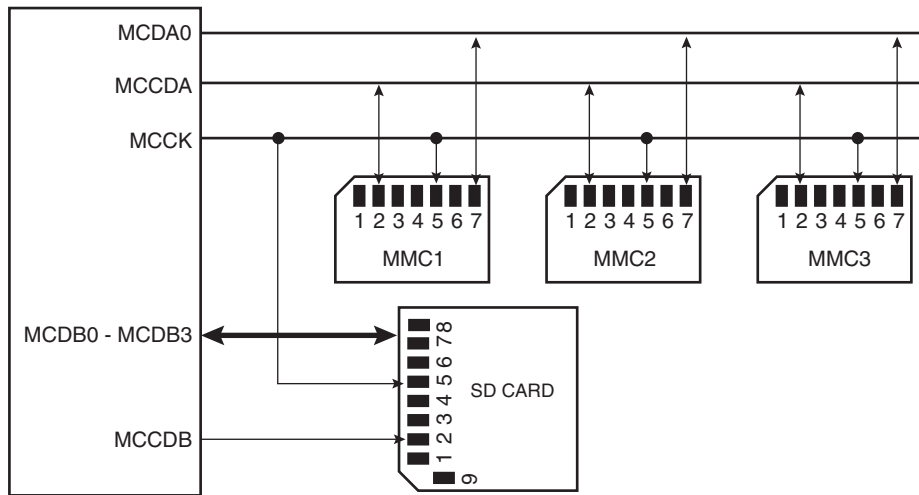
Note: When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx\_CK, MCCDA to MCIx\_CDA, MCDAy to MCIx\_DAy.

**Figure 38-7.** SD Card Bus Connections with Two Slots



Note: When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx\_CK, MCCDA to MCIx\_CDA, MCDAy to MCIx\_DAy, MCCDB to MCIx\_CDB, MCDBy to MCIx\_DBy.

**Figure 38-8.** Mixing MultiMedia and SD Memory Cards with Two Slots



Note: When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx\_CK, MCCDA to MCIx\_CDA, MCDAy to MCIx\_DAy, MCCDB to MCIx\_CDB, MCDBy to MCIx\_DBy.

When the MCI is configured to operate with SD memory cards, the width of the data bus can be selected in the MCI\_SDCR register. Clearing the SDCBUS bit in this register means that the width is one bit; setting it means that the width is four bits. In the case of multimedia cards, only the data line 0 is used. The other data lines can be used as independent PIOs.

## 38.7 MultiMedia Card Operations

After a power-on reset, the cards are initialized by a special message-based MultiMedia Card bus protocol. Each message is represented by one of the following tokens:

- **Command:** A command is a token that starts an operation. A command is sent from the host either to a single card (addressed command) or to all connected cards (broadcast command). A command is transferred serially on the CMD line.
- **Response:** A response is a token which is sent from an addressed card or (synchronously) from all connected cards to the host as an answer to a previously received command. A response is transferred serially on the CMD line.
- **Data:** Data can be transferred from the card to the host or vice versa. Data is transferred via the data line.

Card addressing is implemented using a session address assigned during the initialization phase by the bus controller to all currently connected cards. Their unique CID number identifies individual cards.

The structure of commands, responses and data blocks is described in the MultiMedia-Card System Specification. See also [Table 38-4 on page 630](#).

MultiMediaCard bus data transfers are composed of these tokens.

There are different types of operations. Addressed operations always contain a command and a response token. In addition, some operations have a data token; the others transfer their information directly within the command or response structure. In this case, no data token is present in an operation. The bits on the DAT and the CMD lines are transferred synchronous to the clock MCI Clock.

Two types of data transfer commands are defined:

- **Sequential commands:** These commands initiate a continuous data stream. They are terminated only when a stop command follows on the CMD line. This mode reduces the command overhead to an absolute minimum.
- **Block-oriented commands:** These commands send a data block succeeded by CRC bits.

Both read and write operations allow either single or multiple block transmission. A multiple block transmission is terminated when a stop command follows on the CMD line similarly to the sequential read or when a multiple block transmission has a pre-defined block count (See [“Data Transfer Operation” on page 631](#)).

The MCI provides a set of registers to perform the entire range of MultiMedia Card operations.

### 38.7.1 Command - Response Operation

After reset, the MCI is disabled and becomes valid after setting the MCIEN bit in the MCI\_CR Control Register.

The PWSEN bit saves power by dividing the MCI clock by  $2^{PWSDIV} + 1$  when the bus is inactive.

The two bits, RDPROOF and WRPROOF in the MCI Mode Register (MCI\_MR) allow stopping the MCI Clock during read or write access if the internal FIFO is full. This guarantees data integrity, not bandwidth.

The command and the response of the card are clocked out with the rising edge of the MCI Clock.

All the timings for MultiMedia Card are defined in the MultiMediaCard System Specification.

The two bus modes (open drain and push/pull) needed to process all the operations are defined in the MCI command register. The MCI\_CMDR allows a command to be carried out.

For example, to perform an ALL\_SEND\_CID command:

CMD	Host Command				N <sub>ID</sub> Cycles				CID					
	S	T	Content	CRC	E	Z	*****	Z	S	T	Content	Z	Z	Z

The command ALL\_SEND\_CID and the fields and values for the MCI\_CMDR Control Register are described in [Table 38-4](#) and [Table 38-5](#).

**Table 38-4.** ALL\_SEND\_CID Command Description

CMD Index	Type	Argument	Resp	Abbreviation	Command Description
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line

Note: bcr means broadcast command with response.

**Table 38-5.** Fields and Values for MCI\_CMDR Command Register

Field	Value
CMDNB (command number)	2 (CMD2)
RSPTYP (response type)	2 (R2: 136 bits response)
SPCMD (special command)	0 (not a special command)
OPCMD (open drain command)	1
MAXLAT (max latency for command to response)	0 (NID cycles ==> 5 cycles)
TRCMD (transfer command)	0 (No transfer)
TRDIR (transfer direction)	X (available only in transfer command)
TRTYP (transfer type)	X (available only in transfer command)
IOSPCMD (SDIO special command)	0 (not a special command)

The MCI\_ARGR contains the argument field of the command.

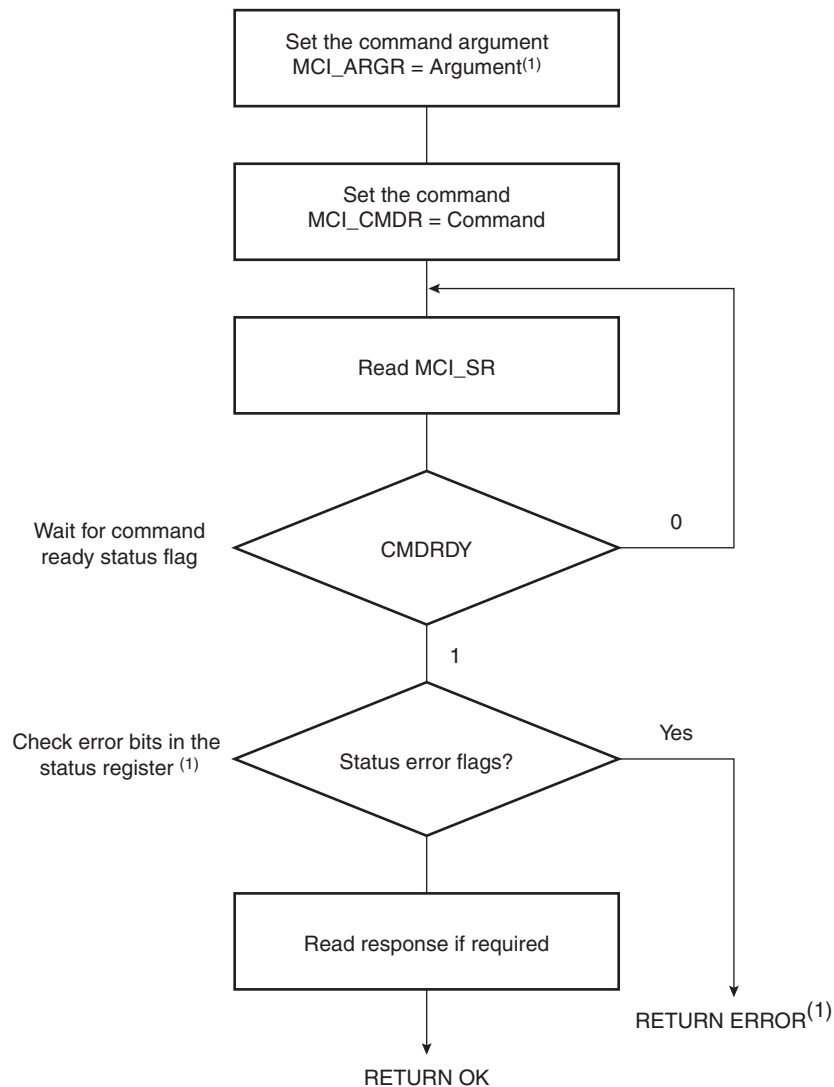
To send a command, the user must perform the following steps:

- Fill the argument register (MCI\_ARGR) with the command argument.
- Set the command register (MCI\_CMDR) (see [Table 38-5](#)).

The command is sent immediately after writing the command register. The status bit CMDRDY in the status register (MCI\_SR) is asserted when the command is completed. If the command requires a response, it can be read in the MCI response register (MCI\_RSPR). The response size can be from 48 bits up to 136 bits depending on the command. The MCI embeds an error detection to prevent any corrupted data during the transfer.

The following flowchart shows how to send a command to the card and read the response if needed. In this example, the status register bits are polled but setting the appropriate bits in the interrupt enable register (MCI\_IER) allows using an interrupt method.

**Figure 38-9.** Command/Response Functional Flow Diagram



Note: 1. If the command is SEND\_OP\_COND, the CRC error flag is always present (refer to R3 response in the MultiMedia Card specification).

### 38.7.2 Data Transfer Operation

The MultiMedia Card allows several read/write operations (single block, multiple blocks, stream, etc.). These kind of transfers can be selected setting the Transfer Type (TRTYP) field in the MCI Command Register (MCI\_CMDR).

These operations can be done using the features of the Peripheral DMA Controller (PDC). If the PDCMODE bit is set in MCI\_MR, then all reads and writes use the PDC facilities.

In all cases, the block length (BLKLEN field) must be defined either in the mode register MCI\_MR, or in the Block Register MCI\_BLKCR. This field determines the size of the data block.

Enabling PDC Force Byte Transfer (PDCFBYTE bit in the MCI\_MR) allows the PDC to manage with internal byte transfers, so that transfer of blocks with a size different from modulo 4 can be supported. When PDC Force Byte Transfer is disabled, the PDC type of transfers are in words, otherwise the type of transfers are in bytes.

Consequent to MMC Specification 3.1, two types of multiple block read (or write) transactions are defined (the host can use either one at any time):

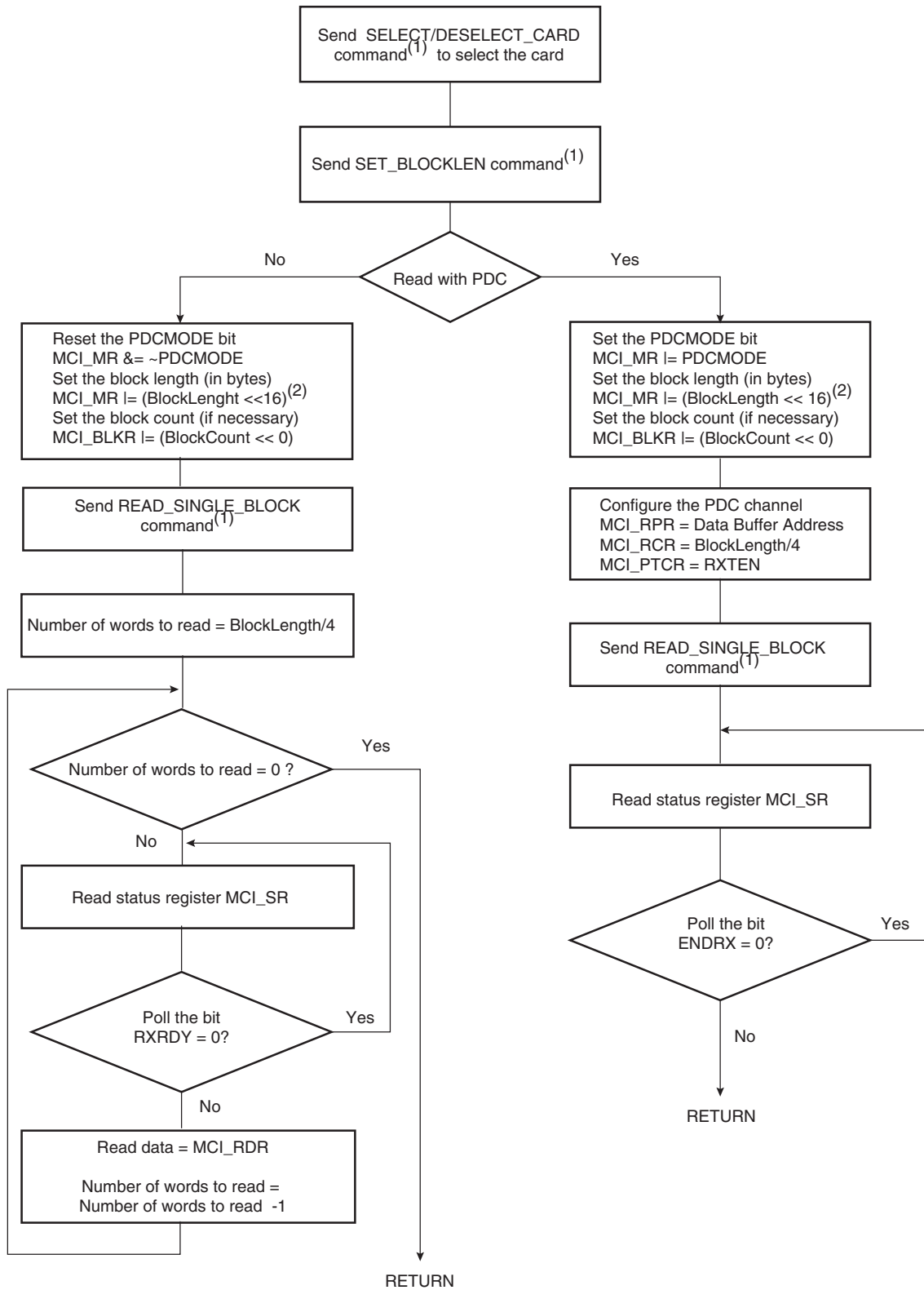
- Open-ended/Infinite Multiple block read (or write):  
The number of blocks for the read (or write) multiple block operation is not defined. The card continuously transfers (or programs) data blocks until a stop transmission command is received.
- Multiple block read (or write) with pre-defined block count (since version 3.1 and higher):  
The card transfers (or programs) the requested number of data blocks and terminate the transaction. The stop command is not required at the end of this type of multiple block read (or write), unless terminated with an error. In order to start a multiple block read (or write) with pre-defined block count, the host must correctly program the MCI Block Register (MCI\_BLK\_R). Otherwise the card starts an open-ended multiple block read. The BCNT field of the Block Register defines the number of blocks to transfer (from 1 to 65535 blocks). Programming the value 0 in the BCNT field corresponds to an infinite block transfer.

### 38.7.3 Read Operation

The following flowchart shows how to read a single block with or without use of PDC facilities. In this example (see [Figure 38-10](#)), a polling method is used to wait for the end of read. Similarly, the user can configure the interrupt enable register (MCI\_IER) to trigger an interrupt at the end of read.



Figure 38-10. Read Functional Flow Diagram



- Note:
1. It is assumed that this command has been correctly sent (see Figure 38-9).
  2. This field is also accessible in the MCI Block Register (MCI\_BLKCR).

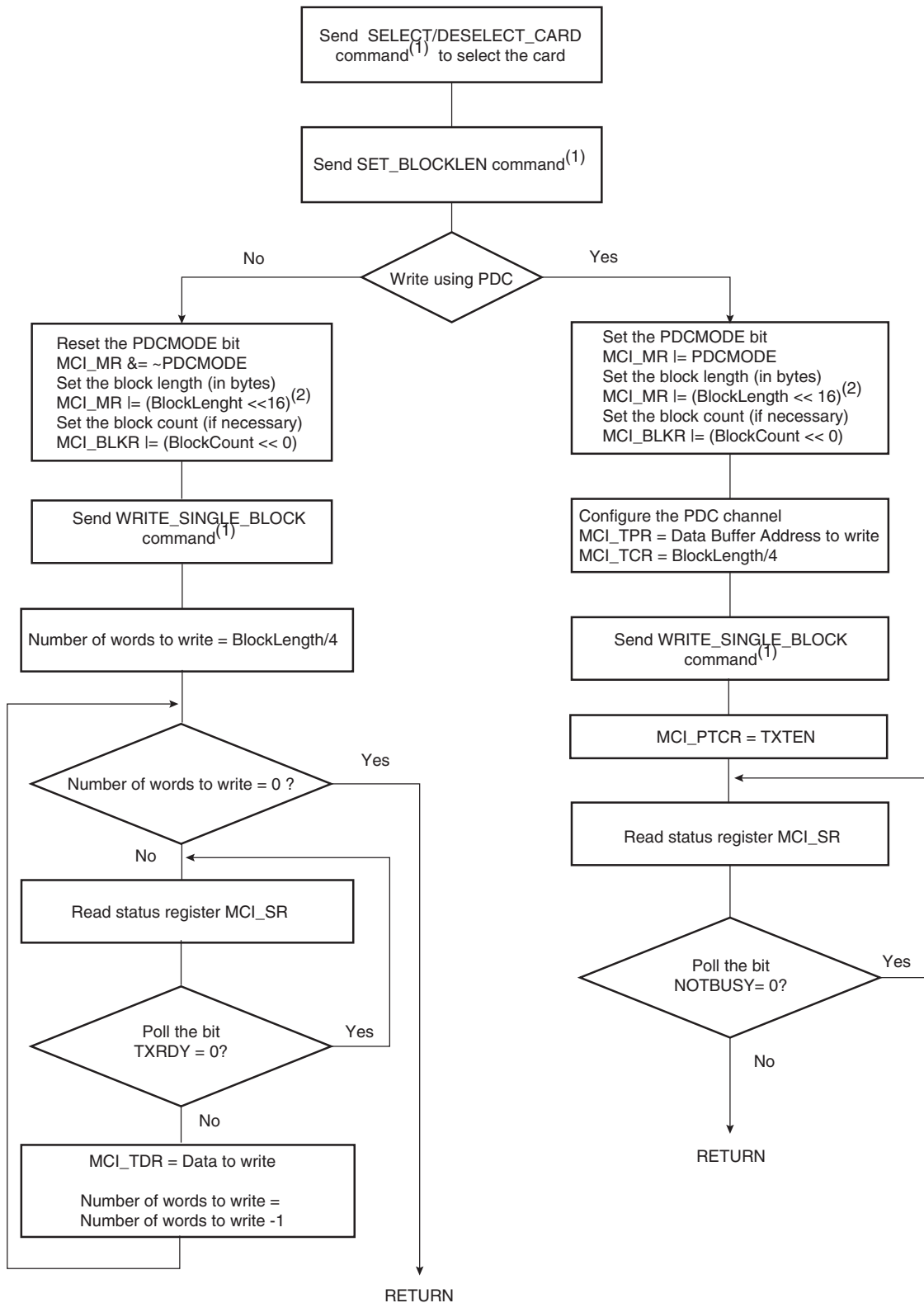
#### 38.7.4 Write Operation

In write operation, the MCI Mode Register (MCI\_MR) is used to define the padding value when writing non-multiple block size. If the bit PDCPADV is 0, then 0x00 value is used when padding data, otherwise 0xFF is used.

If set, the bit PDCMODE enables PDC transfer.

The following flowchart shows how to write a single block with or without use of PDC facilities (see [Figure 38-11](#)). Polling or interrupt method can be used to wait for the end of write according to the contents of the Interrupt Mask Register (MCI\_IMR).

Figure 38-11. Write Functional Flow Diagram

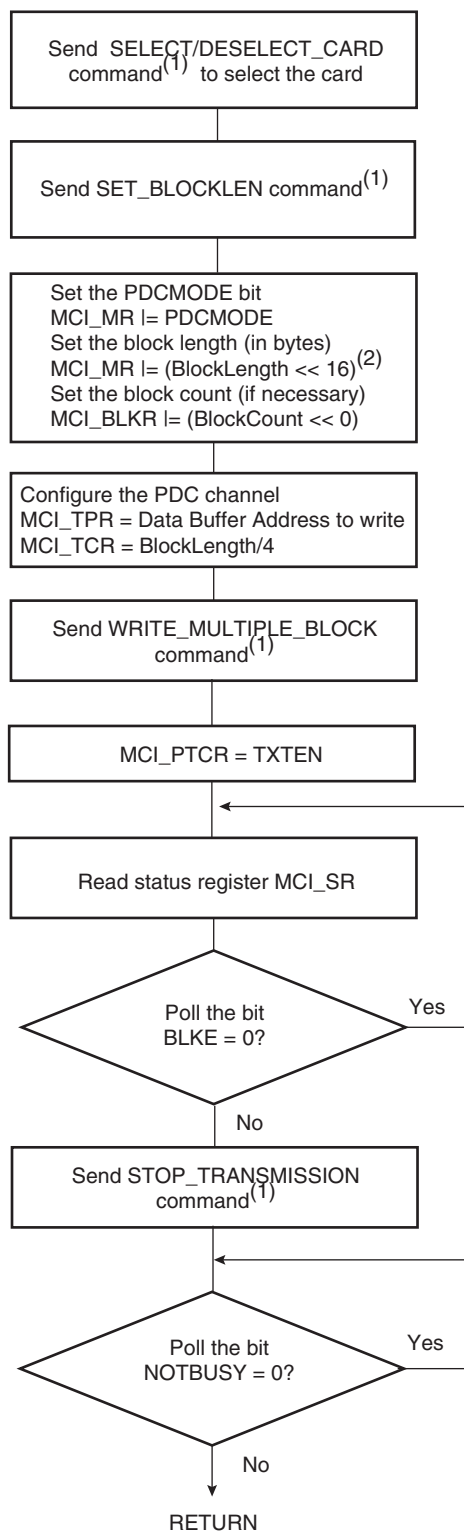


- Note:
1. It is assumed that this command has been correctly sent (see Figure 38-9).
  2. This field is also accessible in the MCI Block Register (MCI\_BLKCR).



The following flowchart shows how to manage a multiple write block transfer with the PDC (see [Figure 38-12](#)). Polling or interrupt method can be used to wait for the end of write according to the contents of the Interrupt Mask Register (MCI\_IMR).

**Figure 38-12.** Multiple Write Functional Flow Diagram



- Note:
1. It is assumed that this command has been correctly sent (see [Figure 38-9](#)).
  2. This field is also accessible in the MCI Block Register (MCI\_BLKR).

## 38.8 SD/SDIO Card Operations

The MultiMedia Card Interface allows processing of SD Memory (Secure Digital Memory Card) and SDIO (SD Input Output) Card commands.

SD/SDIO cards are based on the Multi Media Card (MMC) format, but are physically slightly thicker and feature higher data transfer rates, a lock switch on the side to prevent accidental overwriting and security features. The physical form factor, pin assignment and data transfer protocol are forward-compatible with the MultiMedia Card with some additions. SD slots can actually be used for more than flash memory cards. Devices that support SDIO can use small devices designed for the SD form factor, such as GPS receivers, Wi-Fi or Bluetooth® adapters, modems, barcode readers, IrDA adapters, FM radio tuners, RFID readers, digital cameras and more.

SD/SDIO is covered by numerous patents and trademarks, and licensing is only available through the Secure Digital Card Association.

The SD/SDIO Card communication is based on a 9-pin interface (Clock, Command, 4 x Data and 3 x Power lines). The communication protocol is defined as a part of this specification. The main difference between the SD/SDIO Card and the MultiMedia Card is the initialization process.

The SD/SDIO Card Register (MCI\_SDCR) allows selection of the Card Slot and the data bus width.

The SD/SDIO Card bus allows dynamic configuration of the number of data lines. After power up, by default, the SD/SDIO Card uses only DAT0 for data transfer. After initialization, the host can change the bus width (number of active data lines).

### 38.8.1 SDIO Data Transfer Type

SDIO cards may transfer data in either a multi-byte (1 to 512 bytes) or an optional block format (1 to 511 blocks), while the SD memory cards are fixed in the block transfer mode. The TRTYP field in the MCI Command Register (MCI\_CMDR) allows to choose between SDIO Byte or SDIO Block transfer.

The number of bytes/blocks to transfer is set through the BCNT field in the MCI Block Register (MCI\_BLKCR). In SDIO Block mode, the field BLKLEN must be set to the data block size while this field is not used in SDIO Byte mode.

An SDIO Card can have multiple I/O or combined I/O and memory (called Combo Card). Within a multi-function SDIO or a Combo card, there are multiple devices (I/O and memory) that share access to the SD bus. In order to allow the sharing of access to the host among multiple devices, SDIO and combo cards can implement the optional concept of suspend/resume (Refer to the SDIO Specification for more details). To send a suspend or a resume command, the host must set the SDIO Special Command field (IOSPCMD) in the MCI Command Register.

### 38.8.2 SDIO Interrupts

Each function within an SDIO or Combo card may implement interrupts (Refer to the SDIO Specification for more details). In order to allow the SDIO card to interrupt the host, an interrupt function is added to a pin on the DAT[1] line to signal the card's interrupt to the host. An SDIO interrupt on each slot can be enabled through the MCI Interrupt Enable Register. The SDIO interrupt is sampled regardless of the currently selected slot.

## 38.9 MultiMedia Card Interface (MCI) User Interface

**Table 38-6.** Register Mapping

Offset	Register	Register Name	Read/Write	Reset
0x00	Control Register	MCI_CR	Write	–
0x04	Mode Register	MCI_MR	Read/write	0x0
0x08	Data Timeout Register	MCI_DTOR	Read/write	0x0
0x0C	SD/SDIO Card Register	MCI_SDCR	Read/write	0x0
0x10	Argument Register	MCI_ARGR	Read/write	0x0
0x14	Command Register	MCI_CMDR	Write	–
0x18	Block Register	MCI_BLKCR	Read/write	0x0
0x1C	Reserved	–	–	–
0x20	Response Register <sup>(1)</sup>	MCI_RSPR	Read	0x0
0x24	Response Register <sup>(1)</sup>	MCI_RSPR	Read	0x0
0x28	Response Register <sup>(1)</sup>	MCI_RSPR	Read	0x0
0x2C	Response Register <sup>(1)</sup>	MCI_RSPR	Read	0x0
0x30	Receive Data Register	MCI_RDR	Read	0x0
0x34	Transmit Data Register	MCI_TDR	Write	–
0x38 - 0x3C	Reserved	–	–	–
0x40	Status Register	MCI_SR	Read	0xC0E5
0x44	Interrupt Enable Register	MCI_IER	Write	–
0x48	Interrupt Disable Register	MCI_IDR	Write	–
0x4C	Interrupt Mask Register	MCI_IMR	Read	0x0
0x50-0xFC	Reserved	–	–	–
0x100-0x124	Reserved for the PDC	–	–	–

Note: 1. The response register can be read by N accesses at the same MCI\_RSPR or at consecutive addresses (0x20 to 0x2C). N depends on the size of the response.

### 38.9.1 MCI Control Register

**Name:** MCI\_CR

**Access Type:** Write-only

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

- **MCIEN: Multi-Media Interface Enable**

0 = No effect.

1 = Enables the Multi-Media Interface if MCDIS is 0.

- **MCIDIS: Multi-Media Interface Disable**

0 = No effect.

1 = Disables the Multi-Media Interface.

- **PWSEN: Power Save Mode Enable**

0 = No effect.

1 = Enables the Power Saving Mode if PWSDIS is 0.

**Warning:** Before enabling this mode, the user must set a value different from 0 in the PWSDIV field (Mode Register MCI\_MR).

- **PWSDIS: Power Save Mode Disable**

0 = No effect.

1 = Disables the Power Saving Mode.

- **SWRST: Software Reset**

0 = No effect.

1 = Resets the MCI. A software triggered hardware reset of the MCI interface is performed.



## 38.9.2 MCI Mode Register

**Name:** MCI\_MR

**Access Type:** Read/write

31	30	29	28	27	26	25	24
BLKLEN							
23	22	21	20	19	18	17	16
BLKLEN							
15	14	13	12	11	10	9	8
PDCMODE	PDCPADV	PDCFBYTE	WRPROOF	RDPROOF	PWSDIV		
7	6	5	4	3	2	1	0
CLKDIV							

- **CLKDIV: Clock Divider**

Multimedia Card Interface clock (MCCK or MCI\_CK) is Master Clock (MCK) divided by  $(2^{*(CLKDIV+1)})$ .

- **PWSDIV: Power Saving Divider**

Multimedia Card Interface clock is divided by  $2^{(PWSDIV)} + 1$  when entering Power Saving Mode.

**Warning:** This value must be different from 0 before enabling the Power Save Mode in the MCI\_CR (MCI\_PWSEN bit).

- **RDPROOF Read Proof Enable**

Enabling Read Proof allows to stop the MCI Clock during read access if the internal FIFO is full. This guarantees data integrity, not bandwidth.

0 = Disables Read Proof.

1 = Enables Read Proof.

- **WRPROOF Write Proof Enable**

Enabling Write Proof allows to stop the MCI Clock during write access if the internal FIFO is full. This guarantees data integrity, not bandwidth.

0 = Disables Write Proof.

1 = Enables Write Proof.

- **PDCFBYTE: PDC Force Byte Transfer**

Enabling PDC Force Byte Transfer allows the PDC to manage with internal byte transfers, so that transfer of blocks with a size different from modulo 4 can be supported.

**Warning:** BLKLEN value depends on PDCFBYTE.

0 = Disables PDC Force Byte Transfer. PDC type of transfer are in words.

1 = Enables PDC Force Byte Transfer. PDC type of transfer are in bytes.

- **PDCPADV: PDC Padding Value**

0 = 0x00 value is used when padding data in write transfer (not only PDC transfer).

1 = 0xFF value is used when padding data in write transfer (not only PDC transfer).

- **PDCMODE: PDC-oriented Mode**

0 = Disables PDC transfer

1 = Enables PDC transfer. In this case, UNRE and OVRE flags in the MCI Mode Register (MCI\_SR) are deactivated after the PDC transfer has been completed.

- **BLKLEN: Data Block Length**

This field determines the size of the data block.

This field is also accessible in the MCI Block Register (MCI\_BLKCR).

Bits 16 and 17 must be set to 0 if PDCFBYTE is disabled.

Note: In SDIO Byte mode, BLKLEN field is not used.

## 38.9.3 MCI Data Timeout Register

**Name:** MCI\_DTOR

**Access Type:** Read/write

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

- **DTCYC: Data Timeout Cycle Number**

- **DTOMUL: Data Timeout Multiplier**

These fields determine the maximum number of Master Clock cycles that the MCI waits between two data block transfers. It equals (DTCYC x Multiplier).

Multiplier is defined by DTOMUL as shown in the following table:

DTOMUL			Multiplier
0	0	0	1
0	0	1	16
0	1	0	128
0	1	1	256
1	0	0	1024
1	0	1	4096
1	1	0	65536
1	1	1	1048576

If the data time-out set by DTCYC and DTOMUL has been exceeded, the Data Time-out Error flag (DTOE) in the MCI Status Register (MCI\_SR) raises.

### 38.9.4 MCI SDCard/SDIO Register

**Name:** MCI\_SDCR

**Access Type:** Read/write

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

- **SDCSEL: SDCard/SDIO Slot**

SDCSEL		SDCard/SDIO Slot
0	0	Slot A is selected.
0	1	Slot B selected
1	0	Reserved
1	1	Reserved

- **SDCBUS: SDCard/SDIO Bus Width**

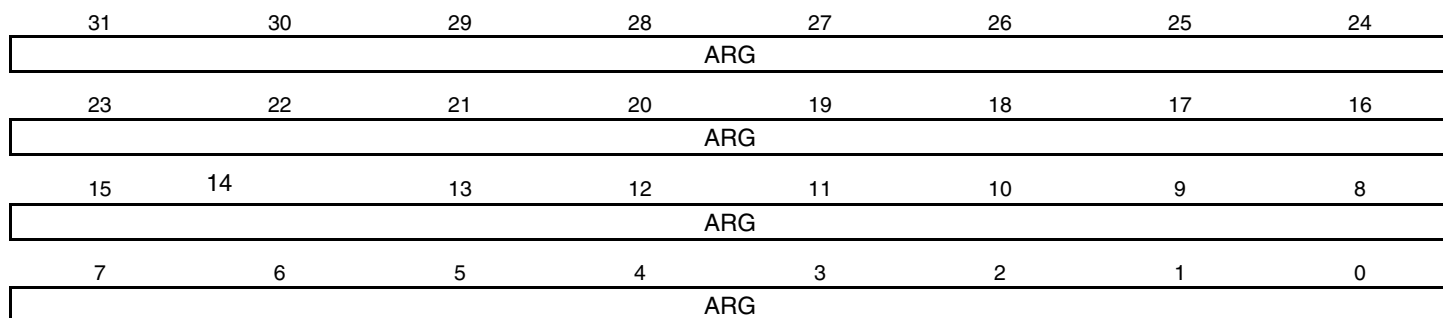
0 = 1-bit data bus

1 = 4-bit data bus

## 38.9.5 MCI Argument Register

Name: MCI\_ARGR

Access Type: Read/write



- ARG: Command Argument

### 38.9.6 MCI Command Register

**Name:** MCI\_CMDR

**Access Type:** Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	IOPCMD	
23	22	21	20	19	18	17	16
–	–	TRTYP			TRDIR	TRCMD	
15	14	13	12	11	10	9	8
–	–	–	MAXLAT	OPDCMD	SPCMD		
7	6	5	4	3	2	1	0
RSPTYP		CMDNB					

This register is write-protected while CMDRDY is 0 in MCI\_SR. If an Interrupt command is sent, this register is only write-able by an interrupt response (field SPCMD). This means that the current command execution cannot be interrupted or modified.

- **CMDNB: Command Number**
- **RSPTYP: Response Type**

RSP		Response Type
0	0	No response.
0	1	48-bit response.
1	0	136-bit response.
1	1	Reserved.

- **SPCMD: Special Command**

SPCMD			Command
0	0	0	Not a special CMD.
0	0	1	Initialization CMD: 74 clock cycles for initialization sequence.
0	1	0	Synchronized CMD: Wait for the end of the current data block transfer before sending the pending command.
0	1	1	Reserved.
1	0	0	Interrupt command: Corresponds to the Interrupt Mode (CMD40).
1	0	1	Interrupt response: Corresponds to the Interrupt Mode (CMD40).

- **OPDCMD: Open Drain Command**

0 = Push pull command

1 = Open drain command

- **MAXLAT: Max Latency for Command to Response**

0 = 5-cycle max latency

1 = 64-cycle max latency

- **TRCMD: Transfer Command**

TRCMD		Transfer Type
0	0	No data transfer
0	1	Start data transfer
1	0	Stop data transfer
1	1	Reserved

- **TRDIR: Transfer Direction**

0 = Write

1 = Read

- **TRTYP: Transfer Type**

TRTYP			Transfer Type
0	0	0	MMC/SDCard Single Block
0	0	1	MMC/SDCard Multiple Block
0	1	0	MMC Stream
0	1	1	Reserved
1	0	0	SDIO Byte
1	0	1	SDIO Block
1	1	0	Reserved
1	1	1	Reserved

- **IOSPCMD: SDIO Special Command**

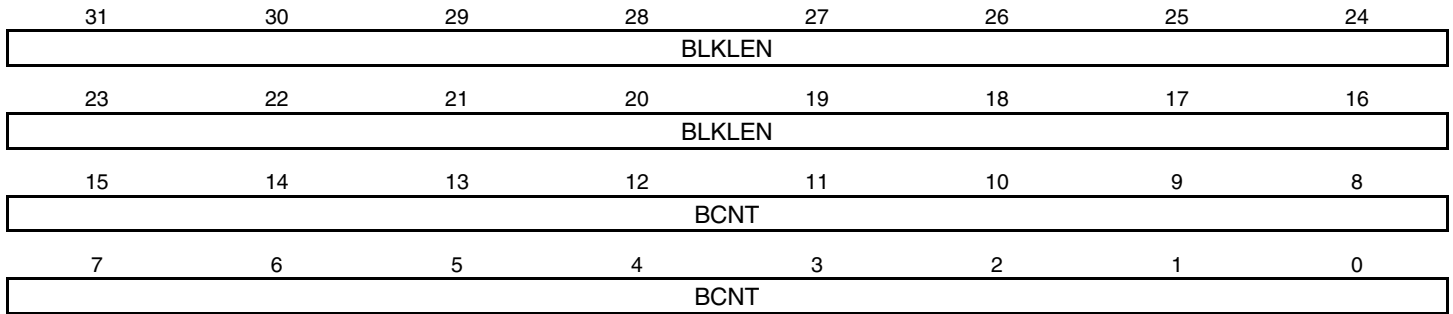
IOSPCMD		SDIO Special Command Type
0	0	Not a SDIO Special Command
0	1	SDIO Suspend Command
1	0	SDIO Resume Command
1	1	Reserved



### 38.9.7 MCI Block Register

Name: MCI\_BLKRR

Access Type: Read/write



• **BCNT: MMC/SDIO Block Count - SDIO Byte Count**

This field determines the number of data byte(s) or block(s) to transfer.

The transfer data type and the authorized values for BCNT field are determined by the TRTYP field in the MCI Command Register (MCI\_CMDR):

TRTYP			Type of Transfer	BCNT Authorized Values
0	0	1	MMC/SDCard Multiple Block	From 1 to 65536: Value 0 corresponds to an infinite block transfer.
1	0	0	SDIO Byte	From 1 to 512 bytes: value 0 corresponds to a 512-byte transfer. Values from 0x200 to 0xFFFF are forbidden.
1	0	1	SDIO Block	From 1 to 511 blocks: value 0 corresponds to an infinite block transfer. Values from 0x200 to 0xFFFF are forbidden.
Other values			-	Reserved.

**Warning:** In SDIO Byte and Block modes, writing to the 7 last bits of BCNT field, is forbidden and may lead to unpredictable results.

• **BLKLEN: Data Block Length**

This field determines the size of the data block.

This field is also accessible in the MCI Mode Register (MCI\_MR).

Bits 16 and 17 must be set to 0 if PDCFBYTE is disabled.

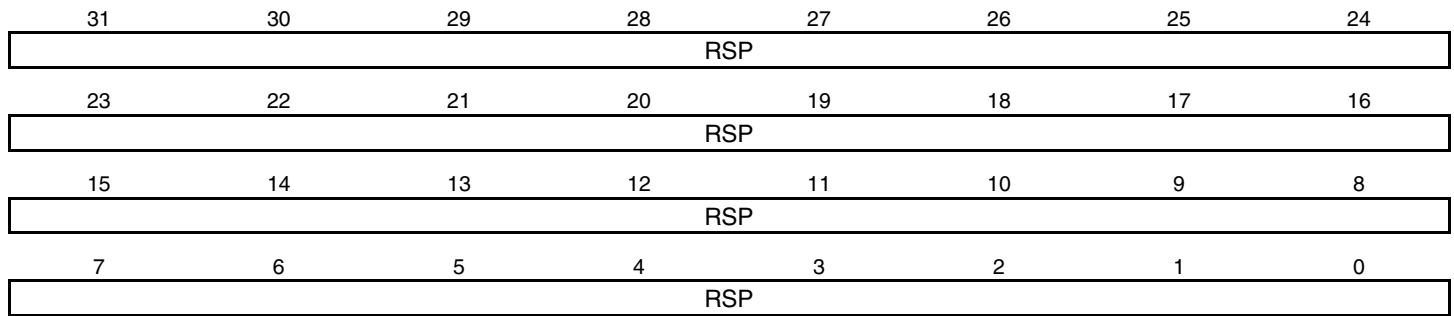
Note: In SDIO Byte mode, BLKLEN field is not used.



## 38.9.8 MCI Response Register

Name: MCI\_RSPR

Access Type: Read-only



- **RSP: Response**

Note: 1. The response register can be read by N accesses at the same MCI\_RSPR or at consecutive addresses (0x20 to 0x2C).  
N depends on the size of the response.



### 38.9.9 MCI Receive Data Register

Name: MCI\_RDR

Access Type: Read-only

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

- DATA: Data to Read

### 38.9.10 MCI Transmit Data Register

Name: MCI\_TDR

Access Type: Write-only

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

- DATA: Data to Write



## 38.9.11 MCI Status Register

**Name:** MCI\_SR

**Access Type:** Read-only

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
TXBUFE	RXBUFF	–	–	–	–	SDIOIRQB	SDIOIRQA
7	6	5	4	3	2	1	0
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

- **CMDRDY: Command Ready**

0 = A command is in progress.

1 = The last command has been sent. Cleared when writing in the MCI\_CMDR.

- **RXRDY: Receiver Ready**

0 = Data has not yet been received since the last read of MCI\_RDR.

1 = Data has been received since the last read of MCI\_RDR.

- **TXRDY: Transmit Ready**

0= The last data written in MCI\_TDR has not yet been transferred in the Shift Register.

1= The last data written in MCI\_TDR has been transferred in the Shift Register.

- **BLKE: Data Block Ended**

This flag must be used only for Write Operations.

0 = A data block transfer is not yet finished. Cleared when reading the MCI\_SR.

1 = A data block transfer has ended, including the CRC16 Status transmission.

In PDC mode (PDCMODE=1), the flag is set when the CRC Status of the last block has been transmitted (TXBUFE already set).

Otherwise (PDCMODE=0), the flag is set for each transmitted CRC Status.

Refer to the MMC or SD Specification for more details concerning the CRC Status.

- **DTIP: Data Transfer in Progress**

0 = No data transfer in progress.

1 = The current data transfer is still in progress, including CRC16 calculation. Cleared at the end of the CRC16 calculation.

- **NOTBUSY: MCI Not Busy**

This flag must be used only for Write Operations.

A block write operation uses a simple busy signalling of the write operation duration on the data (DAT0) line: during a data transfer block, if the card does not have a free data receive buffer, the card indicates this condition by pulling down the data line (DAT0) to LOW. The card stops pulling down the data line as soon as at least one receive buffer for the defined data transfer block length becomes free.

The NOTBUSY flag allows to deal with these different states.

0 = The MCI is not ready for new data transfer. Cleared at the end of the card response.

1 = The MCI is ready for new data transfer. Set when the busy state on the data line has ended. This corresponds to a free internal data receive buffer of the card.

Refer to the MMC or SD Specification for more details concerning the busy behavior.

- **ENDRX: End of RX Buffer**

0 = The Receive Counter Register has not reached 0 since the last write in MCI\_RCR or MCI\_RNCR.

1 = The Receive Counter Register has reached 0 since the last write in MCI\_RCR or MCI\_RNCR.

- **ENDTX: End of TX Buffer**

0 = The Transmit Counter Register has not reached 0 since the last write in MCI\_TCR or MCI\_TNCR.

1 = The Transmit Counter Register has reached 0 since the last write in MCI\_TCR or MCI\_TNCR.

Note: BLKE and NOTBUSY flags can be used to check that the data has been successfully transmitted on the data lines and not only transferred from the PDC to the MCI Controller.

- **RXBUFF: RX Buffer Full**

0 = MCI\_RCR or MCI\_RNCR has a value other than 0.

1 = Both MCI\_RCR and MCI\_RNCR have a value of 0.

- **TXBUFE: TX Buffer Empty**

0 = MCI\_TCR or MCI\_TNCR has a value other than 0.

1 = Both MCI\_TCR and MCI\_TNCR have a value of 0.

Note: BLKE and NOTBUSY flags can be used to check that the data has been successfully transmitted on the data lines and not only transferred from the PDC to the MCI Controller.

- **RINDE: Response Index Error**

0 = No error.

1 = A mismatch is detected between the command index sent and the response index received. Cleared when writing in the MCI\_CMDR.

- **RDIRE: Response Direction Error**

0 = No error.

1 = The direction bit from card to host in the response has not been detected.

- **RCRCE: Response CRC Error**

0 = No error.

1 = A CRC7 error has been detected in the response. Cleared when writing in the MCI\_CMDR.

- **RENDE: Response End Bit Error**

0 = No error.

1 = The end bit of the response has not been detected. Cleared when writing in the MCI\_CMDR.

- **RTOE: Response Time-out Error**

0 = No error.

1 = The response time-out set by MAXLAT in the MCI\_CMDR has been exceeded. Cleared when writing in the MCI\_CMDR.

- **DCRCE: Data CRC Error**

0 = No error.

1 = A CRC16 error has been detected in the last data block. Reset by reading in the MCI\_SR register.

- **DTOE: Data Time-out Error**

0 = No error.

1 = The data time-out set by DTOCYC and DTOMUL in MCI\_DTOR has been exceeded. Reset by reading in the MCI\_SR register.

- **OVRE: Overrun**

0 = No error.

1 = At least one 8-bit received data has been lost (not read). Cleared when sending a new data transfer command.

- **UNRE: Underrun**

0 = No error.

1 = At least one 8-bit data has been sent without valid information (not written). Cleared when sending a new data transfer command.

- **SDIOIRQA: SDIO Interrupt for Slot A**

0 = No interrupt detected on SDIO Slot A.

1 = A SDIO Interrupt on Slot A has reached. Cleared when reading the MCI\_SR.

- **SDIOIRQB: SDIO Interrupt for Slot B**

0 = No interrupt detected on SDIO Slot B.

1 = A SDIO Interrupt on Slot B has reached. Cleared when reading the MCI\_SR.

- **RXBUFF: RX Buffer Full**

0 = MCI\_RCR or MCI\_RNCR has a value other than 0.

1 = Both MCI\_RCR and MCI\_RNCR have a value of 0.

- **TXBUFE: TX Buffer Empty**

0 = MCI\_TCR or MCI\_TNCR has a value other than 0.

1 = Both MCI\_TCR and MCI\_TNCR have a value of 0.

### 38.9.12 MCI Interrupt Enable Register

Name: MCI\_IER

Access Type: Write-only

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
TXBUFE	RXBUFF	–	–	–	–	SDIOIRQB	SDIOIRQA
7	6	5	4	3	2	1	0
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

- **CMDRDY: Command Ready Interrupt Enable**
- **RXRDY: Receiver Ready Interrupt Enable**
- **TXRDY: Transmit Ready Interrupt Enable**
- **BLKE: Data Block Ended Interrupt Enable**
- **DTIP: Data Transfer in Progress Interrupt Enable**
- **NOTBUSY: Data Not Busy Interrupt Enable**
- **ENDRX: End of Receive Buffer Interrupt Enable**
- **ENDTX: End of Transmit Buffer Interrupt Enable**
- **SDIOIRQA: SDIO Interrupt for Slot A Interrupt Enable**
- **SDIOIRQB: SDIO Interrupt for Slot B Interrupt Enable**
- **RXBUFF: Receive Buffer Full Interrupt Enable**
- **TXBUFE: Transmit Buffer Empty Interrupt Enable**
- **RINDE: Response Index Error Interrupt Enable**
- **RDIRE: Response Direction Error Interrupt Enable**
- **RCRCE: Response CRC Error Interrupt Enable**
- **RENDE: Response End Bit Error Interrupt Enable**
- **RTOE: Response Time-out Error Interrupt Enable**
- **DCRCE: Data CRC Error Interrupt Enable**
- **DTOE: Data Time-out Error Interrupt Enable**
- **OVRE: Overrun Interrupt Enable**

- **UNRE: UnderRun Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.





## 38.9.13 MCI Interrupt Disable Register

Name: MCI\_IDR

Access Type: Write-only

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
TXBUFE	RXBUFF	–	–	–	–	SDIOIRQB	SDIOIRQA
7	6	5	4	3	2	1	0
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

- **CMDRDY: Command Ready Interrupt Disable**
- **RXRDY: Receiver Ready Interrupt Disable**
- **TXRDY: Transmit Ready Interrupt Disable**
- **BLKE: Data Block Ended Interrupt Disable**
- **DTIP: Data Transfer in Progress Interrupt Disable**
- **NOTBUSY: Data Not Busy Interrupt Disable**
- **ENDRX: End of Receive Buffer Interrupt Disable**
- **ENDTX: End of Transmit Buffer Interrupt Disable**
- **SDIOIRQA: SDIO Interrupt for Slot A Interrupt Disable**
- **SDIOIRQB: SDIO Interrupt for Slot B Interrupt Disable**
- **RXBUFF: Receive Buffer Full Interrupt Disable**
- **TXBUFE: Transmit Buffer Empty Interrupt Disable**
- **RINDE: Response Index Error Interrupt Disable**
- **RDIRE: Response Direction Error Interrupt Disable**
- **RCRCE: Response CRC Error Interrupt Disable**
- **RENDE: Response End Bit Error Interrupt Disable**
- **RTOE: Response Time-out Error Interrupt Disable**
- **DCRCE: Data CRC Error Interrupt Disable**
- **DTOE: Data Time-out Error Interrupt Disable**
- **OVRE: Overrun Interrupt Disable**



- **UNRE: UnderRun Interrupt Disable**  
0 = No effect.  
1 = Disables the corresponding interrupt.





### 38.9.14 MCI Interrupt Mask Register

Name: MCI\_IMR

Access Type: Read-only

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
TXBUFE	RXBUFF	–	–	–	–	SDIOIRQB	SDIOIRQA
7	6	5	4	3	2	1	0
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

- **CMDRDY: Command Ready Interrupt Mask**
- **RXRDY: Receiver Ready Interrupt Mask**
- **TXRDY: Transmit Ready Interrupt Mask**
- **BLKE: Data Block Ended Interrupt Mask**
- **DTIP: Data Transfer in Progress Interrupt Mask**
- **NOTBUSY: Data Not Busy Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask**
- **ENDTX: End of Transmit Buffer Interrupt Mask**
- **SDIOIRQA: SDIO Interrupt for Slot A Interrupt Mask**
- **SDIOIRQB: SDIO Interrupt for Slot B Interrupt Mask**
- **RXBUFF: Receive Buffer Full Interrupt Mask**
- **TXBUFE: Transmit Buffer Empty Interrupt Mask**
- **RINDE: Response Index Error Interrupt Mask**
- **RDIRE: Response Direction Error Interrupt Mask**
- **RCRCE: Response CRC Error Interrupt Mask**
- **RENDE: Response End Bit Error Interrupt Mask**
- **RTOE: Response Time-out Error Interrupt Mask**
- **DCRCE: Data CRC Error Interrupt Mask**
- **DTOE: Data Time-out Error Interrupt Mask**
- **OVRE: Overrun Interrupt Mask**

- **UNRE: UnderRun Interrupt Mask**

0 = The corresponding interrupt is not enabled.

1 = The corresponding interrupt is enabled.



## 39. LCD Controller (LCDC)

### 39.1 Description

The LCD Controller (LCDC) consists of logic for transferring LCD image data from an external display buffer to an LCD module with integrated common and segment drivers.

The LCD Controller supports single and double scan monochrome and color passive STN LCD modules and single scan active TFT LCD modules. On monochrome STN displays, up to 16 gray shades are supported using a time-based dithering algorithm and Frame Rate Control (FRC) method. This method is also used in color STN displays to generate up to 4096 colors.

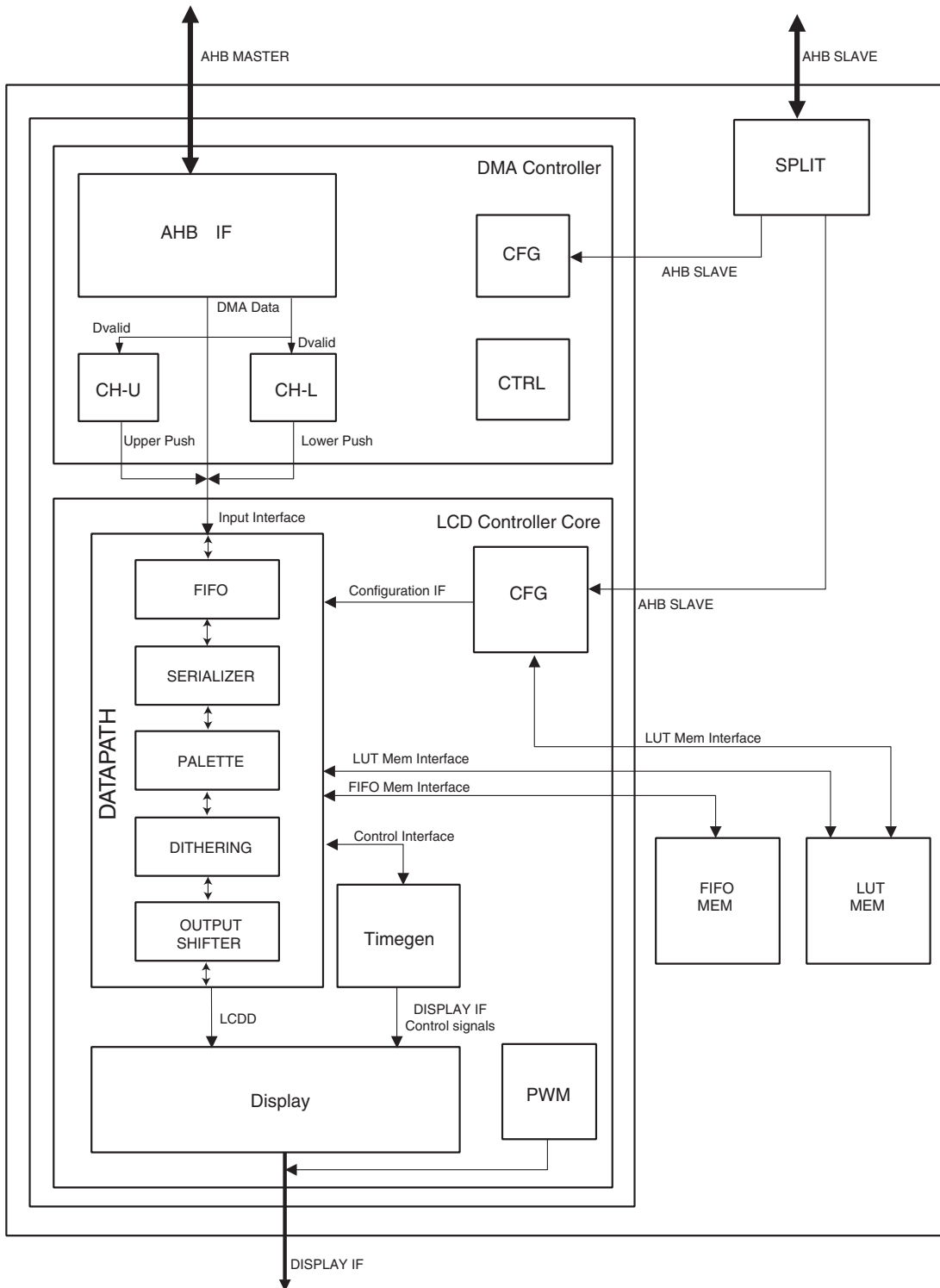
The LCD Controller has a display input buffer (FIFO) to allow a flexible connection of the external AHB master interface, and a lookup table to allow palletized display configurations.

The LCD Controller is programmable in order to support many different requirements such as resolutions up to 2048 x 2048; pixel depth (1, 2, 4, 8, 16, 24 bits per pixel); data line width (4, 8, 16 or 24 bits) and interface timing.

The LCD Controller is connected to the ARM Advanced High Performance Bus (AHB) as a master for reading pixel data. However, the LCD Controller interfaces with the AHB as a slave in order to configure its registers.

## 39.2 Block Diagram

Figure 39-1. LCD Macrocell Block Diagram





## 39.3 I/O Lines Description

**Table 39-1.** I/O Lines Description

Name	Description	Type
LCDDCC	Contrast control signal	Output
LCDHSYNC	Line synchronous signal (STN) or Horizontal synchronous signal (TFT)	Output
LCDDOTCK	LCD clock signal (STN/TFT)	Output
LCDVSYNC	Frame synchronous signal (STN) or Vertical synchronization signal (TFT)	Output
LCDDEN	Data enable signal	Output
LCDD[23:0]	LCD Data Bus output	Output

## 39.4 Product Dependencies

### 39.4.1 I/O Lines

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

### 39.4.2 Power Management

The LCD Controller is not continuously clocked. The user must first enable the LCD Controller clock in the Power Management Controller before using it (PMC\_PCER).

### 39.4.3 Interrupt Sources

The LCD Controller interrupt line is connected to one of the internal sources of the Advanced Interrupt Controller. Using the LCD Controller interrupt requires prior programming of the AIC.

## 39.5 Functional Description

The LCD Controller consists of two main blocks ([Figure 39-1 on page 664](#)), the DMA controller and the LCD controller core (LCDC core). The DMA controller reads the display data from an external memory through a AHB master interface. The LCD controller core formats the display data. The LCD controller core continuously pumps the pixel data into the LCD module via the LCD data bus (LCDD[23:0]); this bus is timed by the LCDDOTCK, LCDDEN, LCDHSYNC, and LCDVSYNC signals.

### 39.5.1 DMA Controller

#### 39.5.1.1 Configuration Block

The configuration block is a set of programmable registers that are used to configure the DMA controller operation. These registers are written via the AHB slave interface. Only word access is allowed.

For details on the configuration registers, see [“LCD Controller \(LCDC\) User Interface” on page 691](#).

#### 39.5.1.2 AHB Interface

This block generates the AHB transactions. It generates undefined-length incrementing bursts as well as 4-, 8- or 16-beat incrementing bursts. The size of the transfer can be configured in the

BRSTLN field of the DMAFRMCFG register. For details on this register, see [“DMA Frame Configuration Register” on page 696](#).

### 39.5.1.3 Channel-U

This block stores the base address and the number of words transferred for this channel (frame in single scan mode and Upper Panel in dual scan mode) since the beginning of the frame. It also generates the end of frame signal.

It has two pointers, the base address and the number of words to transfer. When the module receives a new\_frame signal, it reloads the number of words to transfer pointer with the size of the frame/panel. When the module receives the new\_frame signal, it also reloads the base address with the base address programmed by the host.

The size of the frame/panel can be programmed in the FRMSIZE field of the DMAFRMCFG Register. This size is calculated as follows:

$$\text{Frame\_size} = \left\lceil \frac{\text{X\_size} * \text{Y\_size}}{32} \right\rceil$$

$$\text{X\_size} = ((\text{LINESIZE} + 1) * \text{Bpp} + \text{PIXELOFF}) / 32$$

$$\text{Y\_size} = (\text{LINEVAL} + 1)$$

- LINESIZE is the horizontal size of the display in pixels, minus 1, as programmed in the LINESIZE field of the LCDFRMCFG register of the LCD Controller.
- Bpp is the number of bits per pixel configured.
- PIXELOFF is the pixel offset for 2D addressing, as programmed in the DMA2DCFG register. Applicable only if 2D addressing is being used.
- LINEVAL is the vertical size of the display in pixels, minus 1, as programmed in the LINEVAL field of the LCDFRMCFG register of the LCD Controller.

Note: X\_size is calculated as an up-rounding of a division by 32. (This can also be done adding 31 to the dividend before using an integer division by 32). When using the 2D-addressing mode (see [“2D Memory Addressing” on page 688](#)), it is important to note that the above calculation must be executed and the FRMSIZE field programmed with every movement of the displaying window, since a change in the PIXELOFF field can change the resulting FRMSIZE value.

### 39.5.1.4 Channel-L

This block has the same functionality as Channel-U, but for the Lower Panel in dual scan mode only.

### 39.5.1.5 Control

This block receives the request signals from the LCDC core and generates the requests for the channels.

## 39.5.2 LCD Controller Core

### 39.5.2.1 Configuration Block

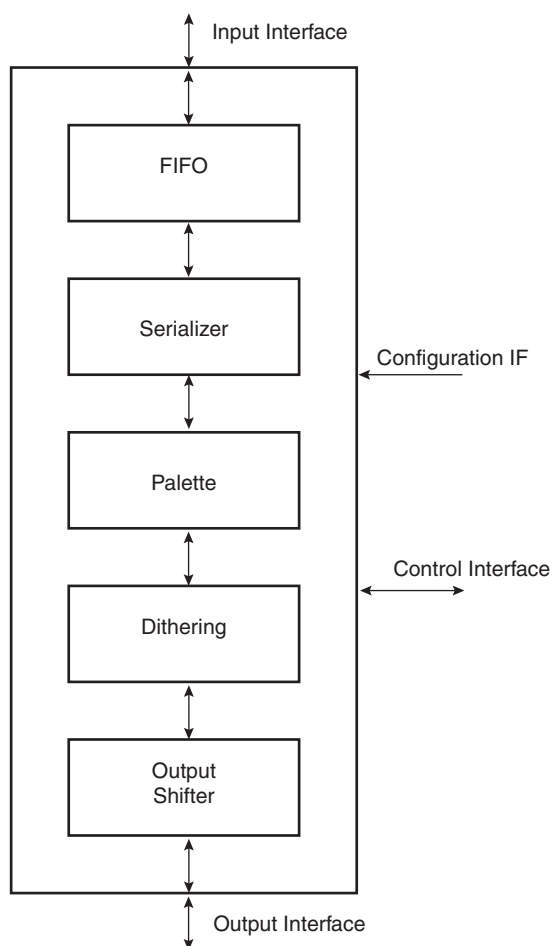
The configuration block is a set of programmable registers that are used to configure the LCDC core operation. These registers are written via the AHB slave interface. Only word access is allowed.

The description of the configuration registers can be found in [“LCD Controller \(LCDC\) User Interface” on page 691](#).

### 39.5.2.2 Datapath

The datapath block contains five submodules: FIFO, Serializer, Palette, Dithering and Shifter. The structure of the datapath is shown in [Figure 39-2](#).

**Figure 39-2.** Datapath Structure



This module transforms the data read from the memory into a format according to the LCD module used. It has four different interfaces: the input interface, the output interface, the configuration interface and the control interface.

- The input interface connects the datapath with the DMA controller. It is a dual FIFO interface with a data bus and two push lines that are used by the DMA controller to fill the FIFOs.

- The output interface is a 24-bit data bus. The configuration of this interface depends on the type of LCD used (TFT or STN, Single or Dual Scan, 4-bit, 8-bit, 16-bit or 24-bit interface).
- The configuration interface connects the datapath with the configuration block. It is used to select between the different datapath configurations.
- The control interface connects the datapath with the timing generation block. The main control signal is the data-request signal, used by the timing generation module to request new data from the datapath.

The datapath can be characterized by two parameters: `initial_latency` and `cycles_per_data`. The parameter `initial_latency` is defined as the number of LCDC Core Clock cycles until the first data is available at the output of the datapath. The parameter `cycles_per_data` is the minimum number of LCDC Core clock cycles between two consecutive data at the output interface.

These parameters are different for the different configurations of the LCD Controller and are shown in [Table 39-2](#).

**Table 39-2.** Datapath Parameters

Configuration			initial_latency	cycles_per_data
DISTYPE	SCAN	IFWIDTH		
TFT			9	1
STN Mono	Single	4	13	4
STN Mono	Single	8	17	8
STN Mono	Dual	8	17	8
STN Mono	Dual	16	25	16
STN Color	Single	4	11	2
STN Color	Single	8	12	3
STN Color	Dual	8	14	4
STN Color	Dual	16	15	6

### 39.5.2.3 FIFO

The FIFO block buffers the input data read by the DMA module. It contains two input FIFOs to be used in Dual Scan configuration that are configured as a single FIFO when used in single scan configuration.

The size of the FIFOs allows a wide range of architectures to be supported.

The upper threshold of the FIFOs can be configured in the FIFOTH field of the LCDFIFO register. The LCDC core will request a DMA transfer when the number of words in each FIFO is less than FIFOTH words. To avoid overwriting in the FIFO and to maximize the FIFO utilization, the FIFOTH should be programmed with:

$$\text{FIFOTH} = 2048 - (2 \times \text{DMA\_BURST\_LENGTH} + 3)$$

where:

- 2048 is the effective size of the FIFO. It is the total FIFO memory size in single scan mode and half that size in dual scan mode.
- `DMA_burst_length` is the burst length of the transfers made by the DMA

## 39.5.2.4 Serializer

This block serializes the data read from memory. It reads words from the FIFO and outputs pixels (1 bit, 2 bits, 4 bits, 8 bits, 16 bits or 24 bits wide) depending on the format specified in the PIXELSIZE field of the LCDCON2 register. It also adapts the memory-ordering format. Both big-endian and little-endian formats are supported. They are configured in the MEMOR field of the LCDCON2 register.

The organization of the pixel data in the memory depends on the configuration and is shown in [Table 39-3](#) and [Table 39-4](#).

Note: For a color depth of 24 bits per pixel there are two different formats supported: packed and unpacked. The packed format needs less memory but has some limitations when working in 2D addressing mode (See “2D Memory Addressing” on page 688.).

**Table 39-3.** Little Endian Memory Organization

Mem Addr	0x3								0x2								0x1								0x0							
	Bit 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
Pixel 1bpp	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
Pixel 2bpp	15		14		13		12		11		10		9		8		7		6		5		4		3		2		1		0	
Pixel 4bpp	7				6				5				4				3				2				1				0			
Pixel 8bpp	3								2								1								0							
Pixel 16bpp	1																0															
Pixel 24bpp packed	1								0																							
Pixel 24bpp packed	2																1															
Pixel 24bpp packed	3																2															
Pixel 24bpp unpacked	not used								0																							

**Table 39-4.** Big Endian Memory Organization

Mem Addr	0x3								0x2								0x1								0x0							
	Bit 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
Pixel 1bpp	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Pixel 2bpp	0		1		2		3		4		5		6		7		8		9		10		11		12		13		14		15	
Pixel 4bpp	0				1				2				3				4				5				6				7			
Pixel 8bpp	0								1								2								3							
Pixel 16bpp	0																1															
Pixel 24bpp packed	0																								1							
Pixel 24bpp packed	1												2																			
Pixel 24bpp packed	2								3																							
Pixel 24bpp packed	4																5															
Pixel 24bpp unpacked	not used								0																							

**Table 39-5.** WinCE Pixel Memory Organization

Mem Addr	0x3								0x2								0x1								0x0							
	Bit 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
Pixel 1bpp	24	25	26	27	28	29	30	31	16	17	18	19	20	21	22	23	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
Pixel 2bpp	12		13		14		15		8		9		10		11		4		5		6		7		0		1		3		3	
Pixel 4bpp	6				7				4				5				2				3				0				1			
Pixel 8bpp	3								2								1								0							
Pixel 16bpp	1																0															
Pixel 24bpp packed	1								0																							
Pixel 24bpp packed	2																1															
Pixel 24bpp packed	3																2															
Pixel 24bpp unpacked	not used								0																							

### 39.5.2.5 Palette

This block is used to generate the pixel gray or color information in palletized configurations. The different modes with the palletized/non-palletized configuration can be found in [Table 39-6](#). In these modes, 1, 2, 4 or 8 input bits index an entry in the lookup table. The corresponding entry in the lookup table contains the color or gray shade information for the pixel.

**Table 39-6.** Palette Configurations

Configuration		Palette
DISTYPE	PIXELSIZE	
TFT	1, 2, 4, 8	Palletized
TFT	16, 24	Non-palletized
STN Mono	1, 2	Palletized
STN Mono	4	Non-palletized
STN Color	1, 2, 4, 8	Palletized
STN Color	16	Non-palletized

The lookup table can be accessed by the host in R/W mode to allow the host to program and check the values stored in the palette. It is mapped in the LCD controller configuration memory map. The LUT is mapped as 16-bit half-words aligned at word boundaries, only word write

access is allowed (the 16 MSB of the bus are not used). For the detailed memory map, see [Table 39-13 on page 691](#).

The lookup table contains 256 16-bit wide entries. The 256 entries are chosen by the programmer from the  $2^{16}$  possible combinations.

For the structure of each LUT entry, see [Table 39-7](#).

**Table 39-7.** Lookup Table Structure in the Memory

Address	Data Output [15:0]			
00	Intensity_bit_0	Blue_value_0[4:0]	Green_value_0[4:0]	Red_value_0[4:0]
01	Intensity_bit_1	Blue_value_1[4:0]	Green_value_1[4:0]	Red_value_1[4:0]
...				
FE	Intensity_bit_254	Blue_value_254[4:0]	Green_value_254[4:0]	Red_value_254[4:0]
FF	Intensity_bit_255	Blue_value_255[4:0]	Green_value_255[4:0]	Red_value_255[4:0]

In STN Monochrome, only the four most significant bits of the red value are used (16 gray shades). In STN Color, only the four most significant bits of the blue, green and red value are used (4096 colors).

In TFT mode, all the bits in the blue, green and red values are used (32768 colors). In this mode, there is also a common intensity bit that can be used to double the possible colors. This bit is the least significant bit of each color component in the LCDD interface (LCDD[18], LCDD[10], LCDD[2]). The LCDD unused bits are tied to 0 when TFT palletized configurations are used (LCDD[17:16], LCDD[9:8], LCDD[1:0]).

### 39.5.2.6 Dithering

The dithering block is used to generate the shades of gray or color when the LCD Controller is used with an STN LCD Module. It uses a time-based dithering algorithm and Frame Rate Control method.

The Frame Rate Control varies the duty cycle for which a given pixel is turned on, giving the display an appearance of multiple shades. In order to reduce the flicker noise caused by turning on and off adjacent pixels at the same time, a time-based dithering algorithm is used to vary the pattern of adjacent pixels every frame. This algorithm is expressed in terms of Dithering Pattern registers (DP\_i) and considers not only the pixel gray level number, but also its horizontal coordinate.

[Table 39-8](#) shows the correspondences between the gray levels and the duty cycle.

**Table 39-8.** Dithering Duty Cycle

Gray Level	Duty Cycle	Pattern Register
15	1	-
14	6/7	DP6_7
13	4/5	DP4_5
12	3/4	DP3_4
11	5/7	DP5_7
10	2/3	DP2_3
9	3/5	DP3_5



**Table 39-8.** Dithering Duty Cycle

Gray Level	Duty Cycle	Pattern Register
8	4/7	DP4_7
7	1/2	~DP1_2
6	3/7	~DP4_7
5	2/5	~DP3_5
4	1/3	~DP2_3
3	1/4	~DP3_4
2	1/5	~DP4_5
1	1/7	~DP6_7
0	0	-

The duty cycles for gray levels 0 and 15 are 0 and 1, respectively.

The same DP\_i register can be used for the pairs for which the sum of duty cycles is 1 (e.g., 1/7 and 6/7). The dithering pattern for the first pair member is the inversion of the one for the second.

The DP\_i registers contain a series of 4-bit patterns. The (3-m)<sup>th</sup> bit of the pattern determines if a pixel with horizontal coordinate  $x = 4n + m$  (n is an integer and m ranges from 0 to 3) should be turned on or off in the current frame. The operation is shown by the examples below.

Consider the pixels a, b, c and d with the horizontal coordinates  $4*n+0$ ,  $4*n+1$ ,  $4*n+2$  and  $4*n+3$ , respectively. The four pixels should be displayed in gray level 9 (duty cycle 3/5) so the register used is DP3\_5 = "1010 0101 1010 0101 1111".

The output sequence obtained in the data output for monochrome mode is shown in [Table 39-9](#).

**Table 39-9.** Dithering Algorithm for Monochrome Mode

Frame Number	Pattern	Pixel a	Pixel b	Pixel c	Pixel d
N	1010	ON	OFF	ON	OFF
N+1	0101	OFF	ON	OFF	ON
N+2	1010	ON	OFF	ON	OFF
N+3	0101	OFF	ON	OFF	ON
N+4	1111	ON	ON	ON	ON
N+5	1010	ON	OFF	ON	OFF
N+6	0101	OFF	ON	OFF	ON
N+7	1010	ON	OFF	ON	OFF
...	...	...	...	...	...

Consider now color display mode and two pixels p0 and p1 with the horizontal coordinates  $4*n+0$ , and  $4*n+1$ . A color pixel is composed of three components: {R, G, B}. Pixel p0 will be displayed sending the color components {R0, G0, B0} to the display. Pixel p1 will be displayed sending the color components {R1, G1, B1}. Suppose that the data read from memory and mapped to the lookup tables corresponds to shade level 10 for the three color components of

both pixels, with the dithering pattern to apply to all of them being DP2\_3 = “1101 1011 0110”. [Table 39-10](#) shows the output sequence in the data output bus for single scan configurations. (In Dual Scan Configuration, each panel data bus acts like in the equivalent single scan configuration.)

**Table 39-10.** Dithering Algorithm for Color Mode

Frame	Signal	Shadow Level	Bit used	Dithering Pattern	4-bit LCDD	8-bit LCDD	Output
N	red_data_0	1010	3	1101	LCDD[3]	LCDD[7]	R0
N	green_data_0	1010	2	1101	LCDD[2]	LCDD[6]	G0
N	blue_data_0	1010	1	1101	LCDD[1]	LCDD[5]	b0
N	red_data_1	1010	0	1101	LCDD[0]	LCDD[4]	R1
N	green_data_1	1010	3	1101	LCDD[3]	LCDD[3]	G1
N	blue_data_1	1010	2	1101	LCDD[2]	LCDD[2]	B1
...	...	...	...	...	...	...	...
N+1	red_data_0	1010	3	1011	LCDD[3]	LCDD[7]	R0
N+1	green_data_0	1010	2	1011	LCDD[2]	LCDD[6]	g0
N+1	blue_data_0	1010	1	1011	LCDD[1]	LCDD[5]	B0
N+1	red_data_1	1010	0	1011	LCDD[0]	LCDD[4]	R1
N+1	green_data_1	1010	3	1011	LCDD[3]	LCDD[3]	G1
N+1	blue_data_1	1010	2	1011	LCDD[2]	LCDD[2]	b1
...	...	...	...	...	...	...	...
N+2	red_data_0	1010	3	0110	LCDD[3]	LCDD[7]	r0
N+2	green_data_0	1010	2	0110	LCDD[2]	LCDD[6]	G0
N+2	blue_data_0	1010	1	0110	LCDD[1]	LCDD[5]	B0
N+2	red_data_1	1010	0	0110	LCDD[0]	LCDD[4]	r1
N+2	green_data_1	1010	3	0110	LCDD[3]	LCDD[3]	g1
N+2	blue_data_1	1010	2	0110	LCDD[2]	LCDD[2]	B1
...	...	...	...	...	...	...	...

Note: Ri = red pixel component ON. Gi = green pixel component ON. Bi = blue pixel component ON. ri = red pixel component OFF. gi = green pixel component OFF. bi = blue pixel component OFF.

### 39.5.2.7 Shifter

The FIFO, Serializer, Palette and Dithering modules process one pixel at a time in monochrome mode and three sub-pixels at a time in color mode (R,G,B components). This module packs the data according to the output interface. This interface can be programmed in the DISTYPE, SCANMOD, and IFWIDTH fields of the LDCCON3 register.

The DISTYPE field selects between TFT, STN monochrome and STN color display. The SCANMODE field selects between single and dual scan modes; in TFT mode, only single scan is supported. The IFWIDTH field configures the width of the interface in STN mode: 4-bit (in single scan mode only), 8-bit and 16-bit (in dual scan mode only).

For a more detailed description of the fields, see [“LCD Controller \(LCDC\) User Interface” on page 691](#).

For a more detailed description of the LCD Interface, see [“LCD Interface” on page 680](#).

## 39.5.2.8 Timegen

The time generator block generates the control signals LCDDOTCK, LCDHSYNC, LCDVSYNC, LCDDEN, used by the LCD module. This block is programmable in order to support different types of LCD modules and obtain the output clock signals, which are derived from the LCDC Core clock.

The LCDDOTCK signal is used to clock the data into the LCD drivers' shift register. The data is sent through LCDD[23:0] synchronized by default with LCDDOTCK falling edge (rising edge can be selected). The CLKVAL field of LCDCON1 register controls the rate of this signal. The divisor can also be bypassed with the BYPASS bit in the LCDCON1 register. In this case, the rate of LCDDOTCK is equal to the frequency of the LCDC Core clock. The minimum period of the LCD-DOTCK signal depends on the configuration. This information can be found in [Table 39-11](#).

$$f_{\text{LCDDOTCK}} = \frac{f_{\text{LCDC\_clock}}}{2 \times \text{CLKVAL}}$$

The LCDDOTCK signal has two different timings that are selected with the CLKMOD field of the LCDCON2 register:

- Always Active (used with TFT LCD Modules)
- Active only when data is available (used with STN LCD Modules)

**Table 39-11.** Minimum LCDDOTCK Period in LCDC Core Clock Cycles

Configuration			LCDDOTCK Period
DISTYPE	SCAN	IFWIDTH	
TFT			1
STN Mono	Single	4	4
STN Mono	Single	8	8
STN Mono	Dual	8	8
STN Mono	Dual	16	16
STN Color	Single	4	2
STN Color	Single	8	2
STN Color	Dual	8	4
STN Color	Dual	16	6

The LCDDEN signal indicates valid data in the LCD Interface.

After each horizontal line of data has been shifted into the LCD, the LCDHSYNC is asserted to cause the line to be displayed on the panel.

The following timing parameters can be configured:

- Vertical to Horizontal Delay (VHDLY): The delay between begin\_of\_line and the generation of LCDHSYNC is configurable in the VHDLY field of the LCDTIM1 register. The delay is equal to (VHDLY+1) LCDDOTCK cycles.
- Horizontal Pulse Width (HPW): The LCDHSYNC pulse width is configurable in HPW field of LCDTIM2 register. The width is equal to (HPW + 1) LCDDOTCK cycles.
- Horizontal Back Porch (HBP): The delay between the LCDHSYNC falling edge and the first LCDDOTCK rising edge with valid data at the LCD Interface is configurable in the HBP field of the LCDTIM2 register. The delay is equal to (HBP+1) LCDDOTCK cycles.
- Horizontal Front Porch (HFP): The delay between end of valid data and the end of the line is configurable in the HFP field of the LCDTIM2 register. The delay is equal to (HFP+1) LCDDOTCK cycles.

There is a limitation in the minimum values of VHDLY, HPW and HBP parameters imposed by the initial latency of the datapath. The total delay in LCDC clock cycles must be higher than or equal to the latency column in [Table 39-2 on page 668](#). This limitation is given by the following formula:

### 39.5.2.9 Equation 1

$$(VHDLY + HPW + HBP + 3) \times PCLK\_PERIOD \geq DPATH\_LATENCY$$

where:

- VHDLY, HPW, HBP are the value of the fields of LCDTIM1 and LCDTIM2 registers
- PCLK\_PERIOD is the period of LCDDOTCK signal measured in LCDC Clock cycles
- DPATH\_LATENCY is the datapath latency of the configuration, given in [Table 39-2 on page 668](#)

The LCDVSYNC is asserted once per frame. This signal is asserted to cause the LCD's line pointer to start over at the top of the display. The timing of this signal depends on the type of LCD: STN or TFT LCD.

In STN mode, the high phase corresponds to the complete first line of the frame. In STN mode, this signal is synchronized with the first active LCDDOTCK rising edge in a line.

In TFT mode, the high phase of this signal starts at the beginning of the first line. The following timing parameters can be selected:

- Vertical Pulse Width (VPW): LCDVSYNC pulse width is configurable in VPW field of the LCDTIM1 register. The pulse width is equal to (VPW+1) lines.
- Vertical Back Porch: Number of inactive lines at the beginning of the frame is configurable in VBP field of LCDTIM1 register. The number of inactive lines is equal to VBP. This field should be programmed with 0 in STN Mode.
- Vertical Front Porch: Number of inactive lines at the end of the frame is configurable in VFP field of LCDTIM2 register. The number of inactive lines is equal to VFP. This field should be programmed with 0 in STN mode.

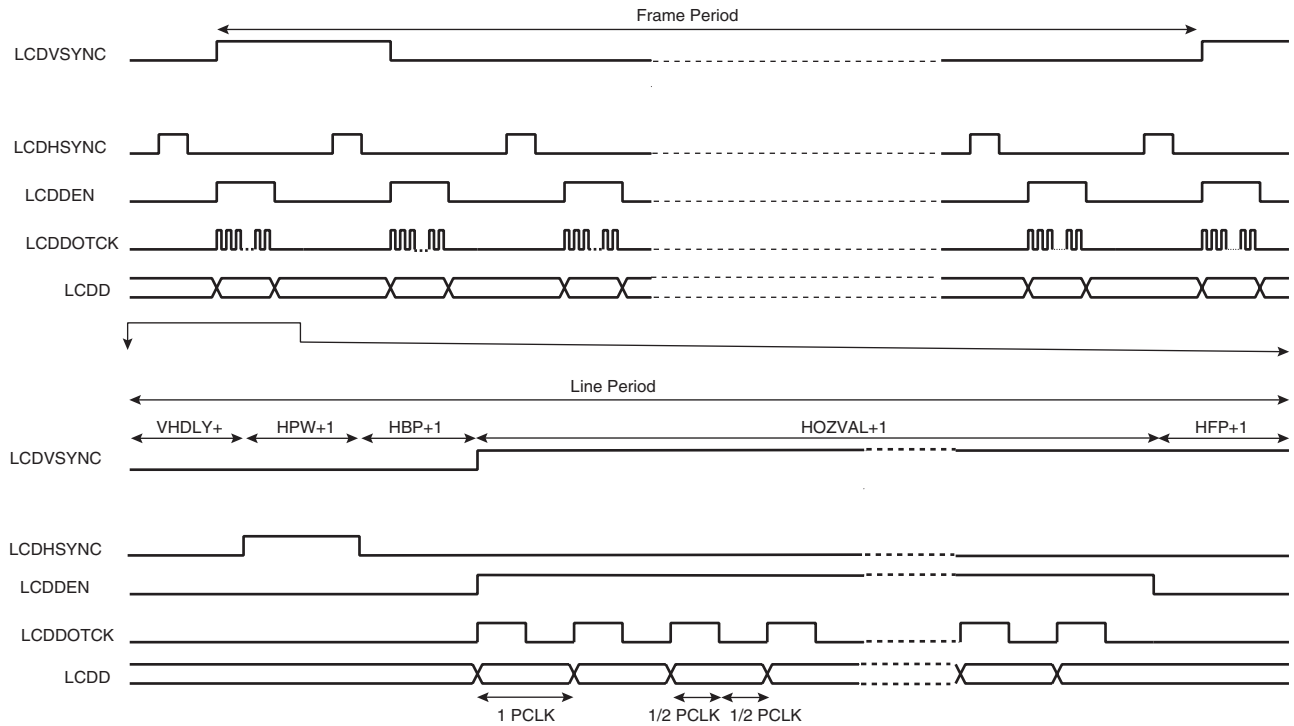
There are two other parameters to configure in this module, the HOZVAL and the LINEVAL fields of the LCDFRMCFG:

- HOZVAL configures the number of active LCDDOTCK cycles in each line. The number of active cycles in each line is equal to (HOZVAL+1) cycles. The minimum value of this parameter is 1.

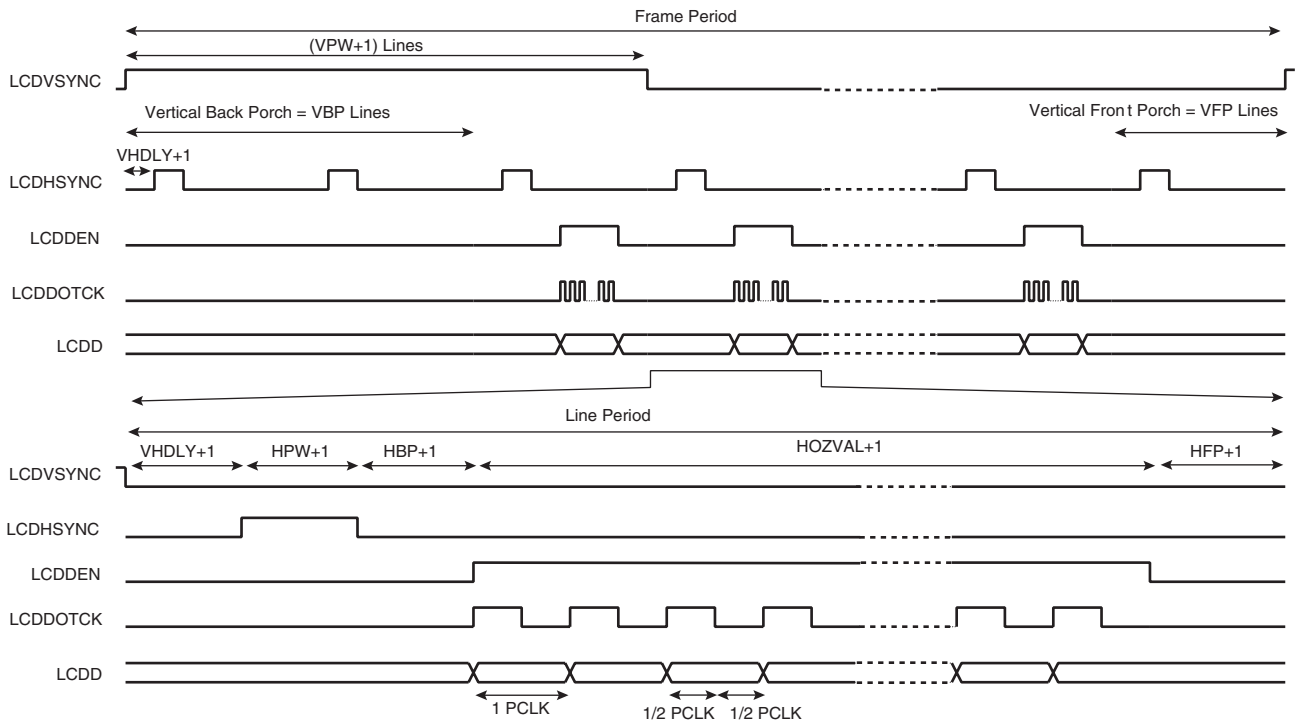
- LINEVAL configures the number of active lines per frame. This number is equal to (LINEVAL+1) lines. The minimum value of this parameter is 1.

Figure 39-3, Figure 39-4 and Figure 39-5 show the timing of LCDDOTCK, LCDDEN, LCDH-SYNC and LCDVSYNC signals:

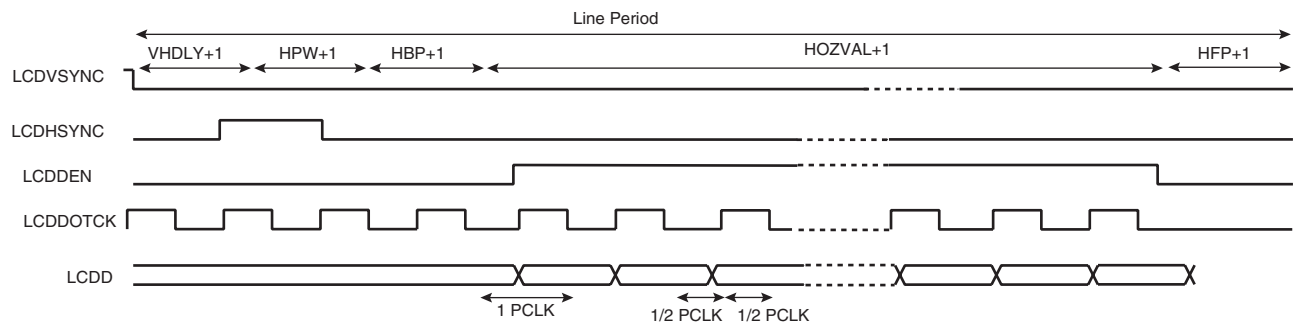
**Figure 39-3.** STN Panel Timing, CLKMOD 0



**Figure 39-4.** TFT Panel Timing, CLKMOD = 0, VPW = 2, VBP = 2, VFP = 1



**Figure 39-5.** TFT Panel Timing (Line Expanded View), CLKMOD=1



Usually the LCD\_FRM rate is about 70 Hz to 75 Hz. It is given by the following equation:

$$\frac{1}{f_{\text{LCDVSYNC}}} = \left( \frac{\text{VHDLY} + \text{HPW} + \text{HBP} + \text{HOZVAL} + \text{HFP} + 5}{f_{\text{LCDDOTCK}}} \right) (\text{VBP} + \text{LINEVAL} + \text{VFP} + 1)$$

where:

- HOZVAL determines the number of LCDDOTCK cycles per line
- LINEVAL determines the number of LCDHSYNC cycles per frame, according to the expressions shown below:

In STN Mode:

$$\text{HOZVAL} = \frac{\text{Horizontal\_display\_size}}{\text{Number\_data\_lines}} - 1$$

$$\text{LINEVAL} = \text{Vertical\_display\_size} - 1$$

In monochrome mode, `Horizontal_display_size` is equal to the number of horizontal pixels. The `number_data_lines` is equal to the number of bits of the interface in single scan mode; `number_data_lines` is equal to half the bits of the interface in dual scan mode.

In color mode, `Horizontal_display_size` equals three times the number of horizontal pixels.

In TFT Mode:

$$\text{HOZVAL} = \text{Horizontal\_display\_size} - 1$$

$$\text{LINEVAL} = \text{Vertical\_display\_size} - 1$$

The frame rate equation is used first without considering the clock periods added at the end beginning or at the end of each line to determine, approximately, the LCDDOTCK rate:

$$f_{\text{lcd\_pclk}} = (\text{HOZVAL} + 5) \times (f_{\text{lcd\_vsync}} \times (\text{LINEVAL} + 1))$$

With this value, the CLKVAL is fixed, as well as the corresponding LCDDOTCK rate.

Then select VHDLY, HPW and HBP according to the type of LCD used and [“Equation 1” on page 676](#).

Finally, the frame rate is adjusted to 70 Hz - 75 Hz with the HFP value:

$$\text{HFP} = f_{\text{LCDDOTCK}} \times \left[ \frac{1}{f_{\text{LCDVSYNC}} \times (\text{LINEVAL} + \text{VBP} + \text{VFP} + 1)} \right] - (\text{VHDLY} + \text{VPW} + \text{VBP} + \text{HOZVAL} + 5)$$

The line counting is controlled by the read-only field LINECNT of LCDCON1 register. The LINECNT field decreases by one unit at each falling edge of LCDHSYNC.

### 39.5.2.10 Display

This block is used to configure the polarity of the data and control signals. The polarity of all clock signals can be configured by LCDCON2[12:8] register setting.

This block also generates the `lcd_pwr` signal internally used to control the state of the LCD pins and to turn on and off by software the LCD module.

This signal is controlled by the PWRCON register and respects the number of frames configured in the GUARD\_TIME field of PWRCON register (PWRCON[7:1]) between the write access to LCD\_PWR field (PWRCON[0]) and the activation/deactivation of `lcd_pwr` signal.

The minimum value for the GUARD\_TIME field is one frame. This gives the DMA Controller enough time to fill the FIFOs before the start of data transfer to the LCD.

### 39.5.2.11 PWM

This block generates the LCD contrast control signal (LCDCC) to make possible the control of the display's contrast by software. This is an 8-bit PWM (Pulse Width Modulation) signal that can be converted to an analog voltage with a simple passive filter.

The PWM module has a free-running counter whose value is compared against a compare register (CONSTRAST\_VAL register). If the value in the counter is less than that in the register, the

output brings the value of the polarity (POL) bit in the PWM control register: CONTRAST\_CTR. Otherwise, the opposite value is output. Thus, a periodic waveform with a pulse width proportional to the value in the compare register is generated.

Due to the comparison mechanism, the output pulse has a width between zero and 255 PWM counter cycles. Thus by adding a simple passive filter outside the chip, an analog voltage between 0 and  $(255/256) \times VDD$  can be obtained (for the positive polarity case, or between  $(1/256) \times VDD$  and  $VDD$  for the negative polarity case). Other voltage values can be obtained by adding active external circuitry.

For PWM mode, the frequency of the counter can be adjusted to four different values using field PS of CONTRAST\_CTR register.

### 39.5.3 LCD Interface

The LCD Controller interfaces with the LCD Module through the LCD Interface ([Table 39-12 on page 685](#)). The Controller supports the following interface configurations: 24-bit TFT single scan, 16-bit STN Dual Scan Mono (Color), 8-bit STN Dual (Single) Scan Mono (Color), 4-bit single scan Mono (Color).

A 4-bit single scan STN display uses 4 parallel data lines to shift data to successive single horizontal lines one at a time until the entire frame has been shifted and transferred. The 4 LSB pins of LCD Data Bus (LCDD [3:0]) can be directly connected to the LCD driver; the 20 MSB pins (LCDD [23:4]) are not used.

An 8-bit single scan STN display uses 8 parallel data lines to shift data to successive single horizontal lines one at a time until the entire frame has been shifted and transferred. The 8 LSB pins of LCD Data Bus (LCDD [7:0]) can be directly connected to the LCD driver; the 16 MSB pins (LCDD [23:8]) are not used.

An 8-bit Dual Scan STN display uses two sets of 4 parallel data lines to shift data to successive upper and lower panel horizontal lines one at a time until the entire frame has been shifted and transferred. The bus LCDD[3:0] is connected to the upper panel data lines and the bus LCDD[7:4] is connected to the lower panel data lines. The rest of the LCD Data Bus lines (LCDD[23:8]) are not used.

A 16-bit Dual Scan STN display uses two sets of 8 parallel data lines to shift data to successive upper and lower panel horizontal lines one at a time until the entire frame has been shifted and transferred. The bus LCDD[7:0] is connected to the upper panel data lines and the bus LCDD[15:8] is connected to the lower panel data lines. The rest of the LCD Data Bus lines (LCDD[23:16]) are not used.

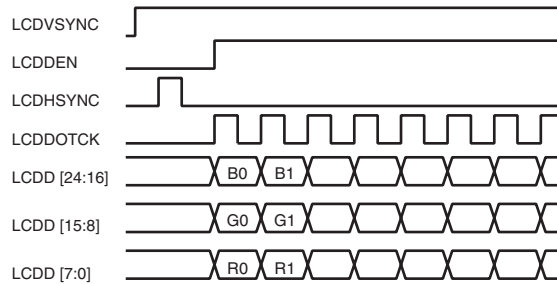
STN Mono displays require one bit of image data per pixel. STN Color displays require three bits (Red, Green and Blue) of image data per pixel, resulting in a horizontal shift register of length three times the number of pixels per horizontal line. This RGB or Monochrome data is shifted to the LCD driver as consecutive bits via the parallel data lines.

A TFT single scan display uses up to 24 parallel data lines to shift data to successive horizontal lines one at a time until the entire frame has been shifted and transferred. The 24 data lines are divided in three bytes that define the color shade of each color component of each pixel. The LCDD bus is split as LCDD[23:16] for the blue component, LCDD[15:8] for the green component and LCDD[7:0] for the red component. If the LCD Module has lower color resolution (fewer bits per color component), only the most significant bits of each component are used.

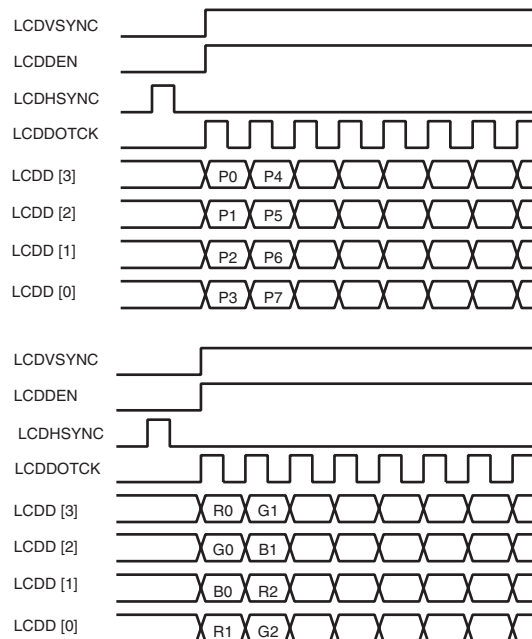


All these interfaces are shown in [Figure 39-6](#) to [Figure 39-10](#). [Figure 39-6](#) on page 681 shows the 24-bit single scan TFT display timing; [Figure 39-7](#) on page 681 shows the 4-bit single scan STN display timing for monochrome and color modes; [Figure 39-8](#) on page 682 shows the 8-bit single scan STN display timing for monochrome and color modes; [Figure 39-9](#) on page 683 shows the 8-bit Dual Scan STN display timing for monochrome and color modes; [Figure 39-10](#) on page 684 shows the 16-bit Dual Scan STN display timing for monochrome and color modes.

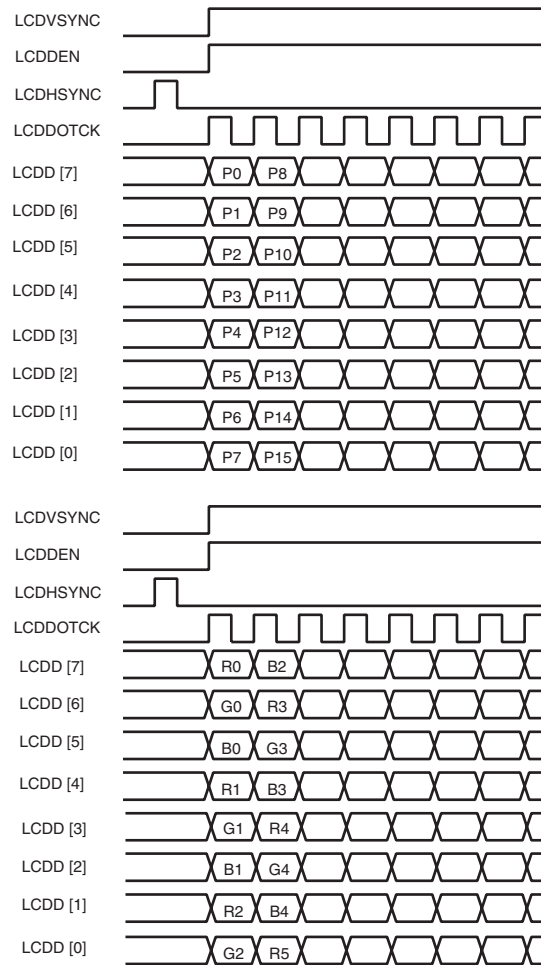
**Figure 39-6.** TFT Timing (First Line Expanded View)



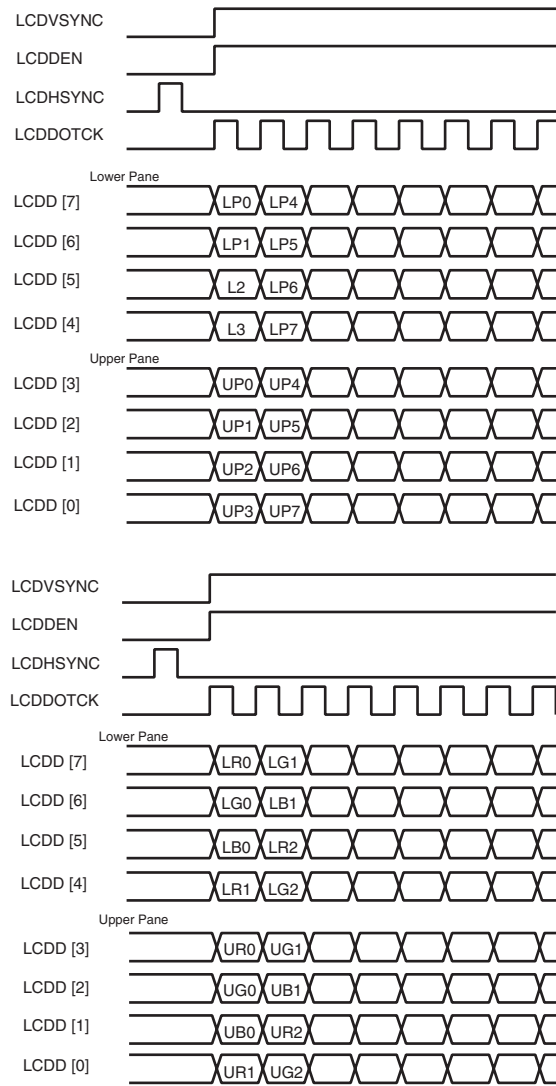
**Figure 39-7.** Single Scan Monochrome and Color 4-bit Panel Timing (First Line Expanded View)



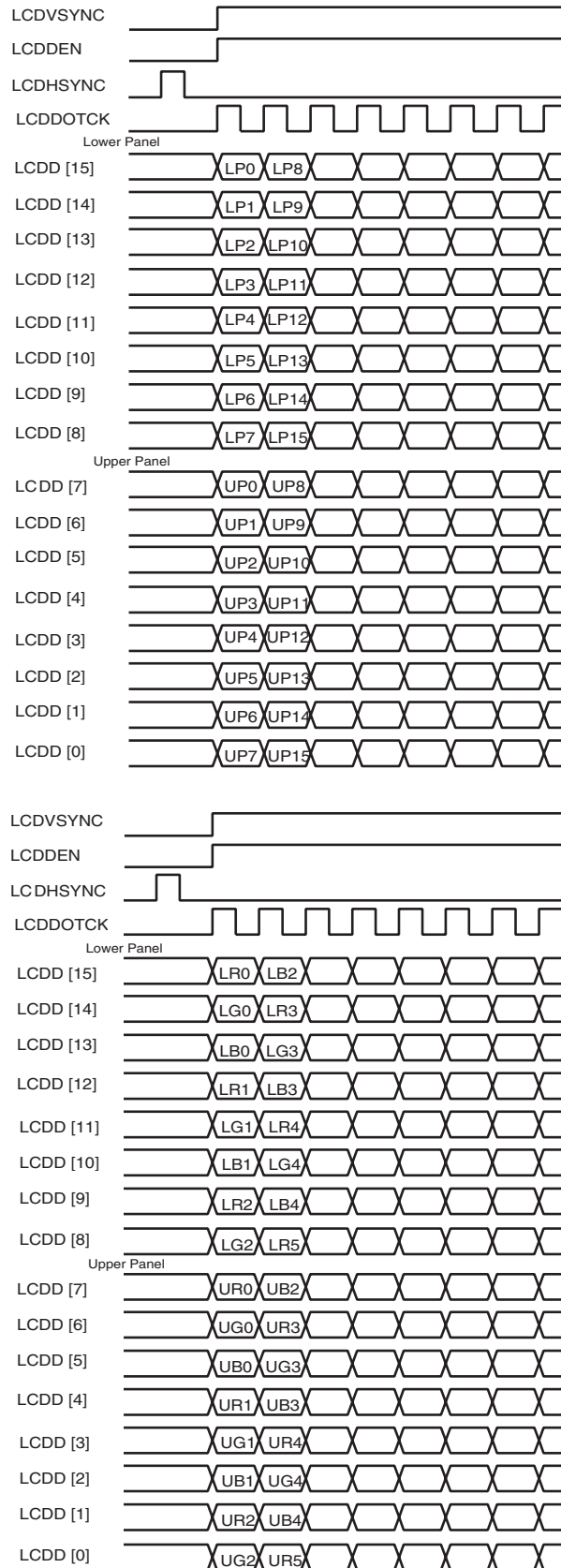
**Figure 39-8.** Single Scan Monochrome and Color 8-bit Panel Timing (First Line Expanded View)



**Figure 39-9.** Dual Scan Monochrome and Color 8-bit Panel Timing (First Line Expanded View)



**Figure 39-10. Dual Scan Monochrome and Color 16-bit Panel Timing (First Line Expanded View)**



**Table 39-12.** LCD Signal Multiplexing

LCD Data Bus	4-bit STN Single Scan (mono, color)	8-bit STN Single Scan (mono, color)	8-bit STN Dual Scan (mono, color)	16-bit STN Dual Scan (mono, color)	24-bit TFT	16-bit TFT
LCDD[23]					LCD_BLUE7	LCD_BLUE4
LCDD[22]					LCD_BLUE6	LCD_BLUE3
LCDD[21]					LCD_BLUE5	LCD_BLUE2
LCDD[20]					LCD_BLUE4	LCD_BLUE1
LCDD[19]					LCD_BLUE3	LCD_BLUE0
LCDD[18]					LCD_BLUE2	Intensity Bit
LCDD[17]					LCD_BLUE1	
LCDD[16]					LCD_BLUE0	
LCDD[15]				LCDLP7	LCD_GREEN7	LCD_GREEN4
LCDD[14]				LCDLP6	LCD_GREEN6	LCD_GREEN3
LCDD[13]				LCDLP5	LCD_GREEN5	LCD_GREEN2
LCDD[12]				LCDLP4	LCD_GREEN4	LCD_GREEN1
LCDD[11]				LCDLP3	LCD_GREEN3	LCD_GREEN0
LCDD[10]				LCDLP2	LCD_GREEN2	Intensity Bit
LCDD[9]				LCDLP1	LCD_GREEN1	
LCDD[8]				LCDLP0	LCD_GREEN0	
LCDD[7]		LCD7	LCDLP3	LCDUP7	LCD_RED7	LCD_RED4
LCDD[6]		LCD6	LCDLP2	LCDUP6	LCD_RED6	LCD_RED3
LCDD[5]		LCD5	LCDLP1	LCDUP5	LCD_RED5	LCD_RED2
LCDD[4]		LCD4	LCDLP0	LCDUP4	LCD_RED4	LCD_RED1
LCDD[3]	LCD3	LCD3	LCDUP3	LCDUP3	LCD_RED3	LCD_RED0
LCDD[2]	LCD2	LCD2	LCDUP2	LCDUP2	LCD_RED2	Intensity Bit
LCDD[1]	LCD1	LCD1	LCDUP1	LCDUP1	LCD_RED1	
LCDD[0]	LCD0	LCD0	LCDUP0	LCDUP0	LCD_RED0	

## 39.6 Interrupts

The LCD Controller generates six different IRQs. All the IRQs are synchronized with the internal LCD Core Clock. The IRQs are:

- DMA Memory error IRQ. Generated when the DMA receives an error response from an AHB slave while it is doing a data transfer.
- FIFO underflow IRQ. Generated when the Serializer tries to read a word from the FIFO when the FIFO is empty.
- FIFO overwrite IRQ. Generated when the DMA Controller tries to write a word in the FIFO while the FIFO is full.
- DMA end of frame IRQ. Generated when the DMA controller updates the Frame Base Address pointers. This IRQ can be used to implement a double-buffer technique. For more information, see [“Double-buffer Technique” on page 687](#).
- End of Line IRQ. This IRQ is generated when the LINEBLANK period of each line is reached and the DMA Controller is in inactive state.
- End of Last Line IRQ. This IRQ is generated when the LINEBLANK period of the last line of the current frame is reached and the DMA Controller is in inactive state.

Each IRQ can be individually enabled, disabled or cleared, in the LCD\_IER (Interrupt Enable Register), LCD\_IDR (Interrupt Disable Register) and LCD\_ICR (Interrupt Clear Register) registers. The LCD\_IMR register contains the mask value for each IRQ source and the LCD\_ISR contains the status of each IRQ source. A more detailed description of these registers can be found in [“LCD Controller \(LCDC\) User Interface” on page 691](#).

## 39.7 Configuration Sequence

The DMA Controller starts to transfer image data when the LCDC Core is activated (Write to LCD\_PWR field of PWRCON register). Thus, the user should configure the LCDC Core and configure and enable the DMA Controller prior to activation of the LCD Controller. In addition, the image data to be shown should be available when the LCDC Core is activated, regardless of the value programmed in the GUARD\_TIME field of the PWRCON register.

To disable the LCD Controller, the user should disable the LCDC Core and then disable the DMA Controller. The user should not enable LIP again until the LCDC Core is in IDLE state. This is checked by reading the LCD\_BUSY bit in the PWRCON register.

The initialization sequence that the user should follow to make the LCDC work is:

- Create or copy the first image to show in the display buffer memory.
- If a palletized mode is used, create and store a palette in the internal LCD Palette memory (See [“Palette” on page 671](#)).
- Configure the LCD Controller Core without enabling it:
  - LCDCON1 register: Program the *CLKVAL* and *BYPASS* fields: these fields control the pixel clock divisor that is used to generate the pixel clock LCDDOTCK. The value to program depends on the LCD Core clock and on the type and size of the LCD Module used. There is a minimum value of the LCDDOTCK clock period that depends on the LCD Controller Configuration, this minimum value can be found in [Table 39-11 on page 675](#). The equations that are used to calculate the value of the pixel clock divisor can be found at the end of the section [“Timegen” on page 675](#)

- LCDCON2 register: Program its fields following their descriptions in the LCD Controller User Interface section below and considering the type of LCD module used and the desired working mode. Consider that not all combinations are possible.
  - LCDTIM1 and LCDTIM2 registers: Program their fields according to the datasheet of the LCD module used and with the help of the Timegen section in page 10. Note that some fields are not applicable to STN modules and must be programmed with 0 values. Note also that there is a limitation on the minimum value of VHDLY, HPW, HBP that depends on the configuration of the LCD.
  - LCDFRMCFG register: program the dimensions of the LCD module used.
  - LCDFIFO register: To program it, use the formula in section “FIFO” on page 668
  - DP1\_2 to DP6\_7 registers: they are only used for STN displays. They contain the dithering patterns used to generate gray shades or colors in these modules. They are loaded with recommended patterns at reset, so it is not necessary to write anything on them. They can be used to improve the image quality in the display by tuning the patterns in each application.
  - PWRCON Register: this register controls the power-up sequence of the LCD, so take care to use it properly. Do not enable the LCD (writing a 1 in LCD\_PWR field) until the previous steps and the configuration of the DMA have been finished.
  - CONTRAST\_CTR and CONTRAST\_VAL: use this registers to adjust the contrast of the display, when the *LCDCC* line is used. □
- Configure the DMA Controller. The user should configure the base address of the display buffer memory, the size of the AHB transaction and the size of the display image in memory. When the DMA is configured the user should enable the DMA. To do so the user should configure the following registers:
    - DMABADDR1 and DMABADDR2 registers: In single scan mode only DMABADDR1 register must be configured with the base address of the display buffer in memory. In dual scan mode DMABADDR1 should be configured with the base address of the Upper Panel display buffer and DMABADDR2 should be configured with the base address of the Lower Panel display buffer.
    - DMAFRMCFG register: Program the FRMSIZE field. Note that in dual scan mode the vertical size to use in the calculation is that of each panel. Respect to the BRSTLN field, a recommended value is a 4-word burst.
    - DMACON register: Once both the LCD Controller Core and the DMA Controller have been configured, enable the DMA Controller by writing a “1” to the DMAEN field of this register. If using a dual scan module or the 2D addressing feature, do not forget to write the DMAUPDT bit after every change to the set of DMA configuration values.
    - DMA2DCFG register: Required only in 2D memory addressing mode (see “2D Memory Addressing” on page 688).
  - Finally, enable the LCD Controller Core by writing a “1” in the LCD\_PWR field of the PWRCON register and do any other action that may be required to turn the LCD module on.

## 39.8 Double-buffer Technique

The double-buffer technique is used to avoid flickering while the frame being displayed is updated. Instead of using a single buffer, there are two different buffers, the backbuffer (background buffer) and the primary buffer (the buffer being displayed).

The host updates the backbuffer while the LCD Controller is displaying the primary buffer. When the backbuffer has been updated the host updates the DMA Base Address registers.

When using a Dual Panel LCD Module, both base address pointers should be updated in the same frame. There are two possibilities:

- Check the DMAFRMPTx register to ensure that there is enough time to update the DMA Base Address registers before the end of frame.
- Update the Frame Base Address Registers when the End Of Frame IRQ is generated.

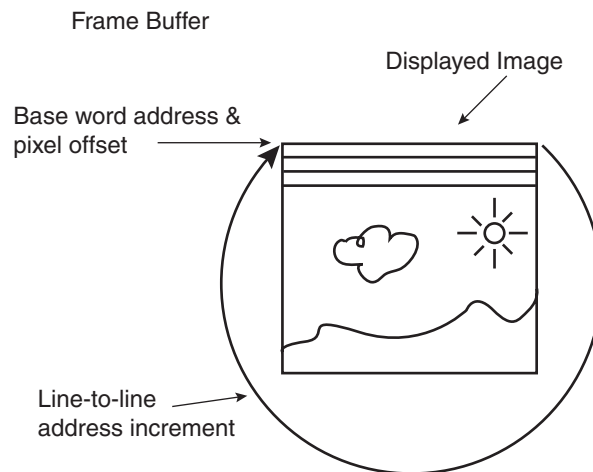
Once the host has updated the Frame Base Address Registers and the next DMA end of frame IRQ arrives, the backbuffer and the primary buffer are swapped and the host can work with the new backbuffer.

When using a dual-panel LCD module, both base address pointers should be updated in the same frame. In order to achieve this, the DMAUPDT bit in DMACON register must be used to validate the new base address.

### 39.9 2D Memory Addressing

The LCDC can be configured to work on a frame buffer larger than the actual screen size. By changing the values in a few registers, it is easy to move the displayed area along the frame buffer width and height.

**Figure 39-11.** Frame Buffer Addressing



In order to locate the displayed window within a larger frame buffer, the software must:

- Program the DMABADDR1 (DMABADDR2) register(s) to make them point to the word containing the first pixel of the area of interest.
- Program the PIXELOFF field of DMA2DCFG register to specify the offset of this first pixel within the 32-bit memory word that contains it.
- Define the width of the complete frame buffer by programming in the field ADDRINC of DMA2DCFG register the address increment between the last word of a line and the first word of the next line (in number of 32-bit words).
- Enable the 2D addressing mode by writing the DMA2DEN bit in DMACON register. If this bit is not activated, the values in the DMA2DCFG register are not considered and the controller assumes that the displayed area occupies a continuous portion of the memory.



The above configuration can be changed frame to frame, so the displayed window can be moved rapidly. Note that the FRMSIZE field of DMAFRMCFG register must be updated with any movement of the displaying window. Note also that the software must write bit DMAUPDT in DMACON register after each configuration for it to be accepted by LCDC.

Note: In 24 bpp packed mode, the DMA base address must point to a word containing a complete pixel (possible values of PIXELOFF are 0 and 8). This means that the horizontal origin of the displaying window must be a multiple of 4 pixels or a multiple of 4 pixels minus 1 ( $x = 4n$  or  $x = 4n-1$ , valid origins are pixel 0,3,4,7,8,11,12, etc.).

## 39.10 Register Configuration Guide

Program the PIO Controller to enable LCD signals.

Enable the LCD controller clock in the Power Management Controller.

### 39.10.1 STN Mode Example

STN color(R,G,B) 320\*240, 8-bit single scan, 70 frames/sec, Master clock = 60 Mhz

Data rate:  $320*240*70*3/8 = 2.016$  MHz

$HOZVAL = ((3*320)/8) - 1$

$LINEVAL = 240 - 1$

$CLKVAL = (60 \text{ MHz} / (2*2.016 \text{ MHz})) - 1 = 14$

$LCDCON1 = CLKVAL \ll 12$

$LCDCON2 = LITTLEENDIAN \mid SINGLESCAN \mid STNCOLOR \mid DISP8BIT \mid PS8BPP;$

$LCDTIM1 = 0;$

$LCDTIM2 = 10 \mid (10 \ll 21);$

$LCDFRMCFG = (HOZVAL \ll 21) \mid LINEVAL;$

$DMAFRMCFG = (7 \ll 24) + (320 * 240 * 8) / 32;$

### 39.10.2 TFT Mode Example

This example is based on the NEC TFT color LCD module NL6448BC20-08.

TFT 640\*480, 16-bit single scan, 60 frames/sec, pixel clock frequency = [21MHz..29MHz] with a typical value = 25.175 MHz.

The Master clock must be  $(2*(n + 1))*\text{pixel clock frequency}$

$HOZVAL = 640 - 1$

$LINEVAL = 480 - 1$

If Master clock is 50 MHz

$CLKVAL = (50 \text{ MHz} / (2*25.175 \text{ MHz})) - 1 = 0$

$VFP = (12 - 1), VBP = (31 - 1), VPW = (2 - 1), VHDLY = (2 - 1)$

$HFP = (16 - 1), HBP = (48 - 1), HPW = (96 - 1)$

$LCDCON1 = CLKVAL \ll 12$

$LCDCON2 = LITTLEENDIAN \mid CLKMOD \mid INVERT\_CLK \mid INVERT\_LINE \mid INVERT\_FRM \mid PS16BPP \mid SINGLESCAN \mid TFT$

$LCDTIM1 = VFP \mid (VBP \ll 8) \mid (VPW \ll 16) \mid (VHDLY \ll 24)$

$LCDTIM2 = HBP \mid (HPW \ll 8) \mid (HFP \ll 21)$

$LCDFRMCFG = (HOZVAL \ll 21) \mid LINEVAL$

$DMAFRMCFG = (7 \ll 24) + (640 * 480 * 16) / 32;$

## 39.11 LCD Controller (LDC) User Interface

**Table 39-13.** LCD Controller (LDC) User Interface

Offset	Register	Register Name	Access	Reset Value
0x0	DMA Base Address Register 1	DMABADDR1	R/W	0x00000000
0x4	DMA Base Address Register 2	DMABADDR2	R/W	0x00000000
0x8	DMA Frame Pointer Register 1	DMAFRMPT1	Read-only	0x00000000
0xC	DMA Frame Pointer Register 2	DMAFRMPT2	Read-only	0x00000000
0x10	DMA Frame Address Register 1	DMAFRMADD1	Read-only	0x00000000
0x14	DMA Frame Address Register 2	DMAFRMADD2	Read-only	0x00000000
0x18	DMA Frame Configuration Register	DMAFRMCFG	R/W	0x00000000
0x1C	DMA Control Register	DMACON	R/W	0x00000000
0x20	DMA Control Register	DMA2DCFG	R/W	0x00000000
0x800	LCD Control Register 1	LCDCON1	R/W	0x00002000
0x804	LCD Control Register 2	LCDCON2	R/W	0x00000000
0x808	LCD Timing Register 1	LCDTIM1	R/W	0x00000000
0x80C	LCD Timing Register 2	LCDTIM2	R/W	0x00000000
0x810	LCD Frame Configuration Register	LCDFRMCFG	R/W	0x00000000
0x814	LCD FIFO Register	LCDFIFO	R/W	0x00000000
0x818	Reserved	–	–	–
0x81C	Dithering Pattern DP1_2	DP1_2	R/W	0xA5
0x820	Dithering Pattern DP4_7	DP4_7	R/W	0x5AF0FA5
0x824	Dithering Pattern DP3_5	DP3_5	R/W	0xA5A5F
0x828	Dithering Pattern DP2_3	DP2_3	R/W	0xA5F
0x82C	Dithering Pattern DP5_7	DP5_7	R/W	0xFAF5FA5
0x830	Dithering Pattern DP3_4	DP3_4	R/W	0xFAF5
0x834	Dithering Pattern DP4_5	DP4_5	R/W	0xFAF5F
0x838	Dithering Pattern DP6_7	DP6_7	R/W	0xF5FFAFF
0x83C	Power Control Register	PWRCON	R/W	0x0000000e
0x840	Contrast Control Register	CONTRAST_CTR	R/W	0x00000000
0x844	Contrast Value Register	CONTRAST_VAL	R/W	0x00000000
0x848	LCD Interrupt Enable Register	LCD_IER	Write-only	0x0
0x84C	LCD Interrupt Disable Register	LCD_IDR	Write-only	0x0
0x850	LCD Interrupt Mask Register	LCD_IMR	Read-only	0x0
0x854	LCD Interrupt Status Register	LCD_ISR	Read-only	0x0
0x858	LCD Interrupt Clear Register	LCD_ICR	Write-only	0x0
0x860	LCD Interrupt Test Register	LCD_ITR	Write-only	0
0x864	LCD Interrupt Raw Status Register	LCD_IRR	Read-only	0

**Table 39-13.** LCD Controller (LCDC) User Interface (Continued)

Offset	Register	Register Name	Access	Reset Value
0xC00	Palette entry 0	LUT ENTRY 0	R/W	
0xC04	Palette entry 1	LUT ENTRY 1	R/W	
0xC08	Palette entry 2	LUT ENTRY 2	R/W	
0xC0C	Palette entry 3	LUT ENTRY 3	R/W	
...		...		
0xFFC	Palette entry 255	LUT ENTRY 255	R/W	

## 39.11.1 DMA Base Address Register 1

**Name:** DMABADDR1

**Access:** Read/Write

**Reset value:** 0x00000000

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

- **BADDR-U**

Base Address for the upper panel in dual scan mode. Base Address for the complete frame in single scan mode.

If a dual scan configuration is selected in LCDCON2 register or bit DMA2DEN in register DMACON is set, the bit DMAUPDT in that same register must be written after writing any new value to this field in order to make the DMA controller use this new value.

## 39.11.2 DMA Base Address Register 2

**Name:** DMABADDR2

**Access:** Read/Write

**Reset value:** 0x00000000

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

- **BADDR-L**

Base Address for the lower panel in dual scan mode only.

If a dual scan configuration is selected in LCDCON2 register or bit DMA2DEN in register DMACON is set, the bit DMAUPDT in that same register must be written after writing any new value to this field in order to make the DMA controller use this new value.

### 39.11.3 DMA Frame Pointer Register 1

**Name:** DMAFRMPT1

**Access:** Read-only

**Reset value:** 0x00000000

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

- **FRMPT-U**

Current value of frame pointer for the upper panel in dual scan mode. Current value of frame pointer for the complete frame in single scan mode. Down count from FRMSIZE to 0.

**Note:** This register is read-only and contains the current value of the frame pointer (number of words to the end of the frame). It can be used as an estimation of the number of words transferred from memory for the current frame.

### 39.11.4 DMA Frame Pointer Register 2

**Name:** DMAFRMPT2

**Access:** Read-only

**Reset value:** 0x00000000

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

- **FRMPT-L**

Current value of frame pointer for the Lower panel in dual scan mode only. Down count from FRMSIZE to 0.

**Note:** This register is read-only and contains the current value of the frame pointer (number of words to the end of the frame). It can be used as an estimation of the number of words transferred from memory for the current frame.

## 39.11.5 DMA Frame Address Register 1

**Name:** DMAFRMADD1

**Access:** Read-only

**Reset value:** 0x00000000

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

- **FRMADD-U**

Current value of frame address for the upper panel in dual scan mode. Current value of frame address for the complete frame in single scan.

Note: This register is read-only and contains the current value of the last DMA transaction in the bus for the panel/frame.

## 39.11.6 DMA Frame Address Register 2

**Name:** DMAFRMADD2

**Access:** Read-only

**Reset value:** 0x00000000

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

- **FRMADD-L**

Current value of frame address for the lower panel in single scan mode only.

Note: This register is read-only and contains the current value of the last DMA transaction in the bus for the panel.

### 39.11.7 DMA Frame Configuration Register

**Name:** DMAFRMCFG

**Access:** Read/Write

**Reset value:** 0x00000000

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

- **FRMSIZE: Frame Size**

In single scan mode, this is the frame size in words. In dual scan mode, this is the size of each panel. If a dual scan configuration is selected in LCDCON2 register or bit DMA2DEN in register DMACON is set, the bit DMAUPDT in that same register must be written after writing any new value to this field in order to make the DMA controller use this new value.

- **BRSTLN: Burst Length**

Program with the desired burst length - 1



## 39.11.8 DMA Control Register

**Name:** DMACON

**Access:** 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
–	–	–	DMA2DEN	DMAUPDT	DMABUSY	DMARST	DMAEN

- **DMAEN: DMA Enable**

0: DMA is disabled.

1: DMA is enabled.

- **DMARST: DMA Reset (Write-only)**

0: No effect.

1: Reset DMA module. DMA Module should be reset only when disabled and in idle state.

- **DMABUSY: DMA Busy**

0: DMA module is idle.

1: DMA module is busy (doing a transaction on the AHB bus).

- **DMAUPDT: DMA Configuration Update**

0: No effect

1: Update DMA Configuration. Used for simultaneous updating of DMA parameters in dual scan mode or when using 2D addressing. The values written in the registers DMABADDR1, DMABADDR2 and DMA2DCFG, and in the field FRMSIZE of register DMAFRMCFG, are accepted by the DMA controller and are applied at the next frame. This bit is used only if a dual scan configuration is selected (bit SCANMOD of LCDCON2 register) or 2D addressing is enabled (bit DMA2DEN in this register). Otherwise, the LCD controller accepts immediately the values written in the registers referred to above.

- **DMA2DEN: DMA 2D Addressing Enable**

0: 2D addressing is disabled (values in register DMA2DCFG are “don’t care”).

1: 2D addressing is enabled.

### 39.11.9 LCD DMA 2D Addressing Register

**Name:** DMA2DCFG

**Access:** Read/Write

**Reset value:** 0x00000000

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

- **ADDRINC: DMA 2D Addressing Address increment**

When 2-D DMA addressing is enabled (bit DMA2DEN is set in register DMACON), this field specifies the number of bytes that the DMA controller must jump between screen lines. Itb must be programmed as: [({address of first 32-bit word in a screen line} - {address of last 32-bit word in previous line})]. In other words, it is equal to 4\*[number of 32-bit words occupied by each line in the complete frame buffer minus the number of 32-bit words occupied by each displayed line]. Bit DMAUPDT in register DMACON must be written after writing any new value to this field in order to make the DMA controller use this new value.

- **PIXELOFF: DAM2D Addressing Pixel offset**

When 2D DMA addressing is enabled (bit DMA2DEN is set in register DMACON), this field specifies the offset of the first pixel in each line within the memory word that contains this pixel. The offset is specified in number of bits in the range 0-31, so for example a value of 4 indicates that the first pixel in the screen starts at bit 4 of the 32-bit word pointed by register DMABADDR1. Bits 0 to 3 of that word are not used. This example is valid for little endian memory organization. When using big endian memory organization, this offset is considered from bit 31 downwards, or equivalently, a given value of this field always selects the pixel in the same relative position within the word, independently of the memory ordering configuration. Bit DMAUPDT in register DMACON must be written after writing any new value to this field in order to make the DMA controller use this new value.

## 39.11.10 LCD Control Register 1

**Name:** LCDCON1

**Access:** Read/Write, except LINECNT: Read-only

**Reset value:** 0x00002000

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

- **BYPASS: Bypass LCDDOTCK divider**

0: The divider is not bypassed. LCDDOTCK frequency defined by the CLKVAL field.

1: The LCDDOTCK divider is bypassed. LCDDOTCK frequency is equal to the LCDC Clock frequency.

- **CLKVAL: Clock divider**

9-bit divider for pixel clock (LCDDOTCK) frequency.

$$\text{Pixel\_clock} = \text{system\_clock} / (\text{CLKVAL} + 1) \times 2$$

- **LINECNT: Line Counter (Read-only)**

Current Value of 11-bit line counter. Down count from LINEVAL to 0.

### 39.11.11 LCD Control Register 2

Name: LCDCON2

Access: Read/Write

Reset value: 0x00000000

31	30	29	28	27	26	25	24
MEMOR		–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CLKMOD	–	–	INVDVAL	INVCLK	INVLIN	INVFRAME	INVVD
7	6	5	4	3	2	1	0
PIXELSIZE			IFWIDTH		SCANMOD	DISTYPE	

- **DISTYPE: Display Type**

DISTYPE		
0	0	STN Monochrome
0	1	STN Color
1	0	TFT
1	1	Reserved

- **SCANMOD: Scan Mode**

0: Single Scan

1: Dual Scan

- **IFWIDTH: Interface width (STN)**

IFWIDTH		
0	0	4-bit (Only valid in single scan STN mono or color)
0	1	8-bit (Only valid in STN mono or Color)
1	0	16-bit (Only valid in dual scan STN mono or color)
1	1	Reserved

- **PIXELSIZE: Bits per pixel**

PIXELSIZE			
0	0	0	1 bit per pixel
0	0	1	2 bits per pixel
0	1	0	4 bits per pixel
0	1	1	8 bits per pixel
1	0	0	16 bits per pixel
1	0	1	24 bits per pixel, packed (Only valid in TFT mode)
1	1	0	24 bits per pixel, unpacked (Only valid in TFT mode)
1	1	1	Reserved

- **INVVD: LCDD polarity**

0: Normal

1: Inverted

- **INVFRAME: LCDVSYNC polarity**

0: Normal (active high)

1: Inverted (active low)

- **INVLIN: LCDHSYNC polarity**

0: Normal (active high)

1: Inverted (active low)

- **INVCLK: LCDDOTCK polarity**

0: Normal (LCDD fetched at LCDDOTCK falling edge)

1: Inverted (LCDD fetched at LCDDOTCK rising edge)

- **INVDVAL: LCDDEN polarity**

0: Normal (active high)

1: Inverted (active low)

- **CLKMOD: LCDDOTCK mode**

0: LCDDOTCK only active during active display period

1: LCDDOTCK always active

- **MEMOR: Memory Ordering Format**

00: Big Endian

10: Little Endian

11: WinCE format

### 39.11.12 LCD Timing Configuration Register 1

**Name:** LCDTIM1

**Access:** Read/Write

**Reset value:** 0x00000000

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

- **VFP: Vertical Front Porch**

In TFT mode, these bits equal the number of idle lines at the end of the frame.

In STN mode, these bits should be set to 0.

- **VBP: Vertical Back Porch**

In TFT mode, these bits equal the number of idle lines at the beginning of the frame.

In STN mode, these bits should be set to 0.

- **VPW: Vertical Synchronization pulse width**

In TFT mode, these bits equal the vertical synchronization pulse width, given in number of lines. LCDVSYNC width is equal to (VPW+1) lines.

In STN mode, these bits should be set to 0.

- **VHDLY: Vertical to horizontal delay**

In TFT mode, this is the delay between LCDVSYNC rising or falling edge and LCDHSYNC rising edge. Delay is (VHDLY+1) LCDDOTCK cycles.

In STN mode, these bits should be set to 0.

## 39.11.13 LCD Timing Configuration Register 2

**Name:** LCDTIM2

**Access:** Read/Write

**Reset value:** 0x00000000

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

- **HBP: Horizontal Back Porch**

Number of idle LCDDOTCK cycles at the beginning of the line. Idle period is (HBP+1) LCDDOTCK cycles.

- **HPW: Horizontal synchronization pulse width**

Width of the LCDHSYNC pulse, given in LCDDOTCK cycles. Width is (HPW+1) LCDDOTCK cycles.

- **HFP: Horizontal Front Porch**

Number of idle LCDDOTCK cycles at the end of the line. Idle period is (HFP+1) LCDDOTCK cycles.

### 39.11.14 LCD Frame Configuration Register

**Name:** LCDFRMCFG

**Access:** Read/Write

**Reset value:** 0x00000000

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

- **LINEVAL: Vertical size of LCD module**

In single scan mode: vertical size of LCD Module, in pixels, minus 1

In dual scan mode: vertical display size of each LCD panel, in pixels, minus 1

- **LINESIZE: Horizontal size of LCD module, in pixels, minus 1**



## 39.11.15 LCD FIFO Register

**Name:** LCDFIFO

**Access:** 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
FIFOTH							
7	6	5	4	3	2	1	0
FIFOTH							

- **FIFOTH: FIFO Threshold**

Must be programmed with:

$$\text{FIFOTH} = 2048 - (2 \times \text{DMA\_BURST\_LENGTH} + 3)$$

where:

- 2048 is the effective size of the FIFO. It is the total FIFO memory size in single scan mode and half that size in dual scan mode.
- DMA\_burst\_length is the burst length of the transfers made by the DMA. Refer to [“BRSTLN: Burst Length” on page 696](#).



### 39.11.16 Dithering Pattern DP1\_2 Register

Name: DP1\_2

Access: Read/Write

Reset value: 0xA5

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
DP1_2							

- DP1\_2: Pattern value for 1/2 duty cycle

### 39.11.17 Dithering Pattern DP4\_7 Register

Name: DP4\_7

Access: Read/Write

Reset value: 0x5AF0FA5

31	30	29	28	27	26	25	24
–	–	–	–	DP4_7			
23	22	21	20	19	18	17	16
DP4_7							
15	14	13	12	11	10	9	8
DP4_7							
7	6	5	4	3	2	1	0
DP4_7							

- DP4\_7: Pattern value for 4/7 duty cycle

### 39.11.18 Dithering Pattern DP3\_5 Register

Name: DP3\_5

Access: Read/Write

Reset value: 0xA5A5F

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	DP3_5			
15	14	13	12	11	10	9	8
DP3_5							
7	6	5	4	3	2	1	0
DP3_5							

- DP3\_5: Pattern value for 3/5 duty cycle

## 39.11.19 Dithering Pattern DP2\_3 Register

**Name:** DP2\_3: Dithering Pattern DP2\_3 Register

**Access:** Read/Write

**Reset value:** 0xA5F

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	DP2_3			
7	6	5	4	3	2	1	0
DP2_3							

- **DP2\_3:** Pattern value for 2/3 duty cycle

## 39.11.20 Dithering Pattern DP5\_7 Register

**Name:** DP5\_7:

**Access:** Read/Write

**Reset value:** 0xFAF5FA5

31	30	29	28	27	26	25	24
–	–	–	–	DP5_7			
23	22	21	20	19	18	17	16
DP5_7							
15	14	13	12	11	10	9	8
DP5_7							
7	6	5	4	3	2	1	0
DP5_7							

- **DP5\_7:** Pattern value for 5/7 duty cycle

## 39.11.21 Dithering Pattern DP3\_4 Register

**Name:** DP3\_4

**Access:** Read/Write

**Reset value:** 0xFAF5

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
DP3_4							
7	6	5	4	3	2	1	0
DP3_4							

- **DP3\_4:** Pattern value for 3/4 duty cycle



### 39.11.22 Dithering Pattern DP4\_5 Register

Name: DP4\_5

Access: Read/Write

Reset value: 0xFAF5F

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	DP4_5			
15	14	13	12	11	10	9	8
DP4_5							
7	6	5	4	3	2	1	0
DP4_5							

- DP4\_5: Pattern value for 4/5 duty cycle

### 39.11.23 Dithering Pattern DP6\_7 Register

Name: DP6\_7

Access: Read/Write

Reset value: 0xF5FFAFF

31	30	29	28	27	26	25	24
–	–	–	–	DP6_7			
23	22	21	20	19	18	17	16
DP6_7							
15	14	13	12	11	10	9	8
DP6_7							
7	6	5	4	3	2	1	0
DP6_7							

- DP6\_7: Pattern value for 6/7 duty cycle

## 39.11.24 Power Control Register

**Name:** PWRCON

**Access:** Read/Write

**Reset value:** 0x0000000e

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

- **LCD\_PWR: LCD module power control**

0 = lcd\_pwr signal is low, other lcd\_\* signals are low.

0->1 = lcd\_\* signals activated, lcd\_pwr is set high with the delay of GUARD\_TIME frame periods.

1 = lcd\_pwr signal is high, other lcd\_\* signals are active.

1->0 = lcd\_pwr signal is low, other lcd\_\* signals are active, but are set low after GUARD\_TIME frame periods.

- **GUARD\_TIME**

Delay in frame periods between applying control signals to the LCD module and setting LCD\_PWR high, and between setting LCD\_PWR low and removing control signals from LCD module

- **LCD\_BUSY**

Read-only field. If 1, it indicates that the LCD is busy (active and displaying data, in power on sequence or in power off sequence).

## 39.11.25 Contrast Control Register

Name: CONTRAST\_CTR

Access: 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
–	–	–	–	ENA	POL	PS	

- **PS**

This 2-bit value selects the configuration of a counter prescaler. The meaning of each combination is as follows:

PS		
0	0	The counter advances at a rate of $f_{\text{COUNTER}} = f_{\text{LCDC\_CLOCK}}$ .
0	1	The counter advances at a rate of $f_{\text{COUNTER}} = f_{\text{LCDC\_CLOCK}}/2$ .
1	0	The counter advances at a rate of $f_{\text{COUNTER}} = f_{\text{LCDC\_CLOCK}}/4$ .
1	1	The counter advances at a rate of $f_{\text{COUNTER}} = f_{\text{LCDC\_CLOCK}}/8$ .

- **POL**

This bit defines the polarity of the output. If 1, the output pulses are high level (the output will be high whenever the value in the counter is less than the value in the compare register CONTRAST\_VAL). If 0, the output pulses are low level.

- **ENA**

When 1, this bit enables the operation of the PWM generator. When 0, the PWM counter is stopped.

## 39.11.26 Contrast Value Register

**Name:** CONSTRAST\_VAL

**Access:** 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
CVAL							

- **CVAL**

PWM compare value. Used to adjust the analog value obtained after an external filter to control the contrast of the display.

## 39.11.27 LCD Interrupt Enable Register

Name: LCD\_IER

Access: Write-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
–	MERIE	OWRIE	UFLWIE	–	EOFIE	LSTLNIE	LNIE

- **LNIE: Line interrupt enable**

0: No effect

1: Enable each line interrupt

- **LSTLNIE: Last line interrupt enable**

0: No effect

1: Enable last line interrupt

- **EOFIE: DMA End of frame interrupt enable**

0: No effect

1: Enable End Of Frame interrupt

- **UFLWIE: FIFO underflow interrupt enable**

0: No effect

1: Enable FIFO underflow interrupt

- **OWRIE: FIFO overwrite interrupt enable**

0: No effect

1: Enable FIFO overwrite interrupt

- **MERIE: DMA memory error interrupt enable**

0: No effect

1: Enable DMA memory error interrupt



## 39.11.28 LCD Interrupt Disable Register

**Name:** LCD\_IDR

**Access:** Write-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
–	MERID	OWRID	UFLWID	–	EOFID	LSTLNID	LNID

- **LNID: Line interrupt disable**

0: No effect

1: Disable each line interrupt

- **LSTLNID: Last line interrupt disable**

0: No effect

1: Disable last line interrupt

- **EOFID: DMA End of frame interrupt disable**

0: No effect

1: Disable End Of Frame interrupt

- **UFLWID: FIFO underflow interrupt disable**

0: No effect

1: Disable FIFO underflow interrupt

- **OWRID: FIFO overwrite interrupt disable**

0: No effect

1: Disable FIFO overwrite interrupt

- **MERID: DMA Memory error interrupt disable**

0: No effect

1: Disable DMA Memory error interrupt

## 39.11.29 LCD Interrupt Mask Register

**Name:** LCD\_IMR

**Access:** 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
–	MERIM	OWRIM	UFLWIM	–	EOFIM	LSTLNIM	LNIM

- **LNIM: Line interrupt mask**

0: Line Interrupt disabled

1: Line interrupt enabled

- **LSTLNIM: Last line interrupt mask**

0: Last Line Interrupt disabled

1: Last Line Interrupt enabled

- **EOFIM: DMA End of frame interrupt mask**

0: End Of Frame interrupt disabled

1: End Of Frame interrupt enabled

- **UFLWIM: FIFO underflow interrupt mask**

0: FIFO underflow interrupt disabled

1: FIFO underflow interrupt enabled

- **OWRIM: FIFO overwrite interrupt mask**

0: FIFO overwrite interrupt disabled

1: FIFO overwrite interrupt enabled

- **MERIM: DMA Memory error interrupt mask**

0: DMA Memory error interrupt disabled

1: DMA Memory error interrupt enabled

## 39.11.30 LCD Interrupt Status Register

**Name:** LCD\_ISR

**Access:** 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
–	MERIS	OWRIS	UFLWIS	–	EOFIS	LSTLNIS	LNIS

- **LNIS: Line interrupt status**

0: Line Interrupt not active

1: Line Interrupt active

- **LSTLNIS: Last line interrupt status**

0: Last Line Interrupt not active

1: Last Line Interrupt active

- **EOFIS: DMA End of frame interrupt status**

0: End Of Frame interrupt not active

1: End Of Frame interrupt active

- **UFLWIS: FIFO underflow interrupt status**

0: FIFO underflow interrupt not active

1: FIFO underflow interrupt active

- **OWRIS: FIFO overwrite interrupt status**

0: FIFO overwrite interrupt not active

1: FIFO overwrite interrupt active

- **MERIS: DMA Memory error interrupt status**

0: DMA Memory error interrupt not active

1: DMA Memory error interrupt active

## 39.11.31 LCD Interrupt Clear Register

**Name:** LCD\_ICR

**Access:** Write-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
–	MERIC	OWRIC	UFLWIC	–	EOFIC	LSTLNIC	LNIC

- **LNIC: Line interrupt clear**

0: No effect

1: Clear each line interrupt

- **LSTLNIC: Last line interrupt clear**

0: No effect

1: Clear Last line Interrupt

- **EOFIC: DMA End of frame interrupt clear**

0: No effect

1: Clear End Of Frame interrupt

- **UFLWIC: FIFO underflow interrupt clear**

0: No effect

1: Clear FIFO underflow interrupt

- **OWRIC: FIFO overwrite interrupt clear**

0: No effect

1: Clear FIFO overwrite interrupt

- **MERIC: DMA Memory error interrupt clear**

0: No effect

1: Clear DMA Memory error interrupt

## 39.11.32 LCD Interrupt Test Register

**Name:** LCD\_ITR

**Access:** Write-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
–	MERIT	OWRIT	UFLWIT	–	EOFIT	LSTLNIT	LNIT

- **LNIT: Line interrupt test**

0: No effect

1: Set each line interrupt

- **LSTLNIT: Last line interrupt test**

0: No effect

1: Set Last line interrupt

- **EOFIT: DMA End of frame interrupt test**

0: No effect

1: Set End Of Frame interrupt

- **UFLWIT: FIFO underflow interrupt test**

0: No effect

1: Set FIFO underflow interrupt

- **OWRIT: FIFO overwrite interrupt test**

0: No effect

1: Set FIFO overwrite interrupt

- **MERIT: DMA Memory error interrupt test**

0: No effect

1: Set DMA Memory error interrupt

## 39.11.33 LCD Interrupt Raw Status Register

**Name:** LCD\_IRR

**Access:** Write-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
–	MERIR	OWRIR	UFLWIR	–	EOFIR	LSTLNIR	LNIR

- **LNIR: Line interrupt raw status**

0: No effect

1: Line interrupt condition present

- **LSTLNIR: Last line interrupt raw status**

0: No effect

1: Last line Interrupt condition present

- **EOFIR: DMA End of frame interrupt raw status**

0: No effect

1: End Of Frame interrupt condition present

- **UFLWIR: FIFO underflow interrupt raw status**

0: No effect

1: FIFO underflow interrupt condition present

- **OWRIR: FIFO overwrite interrupt raw status**

0: No effect

1: FIFO overwrite interrupt condition present

- **MERIR: DMA Memory error interrupt raw status**

0: No effect

1: DMA Memory error interrupt condition present

## 40. AC'97 Controller (AC'97C)

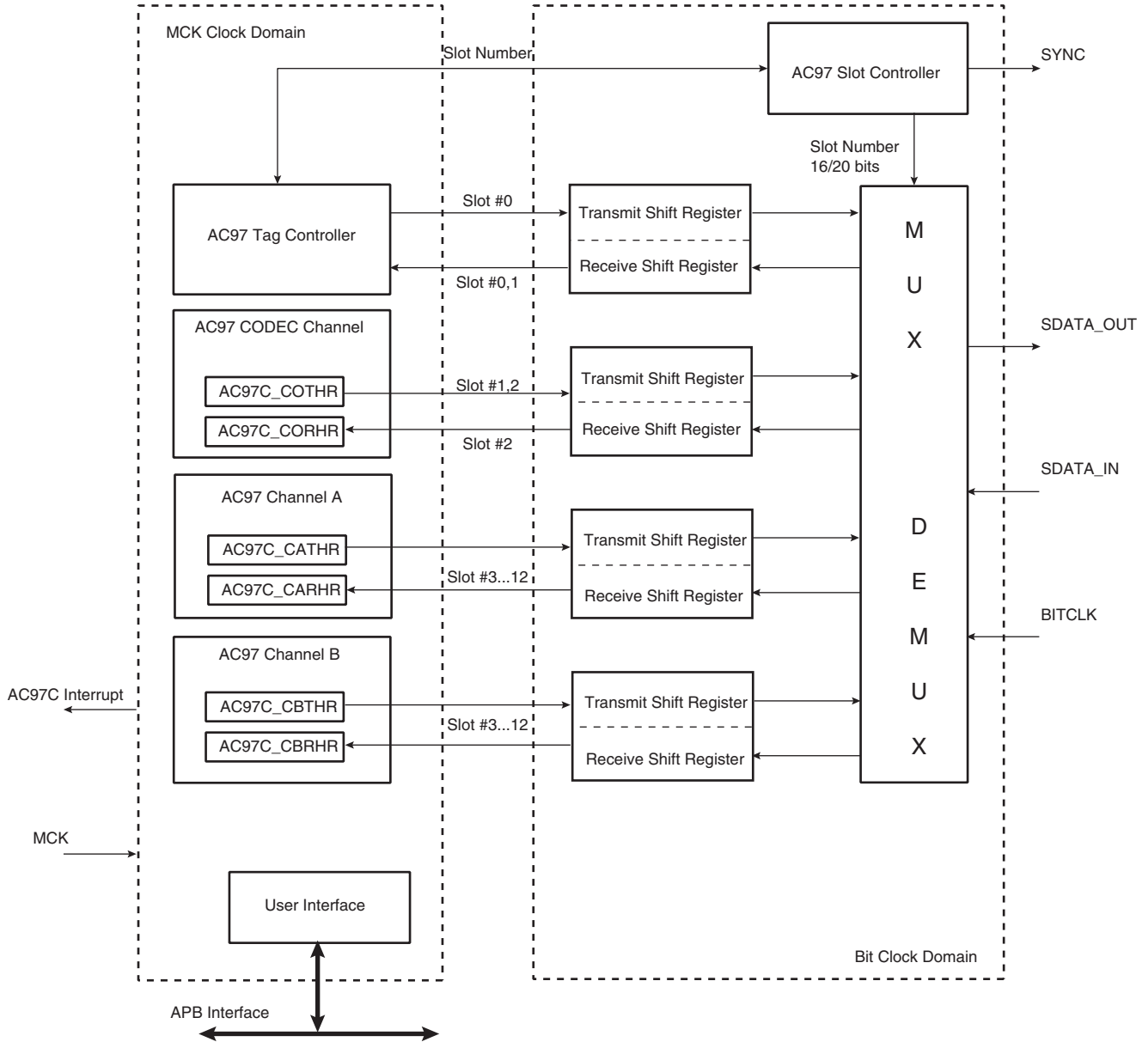
### 40.1 Description

The AC'97 Controller is the hardware implementation of the AC'97 digital controller (DC'97) compliant with AC'97 Component Specification 2.2. The AC'97 Controller communicates with an audio codec (AC'97) or a modem codec (MC'97) via the AC-link digital serial interface. All digital audio, modem and handset data streams, as well as control (command/status) informations are transferred in accordance to the AC-link protocol.

The AC'97 Controller features a Peripheral DMA Controller (PDC) for audio streaming transfers. It also supports variable sampling rate and four Pulse Code Modulation (PCM) sample resolutions of 10, 16, 18 and 20 bits.

## 40.2 Block Diagram

Figure 40-1. Functional Block Diagram





## 40.3 Pin Name List

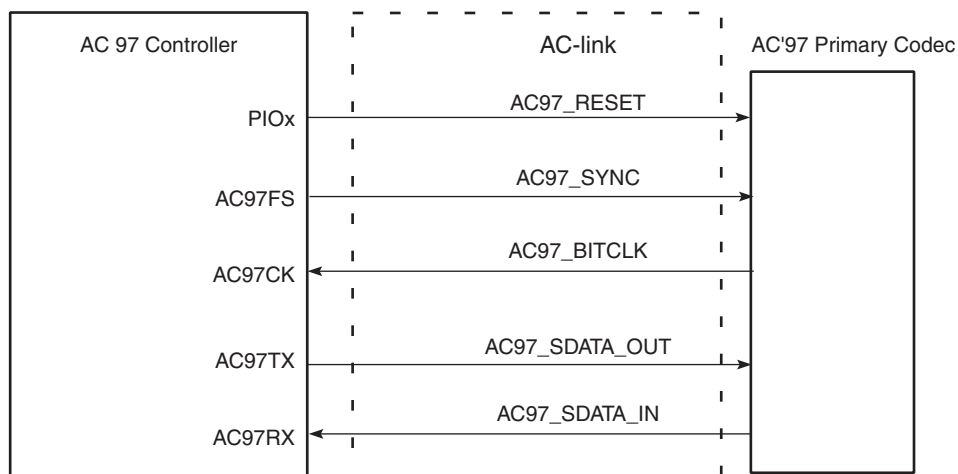
**Table 40-1.** I/O Lines Description

Pin Name	Pin Description	Type
AC97CK	12.288-MHz bit-rate clock	Input
AC97RX	Receiver Data (Referred as SDATA_IN in AC-link spec)	Input
AC97FS	48-KHz frame indicator and synchronizer	Output
AC97TX	Transmitter Data (Referred as SDATA_OUT in AC-link spec)	Output

The AC'97 reset signal provided to the primary codec can be generated by a PIO.

## 40.4 Application Block Diagram

**Figure 40-2.** Application Block diagram



## 40.5 Product Dependencies

### 40.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines.

Before using the AC'97 Controller receiver, the PIO controller must be configured in order for the AC97C receiver I/O lines to be in AC'97 Controller peripheral mode.

Before using the AC'97 Controller transmitter, the PIO controller must be configured in order for the AC97C transmitter I/O lines to be in AC'97 Controller peripheral mode.

### 40.5.2 Power Management

The AC'97 Controller is not continuously clocked. Its interface may be clocked through the Power Management Controller (PMC), therefore the programmer must first configure the PMC to enable the AC'97 Controller clock.

The AC'97 Controller has two clock domains. The first one is supplied by PMC and is equal to MCK. The second one is AC97CK which is sent by the AC97 Codec (Bit clock).

Signals that cross the two clock domains are re-synchronized. MCK clock frequency must be higher than the AC97CK (Bit Clock) clock frequency.

### 40.5.3 Interrupt

The AC'97 Controller interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling interrupts requires programming the AIC before configuring the AC97C.

All AC'97 Controller interrupts can be enabled/disabled by writing to the AC'97 Controller Interrupt Enable/Disable Registers. Each pending and unmasked AC'97 Controller interrupt will assert the interrupt line. The AC'97 Controller interrupt service routine can get the interrupt source in two steps:

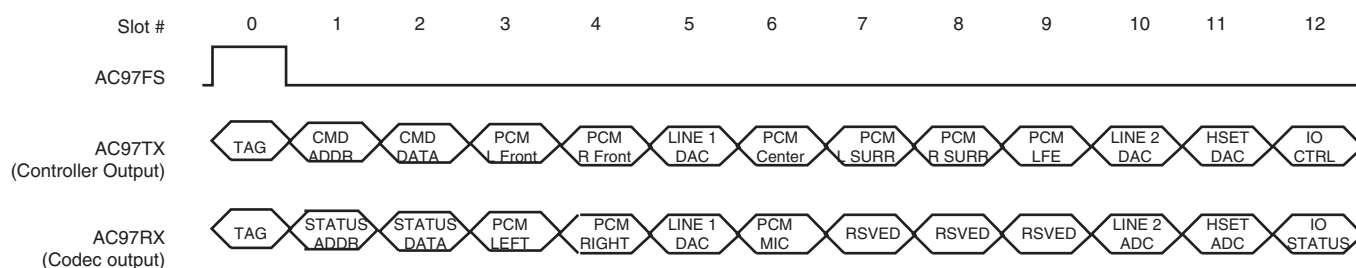
- Reading and ANDing AC'97 Controller Interrupt Mask Register (AC97C\_IMR) and AC'97 Controller Status Register (AC97C\_SR).
- Reading AC'97 Controller Channel x Status Register (AC97C\_CxSR).

## 40.6 Functional Description

### 40.6.1 Protocol overview

AC-link protocol is a bidirectional, fixed clock rate, serial digital stream. AC-link handles multiple input and output Pulse Code Modulation PCM audio streams, as well as control register accesses employing a Time Division Multiplexed (TDM) scheme that divides each audio frame in 12 outgoing and 12 incoming 20-bit wide data slots.

**Figure 40-3.** Bidirectional AC-link Frame with Slot Assignment



**Table 40-2.** AC-link Output Slots Transmitted from the AC'97C Controller

Slot #	Pin Description
0	TAG
1	Command Address Port
2	Command Data Port
3,4	PCM playback Left/Right Channel
5	Modem Line 1 Output Channel
6, 7, 8	PCM Center/Left Surround/Right Surround
9	PCM LFE DAC
10	Modem Line 2 Output Channel
11	Modem Handset Output Channel
12	Modem GPIO Control Channel

**Table 40-3.** AC-link Input Slots Transmitted from the AC'97C Controller

Slot #	Pin Description
0	TAG
1	Status Address Port
2	Status Data Port
3,4	PCM playback Left/Right Channel
5	Modem Line 1 ADC
6	Dedicated Microphone ADC
7, 8, 9	Vendor Reserved

**Table 40-3.** AC-link Input Slots Transmitted from the AC'97C Controller

Slot #	Pin Description
10	Modem Line 2 ADC
11	Modem Handset Input ADC
12	Modem IO Status

#### 40.6.1.1 Slot Description

##### Tag Slot

The tag slot, or slot 0, is a 16-bit wide slot that always goes at the beginning of an outgoing or incoming frame. Within tag slot, the first bit is a global bit that flags the entire frame validity. The next 12 bit positions sampled by the AC'97 Controller indicate which of the corresponding 12 time slots contain valid data. The slot's last two bits (combined) called Codec ID, are used to distinguish primary and secondary codec.

The 16-bit wide tag slot of the output frame is automatically generated by the AC'97 Controller according to the transmit request of each channel and to the SLOTRREQ from the previous input frame, sent by the AC'97 Codec, in Variable Sample Rate mode.

##### Codec Slot 1

The command/status slot is a 20-bit wide slot used to control features, and monitors status for AC'97 Codec functions.

The control interface architecture supports up to sixty-four 16-bit wide read/write registers. Only the even registers are currently defined and addressed.

Slot 1's bitmap is the following:

- Bit 19 is for read/write command, 1= read, 0 = write.
- Bits [18:12] are for control register index.
- Bits [11:0] are reserved.

##### Codec Slot 2

Slot 2 is a 20-bit wide slot used to carry 16-bit wide AC97 Codec control register data. If the current command port operation is a read, the entire slot time is stuffed with zeros. Its bitmap is the following:

- Bits [19:4] are the control register data
- Bits [3:0] are reserved and stuffed with zeros.

##### Data Slots [3:12]

Slots [3:12] are 20-bit wide data slots, they usually carry audio PCM or/and modem I/O data.

## 40.6.2 AC'97 Controller Channel Organization

The AC'97 Controller features a Codec channel and 2 logical channels; Channel A and Channel B.

The Codec channel controls AC'97 Codec registers, it enables write and read configuration values in order to bring the AC97 Codec to an operating state. The Codec channel always runs slot 1 and slot 2 exclusively, in both input and output directions.

Channel A and Channel B transfer data to/from AC97 codec. All audio samples and modem data must transit by these two channels. However, Channel A is connected to PDC channels thus making it suitable for audio streaming applications.

Each slot of the input or the output frame that belongs to this range [3 to 12] can be operated by either Channel A or Channel B. The slot to channel assignment is configured by two registers:

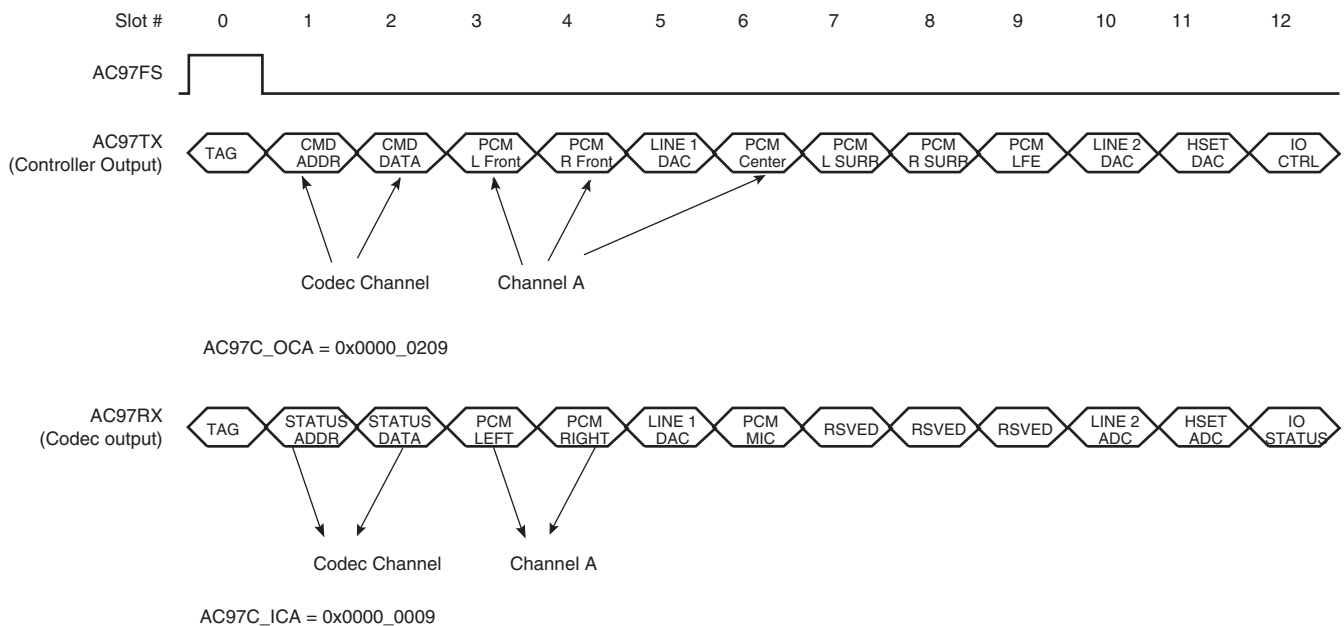
- AC'97 Controller Input Channel Assignment Register (AC97C\_ICA)
- AC'97 Controller Output Channel Assignment Register (AC97C\_OCA)

The AC'97 Controller Input Channel Assignment Register (AC97C\_ICA) configures the input slot to channel assignment. The AC'97 Controller Output Channel Assignment Register (AC97C\_OCA) configures the output slot to channel assignment.

A slot can be left unassigned to a channel by the AC'97 Controller. Slots 0, 1, and 2 cannot be assigned to Channel A or to Channel B through the AC97C\_OCA and AC97C\_ICA Registers.

The width of sample data, that transit via Channel A and Channel B varies and can take one of these values; 10, 16, 18 or 20 bits.

**Figure 40-4.** Logical Channel Assignment



#### 40.6.2.1 AC97 Controller Setup

The following operations must be performed in order to bring the AC'97 Controller into an operating state:

1. Enable the AC97 Controller clock in the PMC controller.
2. Turn on AC97 function by enabling the ENA bit in AC97 Controller Mode Register (AC97C\_MR).
3. Configure the input channel assignment by controlling the AC'97 Controller Input Assignment Register (AC97C\_ICA).
4. Configure the output channel assignment by controlling the AC'97 Controller Input Assignment Register (AC97C\_OCA).
5. Configure sample width for Channel A and Channel B by writing the SIZE bit field in AC97C Channel A Mode Register (AC97C\_CAMR) and AC97C Channel B Mode Register (AC97C\_CBMR). The application can write 10, 16, 18, or 20-bit wide PCM samples through the AC'97 interface and they will be transferred into 20-bit wide slots.
6. Configure data Endianness for Channel A and Channel B by writing CEM bit field in AC97C\_CAMR and AC97C\_CBMR registers. Data on the AC-link are shifted MSB first. The application can write little- or big-endian data to the AC'97 Controller interface.
7. Configure the PIO controller to drive the RESET signal of the external Codec. The RESET signal must fulfill external AC97 Codec timing requirements.
8. Enable Channel A and/or Channel B by writing CEN bit field in AC97C\_CAMR and AC97C\_CBMR registers.

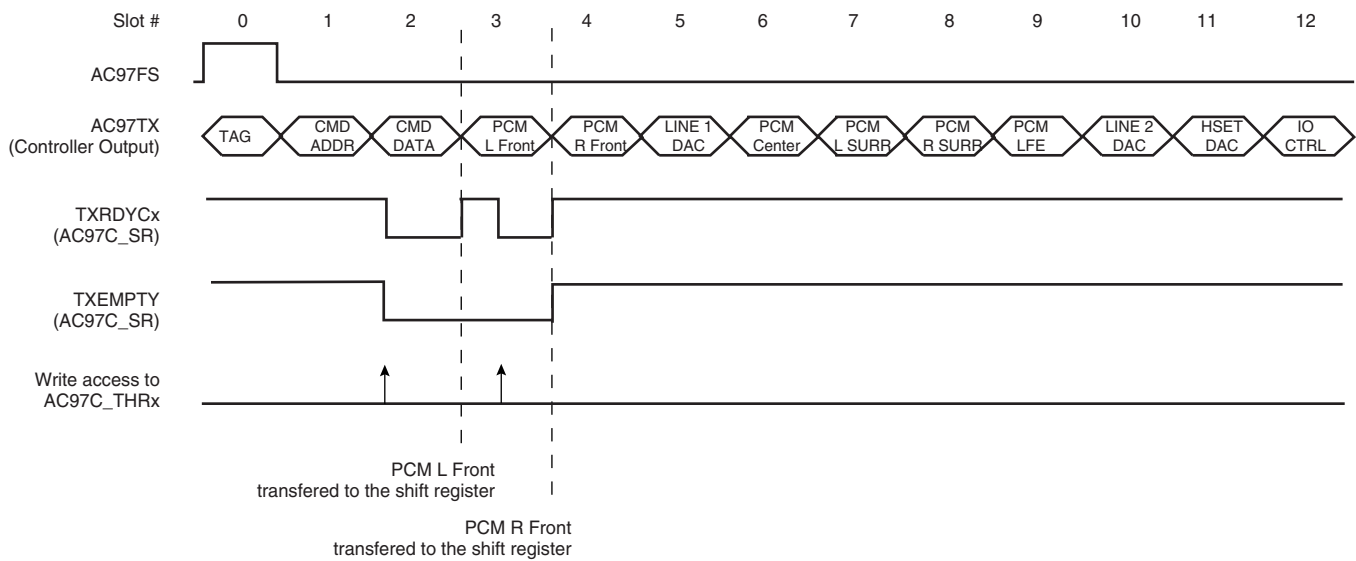
#### 40.6.2.2 Transmit Operation

The application must perform the following steps in order to send data via a channel to the AC97 Codec:

- Check if previous data has been sent by polling TXRDY flag in the AC97C Channel x Status Register (AC97C\_CxSR). x being one of the 2 channels.
- Write data to the AC'97 Controller Channel x Transmit Holding Register (AC97C\_CxTHR).

Once data has been transferred to the Channel x Shift Register, the TXRDY flag is automatically set by the AC'97 Controller which allows the application to start a new write action. The application can also wait for an interrupt notice associated with TXRDY in order to send data. The interrupt remains active until TXRDY flag is cleared..

**Figure 40-5. Audio Transfer (PCM L Front, PCM R Front) on Channel x**



The TXEMPTY flag in the AC'97 Controller Channel x Status Register (AC97C\_CxSR) is set when all requested transmissions for a channel have been shifted on the AC-link. The application can either poll TXEMPTY flag in AC97C\_CxSR or wait for an interrupt notice associated with the same flag.

In most cases, the AC'97 Controller is embedded in chips that target audio player devices. In such cases, the AC'97 Controller is exposed to heavy audio transfers. Using the polling technique increases processor overhead and may fail to keep the required pace under an operating system. In order to avoid these polling drawbacks, the application can perform audio streams by using PDC connected to channel A, which reduces processor overhead and increases performance especially under an operating system.

The PDC transmit counter values must be equal to the number of PCM samples to be transmitted, each sample goes in one slot.

### 40.6.2.3 AC'97 Output Frame

The AC'97 Controller outputs a thirteen-slot frame on the AC-Link. The first slot (tag slot or slot 0) flags the validity of the entire frame and the validity of each slot; whether a slot carries valid data or not. Slots 1 and 2 are used if the application performs control and status monitoring actions on AC97 Codec control/status registers. Slots [3:12] are used according to the content of the AC'97 Controller Output Channel Assignment Register (AC97C\_OCA). If the application performs many transmit requests on a channel, some of the slots associated to this channel or all of them will carry valid data.

### 40.6.2.4 Receive Operation

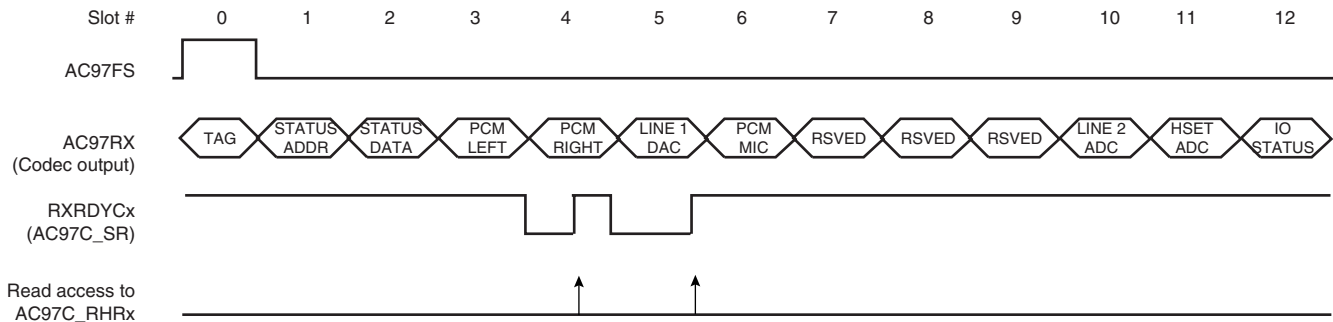
The AC'97 Controller can also receive data from AC'97 Codec. Data is received in the channel's shift register and then transferred to the AC'97 Controller Channel x Read Holding Register. To read the newly received data, the application must perform the following steps:

- Poll RXRDY flag in AC'97 Controller Channel x Status Register (AC97C\_CxSR). x being one of the 2 channels.
- Read data from AC'97 Controller Channel x Read Holding Register.

The application can also wait for an interrupt notice in order to read data from AC97C\_CxRHR. The interrupt remains active until RXRDY is cleared by reading AC97C\_CxSR.

The RXRDY flag in AC97C\_CxSR is set automatically when data is received in the Channel x shift register. Data is then shifted to AC97C\_CxRHR.

**Figure 40-6.** Audio Transfer (PCM L Front, PCM R Front) on Channel x



If the previously received data has not been read by the application, the new data overwrites the data already waiting in AC97C\_CxRHR, therefore the OVRUN flag in AC97C\_CxSR is raised. The application can either poll the OVRUN flag in AC97C\_CxSR or wait for an interrupt notice. The interrupt remains active until the OVRUN flag in AC97C\_CxSR is set.

The AC'97 Controller can also be used in sound recording devices in association with an AC97 Codec. The AC'97 Controller may also be exposed to heavy PCM transfers. The application can use the PDC connected to channel A in order to reduce processor overhead and increase performance especially under an operating system.

The PDC receive counter values must be equal to the number of PCM samples to be received, each sample goes in one slot.

#### 40.6.2.5 AC'97 Input Frame

The AC'97 Controller receives a thirteen slot frame on the AC-Link sent by the AC97 Codec. The first slot (tag slot or slot 0) flags the validity of the entire frame and the validity of each slot; whether a slot carries valid data or not. Slots 1 and 2 are used if the application requires status informations from AC97 Codec. Slots [3:12] are used according to AC'97 Controller Output Channel Assignment Register (AC97C\_ICA) content. The AC'97 Controller will not receive any data from any slot if AC97C\_ICA is not assigned to a channel in input.

#### 40.6.2.6 Configuring and Using Interrupts

Instead of polling flags in AC'97 Controller Global Status Register (AC97C\_SR) and in AC'97 Controller Channel x Status Register (AC97C\_CxSR), the application can wait for an interrupt notice. The following steps show how to configure and use interrupts correctly:

- Set the interruptible flag in AC'97 Controller Channel x Mode Register (AC97C\_CxMR).
- Set the interruptible event and channel event in AC'97 Controller Interrupt Enable Register (AC97C\_IER).

The interrupt handler must read both AC'97 Controller Global Status Register (AC97C\_SR) and AC'97 Controller Interrupt Mask Register (AC97C\_IMR) and AND them to get the real interrupt source. Furthermore, to get which event was activated, the interrupt handler has to read AC'97 Controller Channel x Status Register (AC97C\_CxSR), x being the channel whose event triggers the interrupt.



The application can disable event interrupts by writing in AC'97 Controller Interrupt Disable Register (AC97C\_IDR). The AC'97 Controller Interrupt Mask Register (AC97C\_IMR) shows which event can trigger an interrupt and which one cannot.

## 40.6.2.7 Endianness

Endianness can be managed automatically for each channel, except for the Codec channel, by writing to Channel Endianness Mode (CEM) in AC97C\_CxMR. This enables transferring data on AC-link in Big Endian format without any additional operation.

### To Transmit a Word Stored in Big Endian Format on AC-link

Word to be written in AC'97 Controller Channel x Transmit Holding Register (AC97C\_CxTHR) (as it is stored in memory or microprocessor register).

31	24	23	16	15	8	7	0
Byte0[7:0]		Byte1[7:0]		Byte2[7:0]		Byte3[7:0]	

Word stored in Channel x Transmit Holding Register (AC97C\_CxTHR) (data to transmit).

31	24	23	20	19	16	15	8	7	0
-		-		Byte2[3:0]		Byte1[7:0]		Byte0[7:0]	

Data transmitted on appropriate slot: data[19:0] = {Byte2[3:0], Byte1[7:0], Byte0[7:0]}.

### To Transmit A Halfword Stored in Big Indian Format on AC-link

Halfword to be written in AC'97 Controller Channel x Transmit Holding Register (AC97C\_CxTHR).

31	24	23	16	15	8	7	0
-		-		Byte0[7:0]		Byte1[7:0]	

Halfword stored in AC'97 Controller Channel x Transmit Holding Register (AC97C\_CxTHR) (data to transmit).

31	24	23	16	15	8	7	0
-		-		Byte1[7:0]		Byte0[7:0]	

Data emitted on related slot: data[19:0] = {0x0, Byte1[7:0], Byte0[7:0]}.

### To Transmit a 10-bit Sample Stored in Big Endian Format on AC-link

Halfword to be written in AC'97 Controller Channel x Transmit Holding Register (AC97C\_CxTHR).

31	24	23	16	15	8	7	0
-		-		Byte0[7:0]		{0x00, Byte1[1:0]}	

Halfword stored in AC'97 Controller Channel x Transmit Holding Register (AC97C\_CxTHR) (data to transmit).

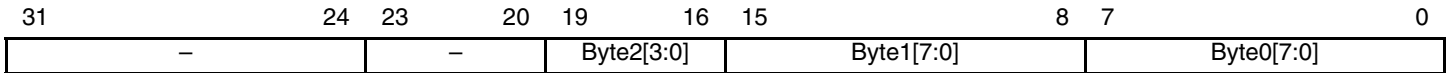
31	24	23	16	15	10	9	8	7	0
-		-		-		Byte1 [1:0]		Byte0[7:0]	

Data emitted on related slot: data[19:0] = {0x000, Byte1[1:0], Byte0[7:0]}.

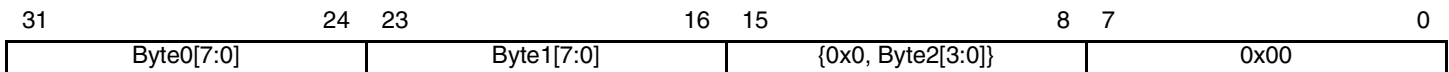
### To Receive Word transfers

Data received on appropriate slot: data[19:0] = {Byte2[3:0], Byte1[7:0], Byte0[7:0]}.

Word stored in AC'97 Controller Channel x Receive Holding Register (AC97C\_CxRHR) (Received Data).



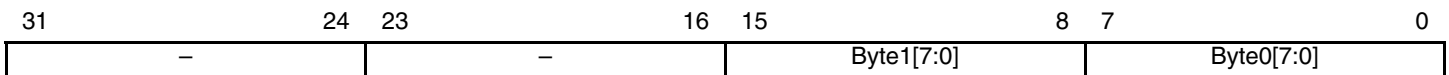
Data is read from AC'97 Controller Channel x Receive Holding Register (AC97C\_CxRHR) when Channel x data size is greater than 16 bits and when big-endian mode is enabled (data written to memory).



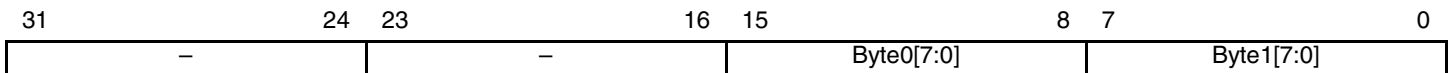
### To Receive Halfword Transfers

Data received on appropriate slot: data[19:0] = {0x0, Byte1[7:0], Byte0[7:0]}.

Halfword stored in AC'97 Controller Channel x Receive Holding Register (AC97C\_CxRHR) (Received Data).

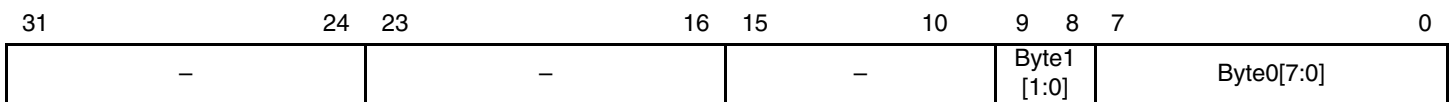


Data is read from AC'97 Controller Channel x Receive Holding Register (AC97C\_CxRHR) when data size is equal to 16 bits and when big-endian mode is enabled.

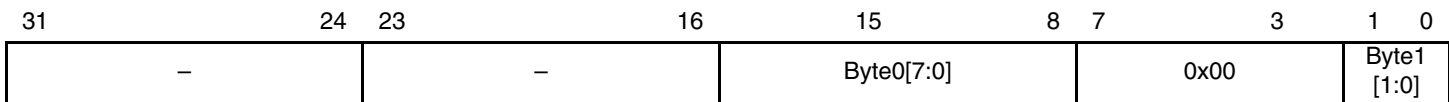


### To Receive 10-bit Samples

Data received on appropriate slot: data[19:0] = {0x000, Byte1[1:0], Byte0[7:0]}. Halfword stored in AC'97 Controller Channel x Receive Holding Register (AC97C\_CxRHR) (Received Data)



Data read from AC'97 Controller Channel x Receive Holding Register (AC97C\_CxRHR) when data size is equal to 10 bits and when big-endian mode is enabled.



## 40.6.3 Variable Sample Rate

The problem of variable sample rate can be summarized by a simple example. When passing a 44.1 kHz stream across the AC-link, for every 480 audio output frames that are sent across, 441 of them must contain valid sample data. The new AC'97 standard approach calls for the addition of "on-demand" slot request flags. The AC'97 Codec examines its sample rate control register, the state of its FIFOs, and the incoming SDATA\_OUT tag bits (slot 0) of each output frame and then determines which SLOTREQ bits to set active (low). These bits are passed from the AC97

Codec to the AC'97 Controller in slot 1/SLOTREQ in every audio input frame. Each time the AC'97 controller sees one or more of the newly defined slot request flags set active (low) in a given audio input frame, it must pass along the next PCM sample for the corresponding slot(s) in the AC-link output frame that immediately follows.

The variable Sample Rate mode is enabled by performing the following steps:

- Setting the VRA bit in the AC'97 Controller Mode Register (AC97C\_MR).
- Enable Variable Rate mode in the AC'97 Codec by performing a transfer on the Codec channel.

Slot 1 of the input frame is automatically interpreted as SLOTREQ signaling bits. The AC'97 Controller will automatically fill the active slots according to both SLOTREQ and AC97C\_OCA register in the next transmitted frame.

## 40.6.4 Power Management

### 40.6.4.1 Powering Down the AC-Link

The AC97 Codecs can be placed in low power mode. The application can bring AC97 Codec to a power down state by performing sequential writes to AC97 Codec powerdown register. Both the bit clock (clock delivered by AC97 Codec, AC97CK) and the input line (AC97RX) are held at a logic low voltage level. This puts AC97 Codec in power down state while all its registers are still holding current values. Without the bit clock, the AC-link is completely in a power down state.

The AC'97 Controller should not attempt to play or capture audio data until it has awakened AC97 Codec.

To set the AC'97 Codec in low power mode, the PR4 bit in the AC'97 Codec powerdown register (Codec address 0x26) must be set to 1. Then the primary Codec drives both AC97CK and AC97RX to a low logic voltage level.

The following operations must be done to put AC97 Codec in low power mode:

- Disable Channel A clearing CEN in the AC97C\_CAMR register.
- Disable Channel B clearing CEN field in the AC97C\_CBMR register.
- Write 0x2680 value in the AC97C\_COTHR register.
- Poll the TXEMPTY flag in AC97C\_CxSR registers for the 2 channels.

At this point AC97 Codec is in low power mode.

### 40.6.4.2 Waking up the AC-link

There are two methods to bring the AC-link out of low power mode. Regardless of the method, it is always the AC97 Controller that performs the wake-up.

#### *Wake-up Triggered by the AC'97 Controller*

The AC'97 Controller can wake up the AC97 Codec by issuing either a cold or a warm reset.

The AC'97 Controller can also wake up the AC97 Codec by asserting AC97FS signal, however this action should not be performed for a minimum period of four audio frames following the frame in which the powerdown was issued.

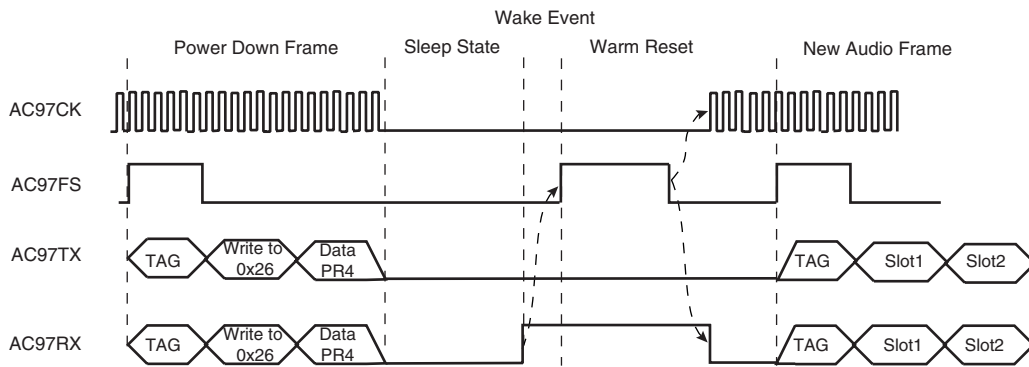
#### *Wake-up Triggered by the AC97 Codec*

This feature is implemented in AC97 modem codecs that need to report events such as Caller-ID and wake-up on ring.

The AC97 Codec can drive AC97RX signal from low to high level and holding it high until the controller issues either a cold or a warm reset. The AC97RX rising edge is asynchronously (regarding AC97FS) detected by the AC'97 Controller. If WKUP bit is enabled in AC97C\_IMR register, an interrupt is triggered that wakes up the AC'97 Controller which should then immediately issue a cold or a warm reset.

If the processor needs to be awakened by an external event, the AC97RX signal must be externally connected to the WAKEUP entry of the system controller.

**Figure 40-7.** AC'97 Power-Down/Up Sequence



#### 40.6.4.3 AC97 Codec Reset

There are three ways to reset an AC97 Codec.

##### Cold AC'97 Reset

A cold reset is generated by asserting the RESET signal low for the minimum specified time (depending on the AC97 Codec) and then by de-asserting RESET high. AC97CK and AC97FS is reactivated and all AC97 Codec registers are set to their default power-on values. Transfers on AC-link can resume.

The RESET signal will be controlled via a PIO line. This is how an application should perform a cold reset:

- Clear and set ENA flag in the AC97C\_MR register to reset the AC'97 Controller
- Clear PIO line output controlling the AC'97 RESET signal
- Wait for the minimum specified time
- Set PIO line output controlling the AC'97 RESET signal

AC97CK, the clock provided by AC97 Codec, is detected by the controller.

##### Warm AC'97 Reset

A warm reset reactivates the AC-link without altering AC97 Codec registers. A warm reset is signaled by driving AC97FX signal high for a minimum of 1us in the absence of AC97CK. In the absence of AC97CK, AC97FX is treated as an asynchronous (regarding AC97FX) input used to signal a warm reset to AC97 Codec.

This is the right way to perform a warm reset:

- Set WRST in the AC97C\_MR register.

- Wait for at least 1 us
- Clear WRST in the AC97C\_MR register.

The application can check that operations have resumed by checking SOF flag in the AC97C\_SR register or wait for an interrupt notice if SOF is enabled in AC97C\_IMR.

## 40.7 AC'97 Controller (AC97C) User Interface

**Table 40-4.** Register Mapping

Offset	Register	Register Name	Access	Reset
0x0-0x4	Reserved	–	–	–
0x8	Mode Register	AC97C_MR	Read/Write	0x0
0xC	Reserved	–	–	–
0x10	Input Channel Assignment Register	AC97C_ICA	Read/Write	0x0
0x14	Output Channel Assignment Register	AC97C_OCA	Read/Write	0x0
0x18-0x1C	Reserved	–	–	–
0x20	Channel A Receive Holding Register	AC97C_CARHR	Read	0x0
0x24	Channel A Transmit Holding Register	AC97C_CATHR	Write	–
0x28	Channel A Status Register	AC97C_CASR	Read	0x0
0x2C	Channel A Mode Register	AC97C_CAMR	Read/Write	0x0
0x30	Channel B Receive Holding Register	AC97C_CBRHR	Read	0x0
0x34	Channel B Transmit Holding Register	AC97C_CBTHR	Write	–
0x38	Channel B Status Register	AC97C_CBSR	Read	0x0
0x3C	Channel B Mode Register	AC97C_CBMR	Read/Write	0x0
0x40	Codec Receive Holding Register	AC97C_CORHR	Read	0x0
0x44	Codec Transmit Holding Register	AC'97C_COTHR	Write	–
0x48	Codec Status Register	AC'97C_COSR	Read	0x0
0x4C	Codec Mode Register	AC'97C_COMR	Read/Write	0x0
0x50	Status Register	AC97C_SR	Read	0x0
0x54	Interrupt Enable Register	AC97C_IER	Write	–
0x58	Interrupt Disable Register	AC97C_IDR	Write	–
0x5C	Interrupt Mask Register	AC97C_IMR	Read	0x0
0x60-0xFB	Reserved	–	–	–
0x100- 0x124	Reserved for Peripheral Data Controller (PDC), registers related to Channel A transfers	AC97C_CARPR, AC97C_CARCR, AC97C_CATPR, AC97C_CATCR, AC97C_CARNPR, AC97C_CARNCR, AC97C_CATNPR, AC97C_CATNCR, AC97C_CAPTCR, AC97C_CAPTSR	–	–

## 40.7.1 AC'97 Controller Mode Register

Name: AC97C\_MR

Access Type: Read-Write

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

- **VRA: Variable Rate (for Data Slots 3-12)**

0: Variable Rate is inactive. (48 KHz only)

1: Variable Rate is active.

- **WRST: Warm Reset**

0: Warm Reset is inactive.

1: Warm Reset is active.

- **ENA: AC'97 Controller Global Enable**

0: No effect. AC'97 function as well as access to other AC'97 Controller registers are disabled.

1: Activates the AC'97 function.

## 40.7.2 AC'97 Controller Input Channel Assignment Register

Register Name: AC97C\_ICA

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	CHID12			CHID11		
23	22	21	20	19	18	17	16
CHID10			CHID9			CHID8	
15	14	13	12	11	10	9	8
CHID8	CHID7			CHID6			CHID5
7	6	5	4	3	2	1	0
CHID5		CHID4			CHID3		

- **CHIDx: Channel ID for the input slot x**

CHIDx	Selected Receive Channel
0x0	None. No data will be received during this Slot x
0x1	Channel A data will be received during this slot time.
0x2	Channel B data will be received during this slot time



### 40.7.3 AC'97 Controller Output Channel Assignment Register

Register Name: AC97C\_OCA

Access Type: Read/Write

31	30	29	28	27	26	25	24
-		CHID12		CHID11			
23	22	21	20	19	18	17	16
CHID10			CHID9			CHID8	
15	14	13	12	11	10	9	8
CHID8	CHID7			CHID6			CHID5
7	6	5	4	3	2	1	0
CHID5		CHID4			CHID3		

• CHIDx: Channel ID for the output slot x

CHIDx	Selected Transmit Channel
0x0	None. No data will be transmitted during this Slot x
0x1	Channel A data will be transferred during this slot time.
0x2	Channel B data will be transferred during this slot time



## 40.7.4 AC'97 Controller Codec Channel Receive Holding Register

**Register Name:** AC97C\_CORHR

**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
SDATA							
7	6	5	4	3	2	1	0
SDATA							

- **SDATA: Status Data**

Data sent by the CODEC in the third AC'97 input frame slot (Slot 2).

## 40.7.5 AC'97 Controller Codec Channel Transmit Holding Register

**Register Name:** AC97C\_COTHR

**Access Type:** Write-only

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

- **READ: Read/Write command**

0: Write operation to the CODEC register indexed by the CADDR address.

1: Read operation to the CODEC register indexed by the CADDR address.

This flag is sent during the second AC'97 frame slot

- **CADDR: CODEC control register index**

Data sent to the CODEC in the second AC'97 frame slot.

- **CDATA: Command Data**

Data sent to the CODEC in the third AC'97 frame slot (Slot 2).



#### 40.7.6 AC'97 Controller Channel A, Channel B Receive Holding Register

Register Name: AC97C\_CARHR, AC97C\_CBRHR

Access Type: Read-only

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

• **RDATA: Receive Data**

Received Data on channel x.

#### 40.7.7 AC'97 Controller Channel A, Channel B Transmit Holding Register

Register Name: AC97C\_CATHR, AC97C\_CBTHR

Access Type: Write-only

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

• **TDATA: Transmit Data**

Data to be sent on channel x.

## 40.7.8 AC'97 Controller Channel A Status Register

Register Name: AC97C\_CASR

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
RXBUFF	ENDRX	–	–	TXBUFE	ENDTX	–	–
7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	UNRUN	TXEMPTY	TXRDY

## 40.7.9 AC'97 Controller Channel B Status Register

Register Name: AC97C\_CBSR

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
–	–	OVRUN	RXRDY	–	UNRUN	TXEMPTY	TXRDY

## 40.7.10 AC'97 Controller Codec Channel Status Register

Register Name: AC97C\_COSR

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
–	–	OVRUN	RXRDY	–	UNRUN	TXEMPTY	TXRDY

- **TXRDY: Channel Transmit Ready**

0: Data has been loaded in Channel Transmit Register and is waiting to be loaded in the Channel Transmit Shift Register.

1: Channel Transmit Register is empty.

- **TXEMPTY: Channel Transmit Empty**

0: Data remains in the Channel Transmit Register or is currently transmitted from the Channel Transmit Shift Register.

1: Data in the Channel Transmit Register have been loaded in the Channel Transmit Shift Register and sent to the codec.

- **UNRUN: Transmit Underrun**



Active only when Variable Rate Mode is enabled (VRA bit set in the AC97C\_MR register). Automatically cleared by a processor read operation.

0: No data has been requested from the channel since the last read of the Status Register, or data has been available each time the CODEC requested new data from the channel since the last read of the Status Register.

1: Data has been emitted while no valid data to send has been previously loaded into the Channel Transmit Shift Register since the last read of the Status Register.

- **RXRDY: Channel Receive Ready**

0: Channel Receive Holding Register is empty.

1: Data has been received and loaded in Channel Receive Holding Register.

- **OVRUN: Receive Overrun**

Automatically cleared by a processor read operation.

0: No data has been loaded in the Channel Receive Holding Register while previous data has not been read since the last read of the Status Register.

1: Data has been loaded in the Channel Receive Holding Register while previous data has not yet been read since the last read of the Status Register.

- **ENDRX: End of Reception for Channel x**

0: The register AC97C\_CxRCCR has not reached 0 since the last write in AC97C\_CxRCCR or AC97C\_CxRNCR.

1: The register AC97C\_CxRCCR has reached 0 since the last write in AC97C\_CxRCCR or AC97C\_CxRNCR.

- **RXBUFF: Receive Buffer Full for Channel x**

0: AC97C\_CxRCCR or AC97C\_CxRNCR have a value other than 0.

1: Both AC97C\_CxRCCR and AC97C\_CxRNCR have a value of 0.

- **ENDTX: End of Transmission for Channel x**

0: The register AC97C\_CxTCR has not reached 0 since the last write in AC97C\_CxTCR or AC97C\_CxTNCR.

1: The register AC97C\_CxTCR has reached 0 since the last write in AC97C\_CxTCR or AC97C\_CxTNCR.

- **TXBUFE: Transmit Buffer Empty for Channel x**

0: AC97C\_CxTCR or AC97C\_CxTNCR have a value other than 0.

1: Both AC97C\_CxTCR and AC97C\_CxTNCR have a value of 0.

## 40.7.11 AC'97 Controller Channel A Mode Register

Register Name: AC97C\_CAMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	PDCEN	CEN	–	–	CEM	SIZE	
15	14	13	12	11	10	9	8
RXBUFF	ENDRX	–	–	TXBUFE	ENDTX	–	–
7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	UNRUN	TXEMPTY	TXRDY

## 40.7.12 AC'97 Controller Channel B Mode Register

Register Name: AC97C\_CBMR

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	CEN	–	–	CEM	SIZE	
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	UNRUN	TXEMPTY	TXRDY

- **CEM: Channel x Endian Mode**

0: Transferring data through Channel x is straightforward (Little-endian).

1: Transferring data through Channel x from/to a memory is performed with from/to Big-endian format translation.

- **SIZE: Channel x Data Size**

SIZE Encoding

SIZE	Selected Channel
0x0	20 bits
0x1	18bits
0x2	16 bits
0x3	10 bits

Note: Each time slot in the data phase is 20 bits long. For example, if a 16-bit sample stream is being played to an AC97 DAC, the first 16 bit positions are presented to the DAC MSB-justified. They are followed by the next four bit positions that the AC'97 Controller fills with zeroes. This process ensures that the least significant bits do not introduce any DC biasing, regardless of the implemented DAC's resolution (16-, 18-, or 20-bit).

- **CEN: Channel x Enable**

0: Data transfer is disabled on Channel x.

1: Data transfer is enabled on Channel x.

- **PDCEN: Peripheral Data Controller Channel Enable**



0: Channel x is not transferred through a Peripheral Data Controller Channel. Related PDC flags are ignored or not generated.

1: Channel x is transferred through a Peripheral Data Controller Channel. Related PDC flags are taken into account or generated.





### 40.7.13 AC'97 Controller Codec Channel Mode Register

Register Name: AC97C\_COMR

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
–	–	OVRUN	RXRDY	–	UNRUN	TXEMPTY	TXRDY

- **TXRDY: Channel Transmit Ready Interrupt Enable**
  - **TXEMPTY: Channel Transmit Empty Interrupt Enable**
  - **UNRUN: Transmit Underrun Interrupt Enable**
  - **RXRDY: Channel Receive Ready Interrupt Enable**
  - **OVRUN: Receive Overrun Interrupt Enable**
  - **ENDRX: End of Reception for channel x Interrupt Enable**
  - **RXBUFF: Receive Buffer Full for channel x Interrupt Enable**
  - **ENDTX: End of Transmission for channel x Interrupt Enable**
  - **TXBUFE: Transmit Buffer Empty for channel x Interrupt Enable**
- 0: Read: the corresponding interrupt is disabled. Write: disables the corresponding interrupt.  
1: Read: the corresponding interrupt is enabled. Write: enables the corresponding interrupt.



## 40.7.14 AC'97 Controller Status Register

**Register Name:** AC97C\_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
–	–		CBEVT	CAEVT	COEVT	WKUP	SOF

WKUP and SOF flags in AC97C\_SR register are automatically cleared by a processor read operation.

- **SOF: Start Of Frame**

0: No Start of Frame has been detected since the last read of the Status Register.

1: At least one Start of frame has been detected since the last read of the Status Register.

- **WKUP: Wake Up detection**

0: No Wake-up has been detected.

1: At least one rising edge on SDATA\_IN has been asynchronously detected. That means AC'97 Codec has notified a wake-up.

- **COEVT: CODEC Channel Event**

A Codec channel event occurs when AC97C\_COSR AND AC97C\_COMR is not 0. COEVT flag is automatically cleared when the channel event condition is cleared.

0: No event on the CODEC channel has been detected since the last read of the Status Register.

1: At least one event on the CODEC channel is active.

- **CAEVT: Channel A Event**

A channel A event occurs when AC97C\_CASR AND AC97C\_CAMR is not 0. CAEVT flag is automatically cleared when the channel event condition is cleared.

0: No event on the channel A has been detected since the last read of the Status Register.

1: At least one event on the channel A is active.

- **CBEVT: Channel B Event**

A channel B event occurs when AC97C\_CBSR AND AC97C\_CBMR is not 0. CBEVT flag is automatically cleared when the channel event condition is cleared.

0: No event on the channel B has been detected since the last read of the Status Register.

1: At least one event on the channel B is active.



### 40.7.15 AC'97 Controller Interrupt Enable Register

Register Name: AC97C\_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
–	–	–	CBEVT	CAEVT	COEVT	WKUP	SOF

- **SOF: Start Of Frame**
- **WKUP: Wake Up**
- **COEVT: Codec Event**
- **CAEVT: Channel A Event**
- **CBEVT: Channel B Event**

0: No effect.

1: Enables the corresponding interrupt.

### 40.7.16 AC'97 Controller Interrupt Disable Register

Register Name: AC97C\_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
–	–	–	CBEVT	CAEVT	COEVT	WKUP	SOF

- **SOF: Start Of Frame**
- **WKUP: Wake Up**
- **COEVT: Codec Event**
- **CAEVT: Channel A Event**
- **CBEVT: Channel B Event**

0: No effect.

1: Disables the corresponding interrupt.

## 40.7.17 AC'97 Controller Interrupt Mask Register

**Register Name:** AC97C\_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
–	–	–	CBEVT	CAEVT	COEVT	WKUP	SOF

- **SOF: Start Of Frame**
- **WKUP: Wake Up**
- **COEVT: Codec Event**
- **CAEVT: Channel A Event**
- **CBEVT: Channel B Event**

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.



## 41. USB High Speed Device Port (UDPHS)

### 41.1 Description

The USB High Speed Device Port (UDPHS) is compliant with the Universal Serial Bus (USB), rev 2.0 High Speed device specification.

Each endpoint can be configured in one of several USB transfer types. It can be associated with one, two or three banks of a dual-port RAM used to store the current data payload. If two or three 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.

**Table 41-1.** UDPHS Endpoint Description

Endpoint #	Mnemonic	Nb Bank	DMA	High Band Width	Max. Endpoint Size	Endpoint Type
0	EPT_0	1	N	N	64	Control
1	EPT_1	2	Y	Y	1024	Ctrl/Bulk/Iso <sup>(1)</sup> /Interrupt
2	EPT_2	2	Y	Y	1024	Ctrl/Bulk/Iso <sup>(1)</sup> /Interrupt
3	EPT_3	3	Y	N	1024	Ctrl/Bulk/Iso <sup>(1)</sup> /Interrupt
4	EPT_4	3	Y	N	1024	Ctrl/Bulk/Iso <sup>(1)</sup> /Interrupt
5	EPT_5	3	Y	Y	1024	Ctrl/Bulk/Iso <sup>(1)</sup> /Interrupt
6	EPT_6	3	Y	Y	1024	Ctrl/Bulk/Iso <sup>(1)</sup> /Interrupt

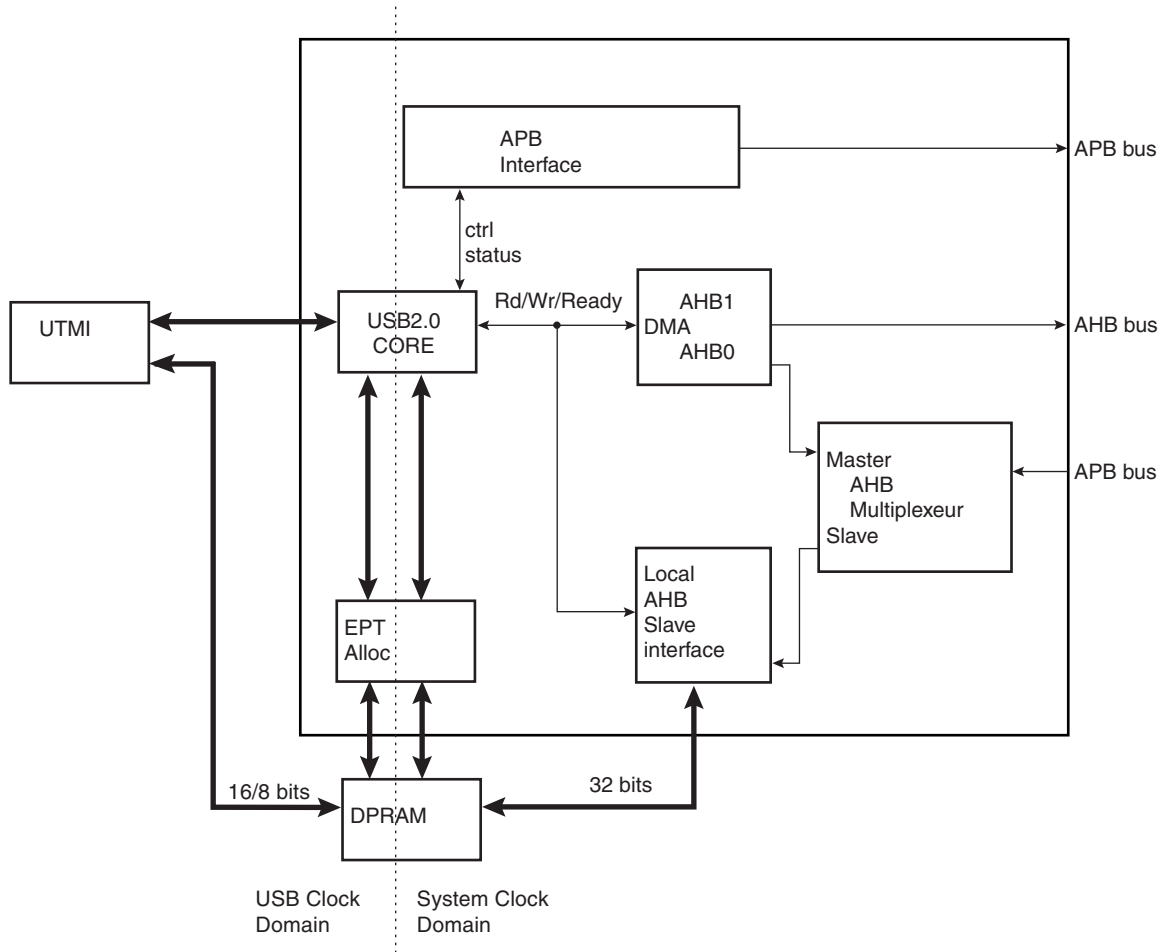
Note: 1. In Isochronous Mode (Iso), it is preferable that High Band Width capability is available.

The size of internal DPRAM is 4 KB.

Suspend and resume are automatically detected by the UDPHS device, which notifies the processor by raising an interrupt.

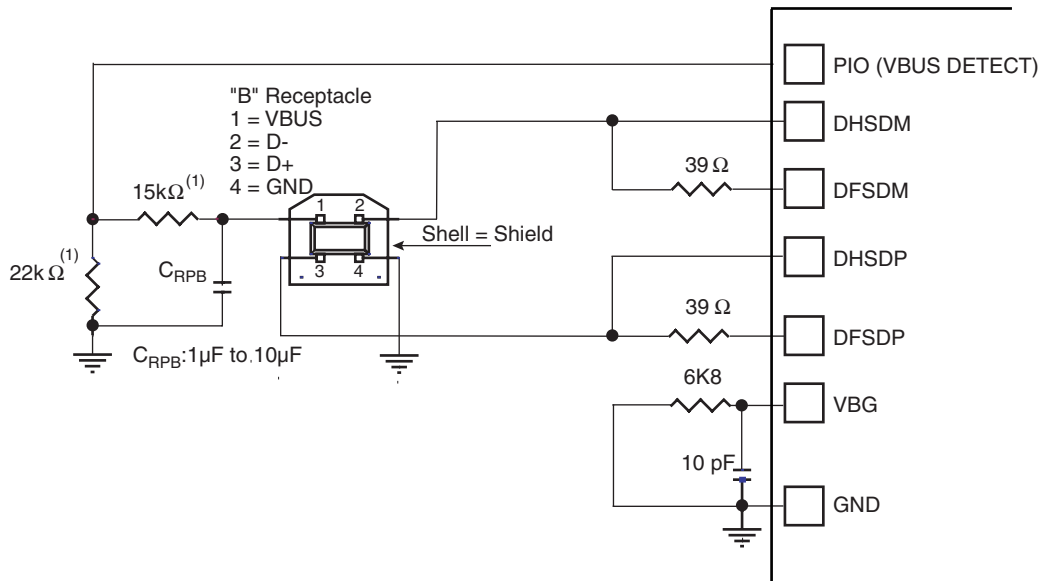
## 41.2 Block Diagram

Figure 41-1. Block Diagram:



## 41.3 Typical Connection

Figure 41-2. Board Schematic



Note: The values shown on the 22 kΩ and 15 kΩ resistors are only valid with 3V3 supplied PIOs.

## 41.4 Functional Description

### 41.4.1 USB V2.0 High Speed Device Port Introduction

The USB V2.0 High Speed Device Port 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.

### 41.4.2 USB V2.0 High Speed Transfer Types

A communication flow is carried over one of four transfer types defined by the USB device.

A device provides several logical communication pipes with the host. To each logical pipe is associated an endpoint. Transfer through a pipe belongs to one of the four transfer types:

- Control Transfers: Used to configure a device at attach time and can be used for other device-specific purposes, including control of other pipes on the device.
- Bulk Data Transfers: Generated or consumed in relatively large burst quantities and have wide dynamic latitude in transmission constraints.
- Interrupt Data Transfers: Used for timely but reliable delivery of data, for example, characters or coordinates with human-perceptible echo or feedback response characteristics.
- Isochronous Data Transfers: Occupy a prenegotiated amount of USB bandwidth with a prenegotiated delivery latency. (Also called streaming real time transfers.)

As indicated below, transfers are sequential events carried out on the USB bus.

Endpoints must be configured according to the transfer type they handle.

**Table 41-2.** USB Communication Flow

Transfer	Direction	Bandwidth	Endpoint Size	Error Detection	Retrying
Control	Bidirectional	Not guaranteed	8,16,32,64	Yes	Automatic
Isochronous	Unidirectional	Guaranteed	8-1024	Yes	No
Interrupt	Unidirectional	Not guaranteed	8-1024	Yes	Yes
Bulk	Unidirectional	Not guaranteed	8-512	Yes	Yes

#### 41.4.3 USB Transfer Event Definitions

A transfer is composed of one or several transactions;

**Table 41-3.** USB Transfer Events

<b>CONTROL (bidirectional)</b>	Control Transfers <sup>(1)</sup>	<ul style="list-style-type: none"> <li>• Setup transaction → Data IN transactions → Status OUT transaction</li> <li>• Setup transaction → Data OUT transactions → Status IN transaction</li> <li>• Setup transaction → Status IN transaction</li> </ul>
<b>IN (device toward host)</b>	Bulk IN Transfer	• Data IN transaction → Data IN transaction
	Interrupt IN Transfer	• Data IN transaction → Data IN transaction
	Isochronous IN Transfer <sup>(2)</sup>	• Data IN transaction → Data IN transaction
<b>OUT (host toward device)</b>	Bulk OUT Transfer	• Data OUT transaction → Data OUT transaction
	Interrupt OUT Transfer	• Data OUT transaction → Data OUT transaction
	Isochronous OUT Transfer <sup>(2)</sup>	• Data OUT transaction → Data OUT transaction

Notes: 1. Control transfer must use endpoints with one bank and can be aborted using a stall handshake.

2. Isochronous transfers must use endpoints configured with two or three banks.

An endpoint handles all transactions related to the type of transfer for which it has been configured.

#### 41.4.4 USB V2.0 High Speed BUS Transactions

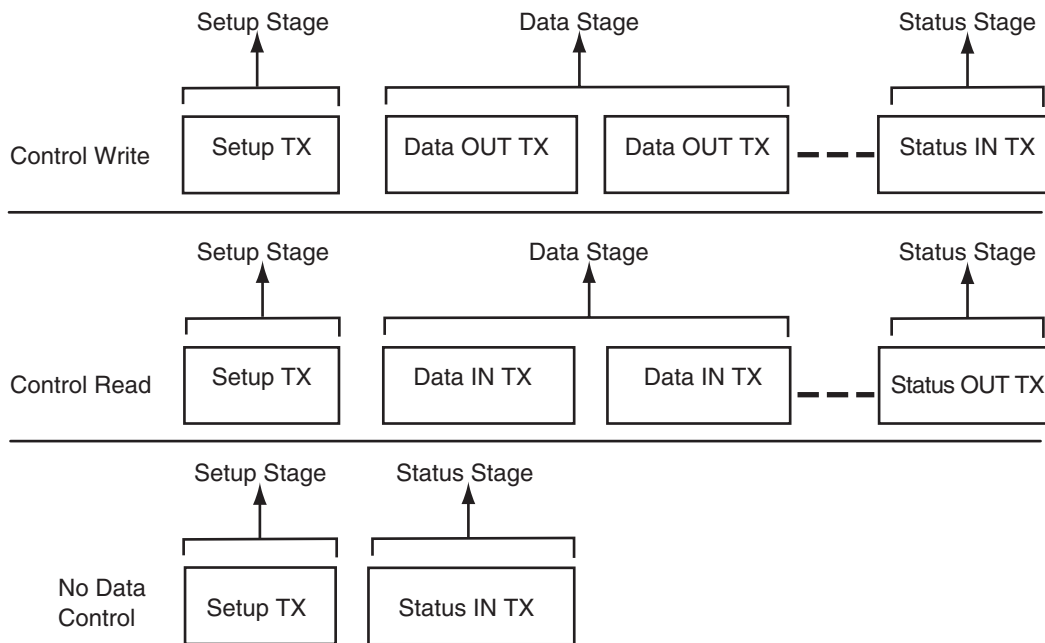
Each transfer results in one or more transactions over the USB bus.

There are five kinds of transactions flowing across the bus in packets:

1. Setup Transaction
2. Data IN Transaction
3. Data OUT Transaction
4. Status IN Transaction
5. Status OUT Transaction



**Figure 41-3.** Control Read and Write Sequences



A status IN or OUT transaction is identical to a data IN or OUT transaction.

#### 41.4.5 Endpoint Configuration

The endpoint 0 is always a control endpoint, it must be programmed and active in order to be enabled when the End Of Reset interrupt occurs.

To configure the endpoints:

- Fill the configuration register (UDPHS\_EPTCFG) with the endpoint size, direction (IN or OUT), type (CTRL, Bulk, IT, ISO) and the number of banks.
- Fill the number of transactions (NB\_TRANS) for isochronous endpoints.

**Note:** For control endpoints the direction has no effect.

- Verify that the EPT\_MAPD flag is set. This flag is set if the endpoint size and the number of banks are correct compared to the FIFO maximum capacity and the maximum number of allowed banks.
- Configure control flags of the endpoint and enable it in UDPHS\_EPTCTLENBx according to [“UDPHS Endpoint Control Register” on page 795](#).

Control endpoints can generate interrupts and use only 1 bank.

All endpoints (except endpoint 0) can be configured either as Bulk, Interrupt or Isochronous. See [Table 41-1. UDPHS Endpoint Description](#).

The maximum packet size they can accept corresponds to the maximum endpoint size.

**Note:** The endpoint size of 1024 is reserved for isochronous endpoints.

The size of the DPRAM is 4 KB. The DPR is shared by all active endpoints. The memory size required by the active endpoints must not exceed the size of the DPRAM.

$$\text{SIZE\_DPRAM} = \text{SIZE\_EPT0}$$

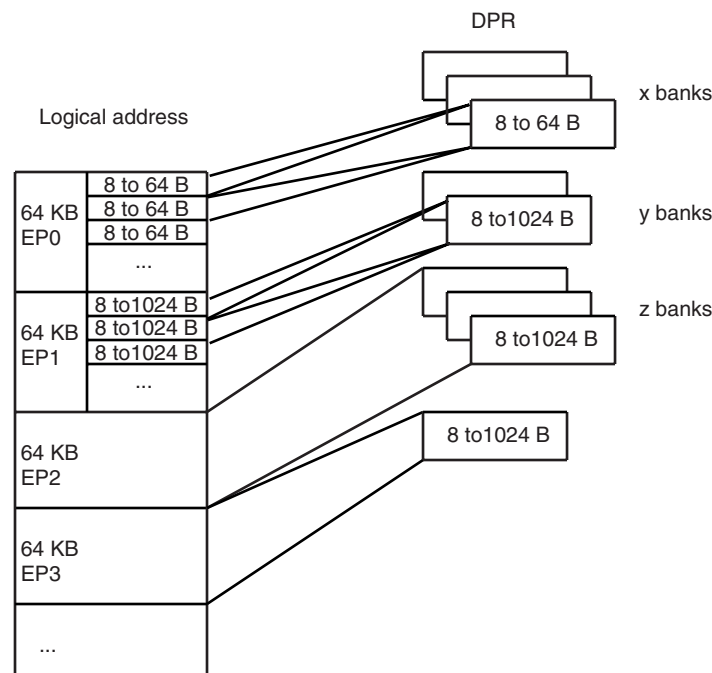
+ NB\_BANK\_EPT1 x SIZE\_EPT1  
 + NB\_BANK\_EPT2 x SIZE\_EPT2  
 + NB\_BANK\_EPT3 x SIZE\_EPT3  
 + NB\_BANK\_EPT4 x SIZE\_EPT4  
 + NB\_BANK\_EPT5 x SIZE\_EPT5  
 + NB\_BANK\_EPT6 x SIZE\_EPT6  
 +... (refer to [41.5.8 UDPHS Endpoint Configuration Register](#))

If a user tries to configure endpoints with a size the sum of which is greater than the DPRAM, then the EPT\_MAPD is not set.

The application has access to the physical block of DPR reserved for the endpoint through a 64 KB logical address space.

The physical block of DPR allocated for the endpoint is remapped all along the 64 KB logical address space. The application can write a 64 KB buffer linearly.

**Figure 41-4.** Logical Address Space for DPR Access:



Configuration examples of UDPHS\_EPTCTLx ([UDPHS Endpoint Control Register](#)) for Bulk IN endpoint type follow below.

- With DMA
  - AUTO\_VALID: Automatically validate the packet and switch to the next bank.
  - EPT\_ENABL: Enable endpoint.
  
- Without DMA:
  - TX\_BK\_RDY: An interrupt is generated after each transmission.

- EPT\_ENABL: Enable endpoint.

Configuration examples of Bulk OUT endpoint type follow below.

- With DMA
  - AUTO\_VALID: Automatically validate the packet and switch to the next bank.
  - EPT\_ENABL: Enable endpoint.
- Without DMA
  - RX\_BK\_RDY: An interrupt is sent after a new packet has been stored in the endpoint FIFO.
  - EPT\_ENABL: Enable endpoint.

#### 41.4.6 Transfer With DMA

USB packets of any length may be transferred when required by the UDPHS Device. These transfers always feature sequential addressing.

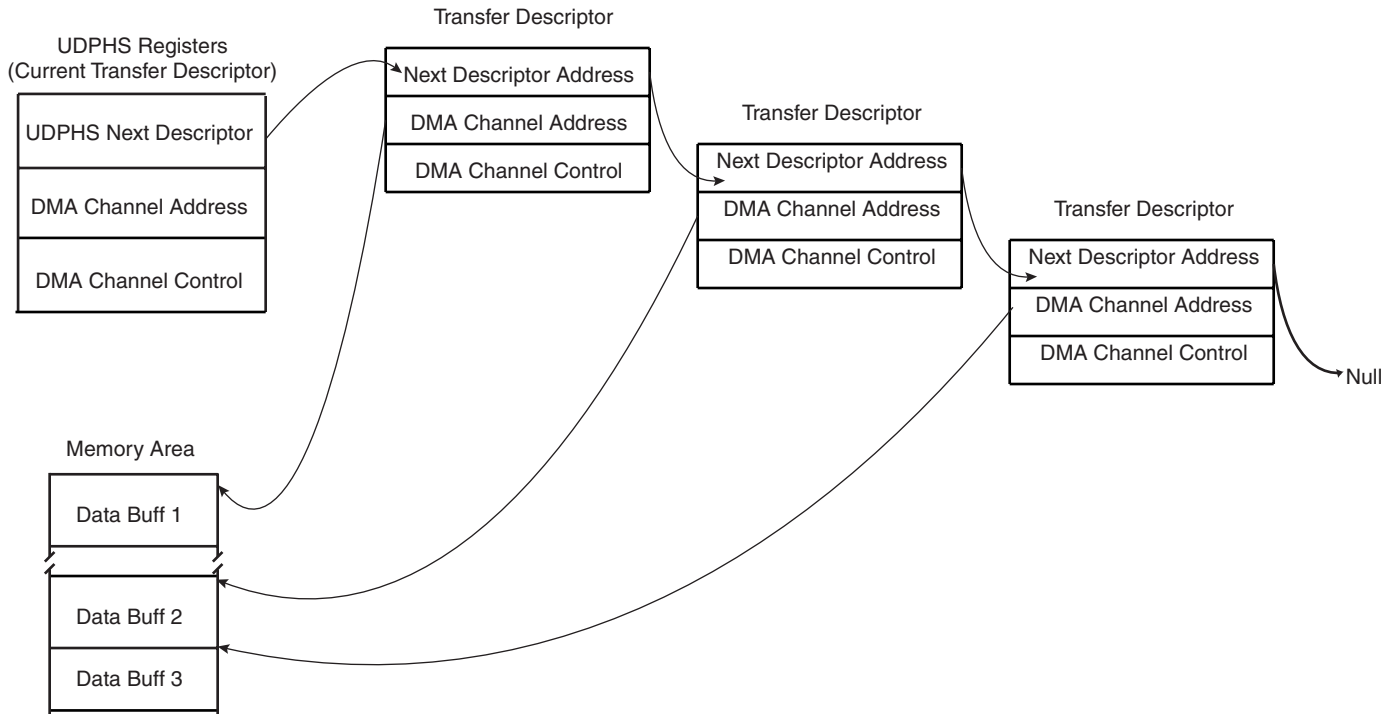
Packet data AHB bursts may be locked on a DMA buffer basis for drastic overall AHB bus bandwidth performance boost with paged memories. These clock-cycle consuming memory row (or bank) changes will then likely not occur, or occur only once instead of dozens times, during a single big USB packet DMA transfer in case another AHB master addresses the memory. This means up to 128-word single-cycle unbroken AHB bursts for Bulk endpoints and 256-word single-cycle unbroken bursts for isochronous endpoints. This maximum burst length is then controlled by the lowest programmed USB endpoint size (EPT\_SIZE bit in the UDPHS\_EPTCFGx register) and DMA Size (BUFF\_LENGTH bit in the UDPHS\_DMACONTROLx register).

The USB 2.0 device average throughput may be up to nearly 60 MBytes. Its internal slave average access latency decreases as burst length increases due to the 0 wait-state side effect of unchanged endpoints. If at least 0 wait-state word burst capability is also provided by the external DMA AHB bus slaves, each of both DMA AHB busses need less than 50% bandwidth allocation for full USB 2.0 bandwidth usage at 30 MHz, and less than 25% at 60 MHz.

The UDPHS DMA Channel Transfer Descriptor is described in [“UDPHS DMA Channel Transfer Descriptor” on page 806](#).

Note: In case of debug, be careful to address the DMA to an SRAM address even if a remap is done.

**Figure 41-5.** Example of DMA Chained List:



## 41.4.7 Transfer Without DMA

**Important.** If the DMA is not to be used, it is necessary that it be disabled because otherwise it can be enabled by previous versions of software **without warning**. If this should occur, the DMA can process data before an interrupt without knowledge of the user.

The recommended means to disable DMA is as follows:

```
// Reset IP UDPHS
    AT91C_BASE_UDPHS->UDPHS_CTRL &= ~AT91C_UDPHS_EN_UDPHS;
    AT91C_BASE_UDPHS->UDPHS_CTRL |= AT91C_UDPHS_EN_UDPHS;
// With OR without DMA !!!
    for( i=1; i<=((AT91C_BASE_UDPHS->UDPHS_IPFEATURES &
AT91C_UDPHS_DMA_CHANNEL_NBR)>>4); i++ ) {
// RESET endpoint canal DMA:
    // DMA stop channel command
    AT91C_BASE_UDPHS->UDPHS_DMA[i].UDPHS_DMACONTROL = 0; // STOP
command
// Disable endpoint
    AT91C_BASE_UDPHS->UDPHS_EPT[i].UDPHS_EPTCTLDIS |= 0xFFFFFFFF;
// Reset endpoint config
    AT91C_BASE_UDPHS->UDPHS_EPT[i].UDPHS_EPTCTLCFG = 0;
// Reset DMA channel (Buff count and Control field)
    AT91C_BASE_UDPHS->UDPHS_DMA[i].UDPHS_DMACONTROL = 0x02; // NON
STOP command
// Reset DMA channel 0 (STOP)
    AT91C_BASE_UDPHS->UDPHS_DMA[i].UDPHS_DMACONTROL = 0; // STOP
command
// Clear DMA channel status (read the register for clear it)
    AT91C_BASE_UDPHS->UDPHS_DMA[i].UDPHS_DMASTATUS =
AT91C_BASE_UDPHS->UDPHS_DMA[i].UDPHS_DMASTATUS;
}
```

## 41.4.8 Handling Transactions with USB V2.0 Device Peripheral

### 41.4.8.1 Setup Transaction

The setup packet is valid in the DPR while RX\_SETUP is set. Once RX\_SETUP is cleared by the application, the UDPHS accepts the next packets sent over the device endpoint.

When a valid setup packet is accepted by the UDPHS:

- the UDPHS device automatically acknowledges the setup packet (sends an ACK response)
- payload data is written in the endpoint
- sets the RX\_SETUP interrupt
- the BYTE\_COUNT field in the UDPHS\_EPTSTAx register is updated

An endpoint interrupt is generated while RX\_SETUP in the UDPHS\_EPTSTAx register is not cleared. This interrupt is carried out to the microcontroller if interrupts are enabled for this endpoint.

Thus, firmware must detect RX\_SETUP polling UDPHS\_EPTSTAx or catching an interrupt, read the setup packet in the FIFO, then clear the RX\_SETUP bit in the UDPHS\_EPTCLRSTA register to acknowledge the setup stage.

If STALL\_SNT was set to 1, then this bit is automatically reset when a setup token is detected by the device. Then, the device still accepts the setup stage. (See Section 41.4.8.15 “STALL” on page 769).

#### 41.4.8.2 NYET

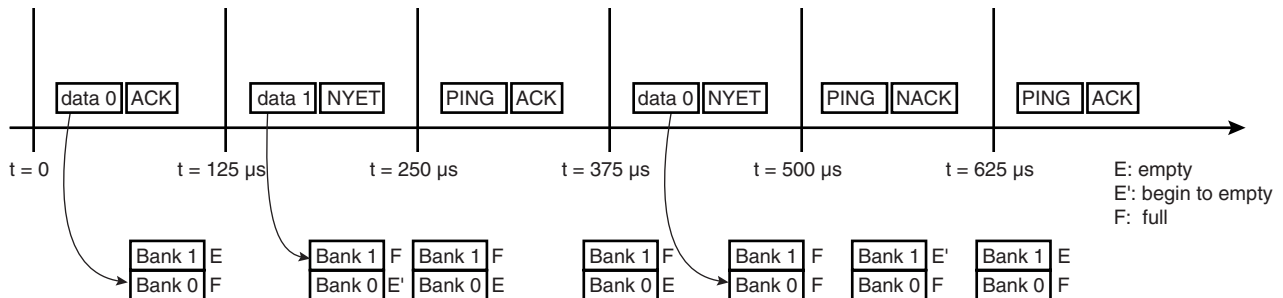
NYET is a High Speed only handshake. It is returned by a High Speed endpoint as part of the PING protocol.

High Speed devices must support an improved NAK mechanism for Bulk OUT and control endpoints (except setup stage). This mechanism allows the device to tell the host whether it has sufficient endpoint space for the next OUT transfer (see USB 2.0 spec 8.5.1 NAK Limiting via Ping Flow Control).

The NYET/ACK response to a High Speed Bulk OUT transfer and the PING response are automatically handled by hardware in the UDPHS\_EPTCTLx register (except when the user wants to force a NAK response by using the NYET\_DIS bit).

If the endpoint responds instead to the OUT/DATA transaction with an NYET handshake, this means that the endpoint accepted the data but does not have room for another data payload. The host controller must return to using a PING token until the endpoint indicates it has space available.

**Figure 41-6.** NYET Example with Two Endpoint Banks



#### 41.4.8.3 Data IN

#### 41.4.8.4 Bulk IN or Interrupt IN

Data IN packets are sent by the device during the data or the status stage of a control transfer or during an (interrupt/bulk/isochronous) IN transfer. Data buffers are sent packet by packet under the control of the application or under the control of the DMA channel.

There are three ways for an application to transfer a buffer in several packets over the USB:

- packet by packet (see 41.4.8.5 below)
- 64 KB (see 41.4.8.5 below)
- DMA (see 41.4.8.6 below)

#### 41.4.8.5 Bulk IN or Interrupt IN: Sending a Packet Under Application Control (Device to Host)

The application can write one or several banks.

A simple algorithm can be used by the application to send packets regardless of the number of banks associated to the endpoint.

Algorithm Description for Each Packet:

- The application waits for TX\_PK\_RDY flag to be cleared in the UDPHS\_EPTSTAx register before it can perform a write access to the DPR.
- The application writes one USB packet of data in the DPR through the 64 KB endpoint logical memory window.
- The application sets TX\_PK\_RDY flag in the UDPHS\_EPTSETSTAx register.

The application is notified that it is possible to write a new packet to the DPR by the TX\_PK\_RDY interrupt. This interrupt can be enabled or masked by setting the TX\_PK\_RDY bit in the UDPHS\_EPTCTLENB/UDPHS\_EPTCTLDIS register.

Algorithm Description to Fill Several Packets:

Using the previous algorithm, the application is interrupted for each packet. It is possible to reduce the application overhead by writing linearly several banks at the same time. The AUTO\_VALID bit in the UDPHS\_EPTCTLx must be set by writing the AUTO\_VALID bit in the UDPHS\_EPTCTLENBx register.

The auto-valid-bank mechanism allows the transfer of data (IN and OUT) without the intervention of the CPU. This means that bank validation (set TX\_PK\_RDY or clear the RX\_BK\_RDY bit) is done by hardware.

- The application checks the BUSY\_BANK\_STA field in the UDPHS\_EPTSTAx register. The application must wait that at least one bank is free.
- The application writes a number of bytes inferior to the number of free DPR banks for the endpoint. Each time the application writes the last byte of a bank, the TX\_PK\_RDY signal is automatically set by the UDPHS.
- If the last packet is incomplete (i.e., the last byte of the bank has not been written) the application must set the TX\_PK\_RDY bit in the UDPHS\_EPTSETSTAx register.

The application is notified that all banks are free, so that it is possible to write another burst of packets by the BUSY\_BANK interrupt. This interrupt can be enabled or masked by setting the BUSY\_BANK flag in the UDPHS\_EPTCTLENB and UDPHS\_EPTCTLDIS registers.

This algorithm must not be used for isochronous transfer. In this case, the ping-pong mechanism does not operate.

A Zero Length Packet can be sent by setting just the TX\_PKTRDY flag in the UDPHS\_EPTSETSTAx register.

#### 41.4.8.6 Bulk IN or Interrupt IN: Sending a Buffer Using DMA (Device to Host)

The UDPHS integrates a DMA host controller. This DMA controller can be used to transfer a buffer from the memory to the DPR or from the DPR to the processor memory under the UDPHS control. The DMA can be used for all transfer types except control transfer.

Example DMA configuration:

1. Program UDPHS\_DMAADDRESS x with the address of the buffer that should be transferred.
2. Enable the interrupt of the DMA in UDPHS\_IEN
3. Program UDPHS\_DMACONTROLx:

- Size of buffer to send: size of the buffer to be sent to the host.
- END\_B\_EN: The endpoint can validate the packet (according to the values programmed in the AUTO\_VALID and SHRT\_PCKT fields of UDPHS\_EPTCTLx.) (See “[UDPHS Endpoint Control Register](#)” on page 795 and [Figure 41-11. Autovalid with DMA](#))
- END\_BUFFIT: generate an interrupt when the BUFF\_COUNT in UDPHS\_DMASTATUSx reaches 0.
- CHANN\_ENB: Run and stop at end of buffer

The auto-valid-bank mechanism allows the transfer of data (IN & OUT) without the intervention of the CPU. This means that bank validation (set TX\_PK\_RDY or clear the RX\_BK\_RDY bit) is done by hardware.

A transfer descriptor can be used. Instead of programming the register directly, a descriptor should be programmed and the address of this descriptor is then given to UDPHS\_DMANXTDSC to be processed after setting the LDNXT\_DSC field (Load Next Descriptor Now) in UDPHS\_DMACONTROLx register.

The structure that defines this transfer descriptor must be aligned.

Each buffer to be transferred must be described by a DMA Transfer descriptor (see “[UDPHS DMA Channel Transfer Descriptor](#)” on page 806). Transfer descriptors are chained. Before executing transfer of the buffer, the UDPHS may fetch a new transfer descriptor from the memory address pointed by the UDPHS\_DMANXTDSCx register. Once the transfer is complete, the transfer status is updated in the UDPHS\_DMASTATUSx register.

To chain a new transfer descriptor with the current DMA transfer, the DMA channel must be stopped. To do so, INTDIS\_DMA and TX\_BK\_RDY may be set in the UDPHS\_EPTCTLENBx register. It is also possible for the application to wait for the completion of all transfers. In this case the LDNXT\_DSC field in the last transfer descriptor UDPHS\_DMACONTROLx register must be set to 0 and CHANN\_ENB set to 1.

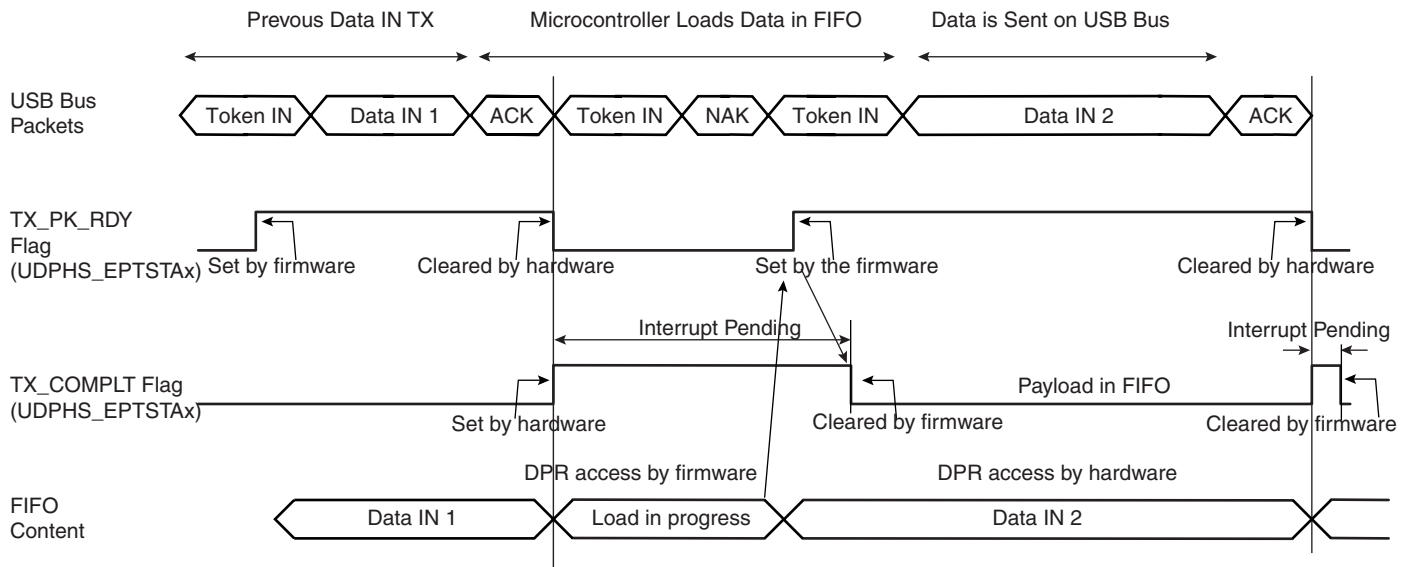
Then the application can chain a new transfer descriptor.

The INTDIS\_DMA can be used to stop the current DMA transfer if an enabled interrupt is triggered. This can be used to stop DMA transfers in case of errors.

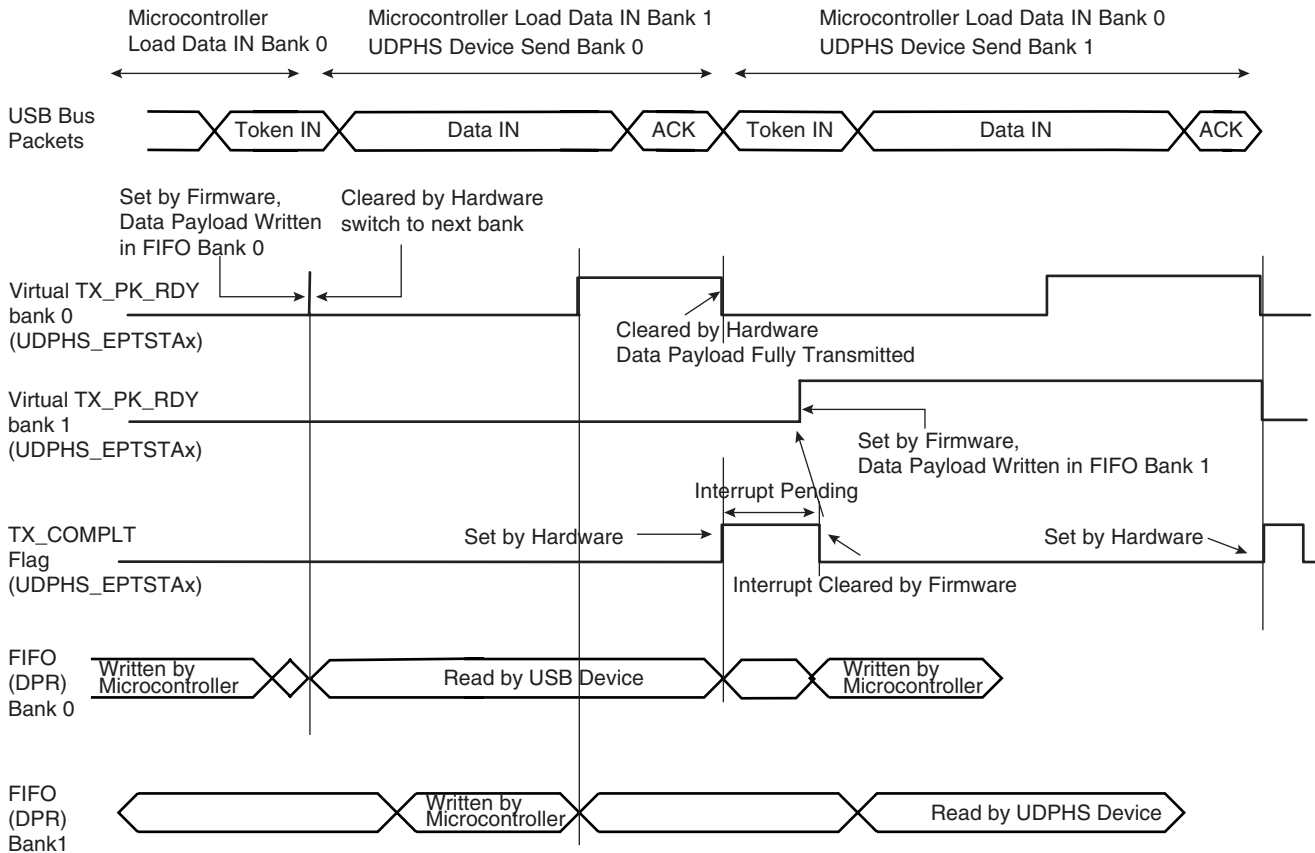
The application can be notified at the end of any buffer transfer (ENB\_BUFFIT bit in the UDPHS\_DMACONTROLx register).



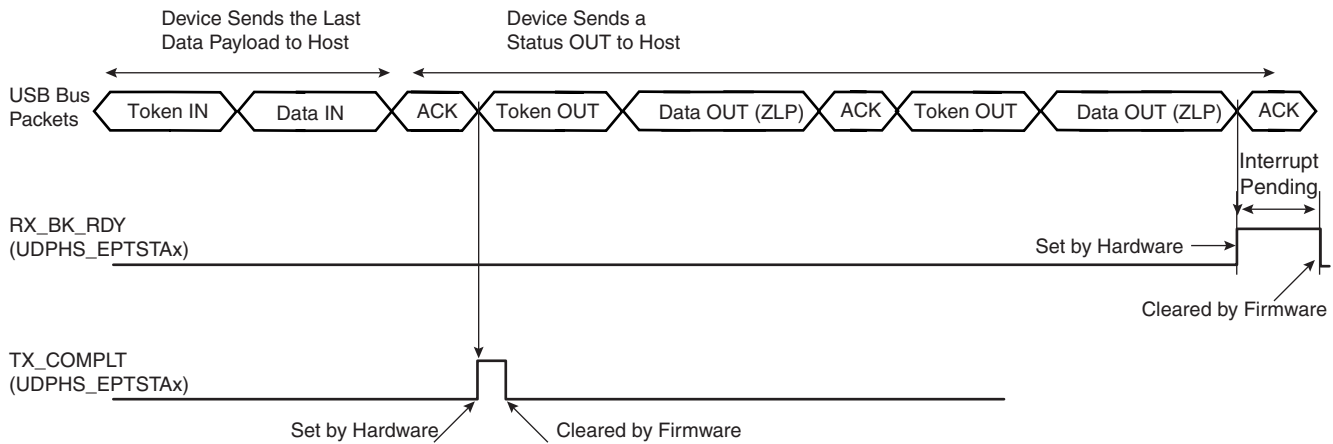
**Figure 41-7.** Data IN Transfer for Endpoint with One Bank



**Figure 41-8.** Data IN Transfer for Endpoint with Two Banks

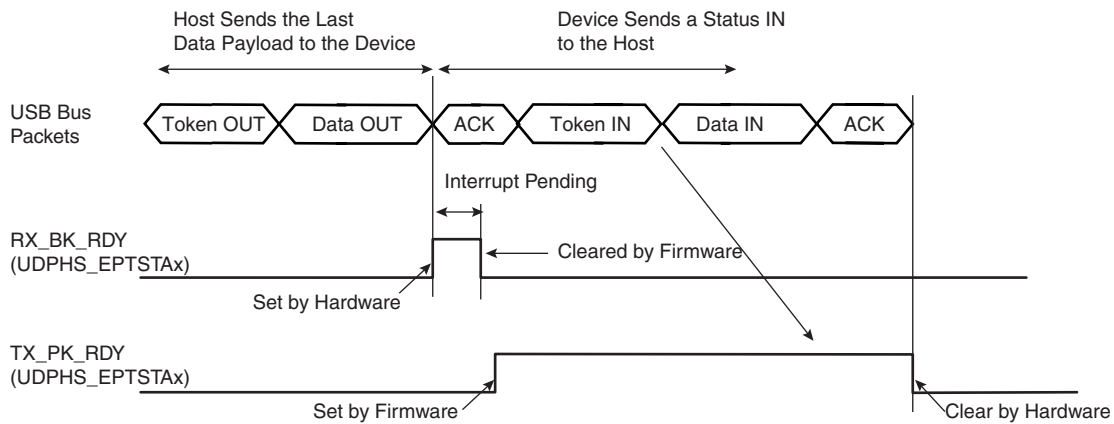


**Figure 41-9.** Data IN Followed By Status OUT Transfer at the End of a Control Transfer



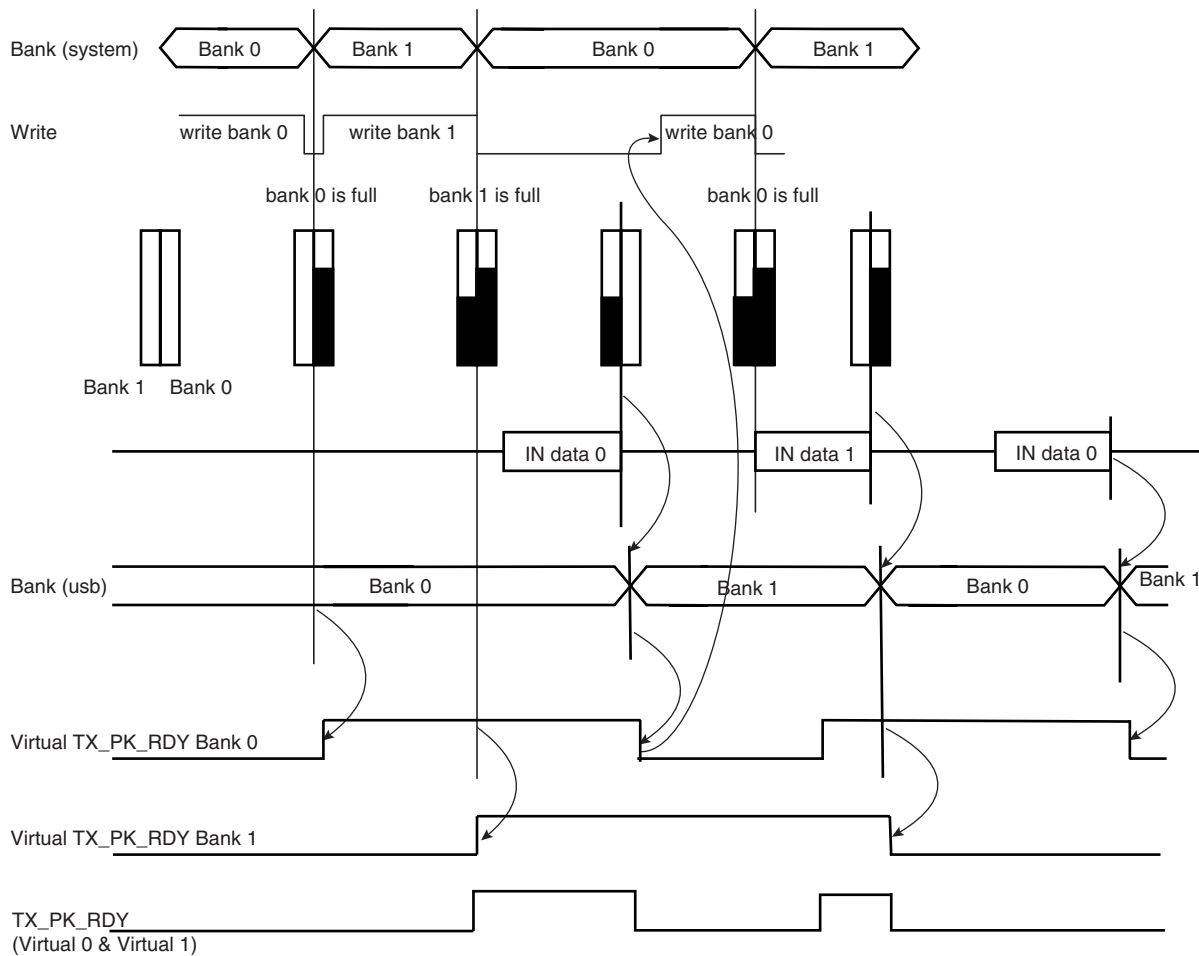
**Note:** A NAK handshake is always generated at the first status stage token.

**Figure 41-10.** Data OUT Followed by Status IN Transfer



**Note:** Before proceeding to the status stage, the software should determine that there is no risk of extra data from the host (data stage). If not certain (non-predictable data stage length), then the software should wait for a NAK-IN interrupt before proceeding to the status stage. This precaution should be taken to avoid collision in the FIFO.

**Figure 41-11. Autovalid with DMA**



**Note:** In the illustration above Autovalid validates a bank as full, although this might not be the case, in order to continue processing data and to send to DMA.

#### 41.4.8.7 Isochronous IN

Isochronous-IN is used to transmit a stream of data whose timing is implied by the delivery rate. Isochronous transfer provides periodic, continuous communication between host and device.

It guarantees bandwidth and low latencies appropriate for telephony, audio, video, etc.

If the endpoint is not available ( $TX\_PK\_RDY = 0$ ), then the device does not answer to the host. An `ERR_FL_ISO` interrupt is generated in the `UDPHS_EPTSTAx` register and once enabled, then sent to the CPU.

The `STALL_SNT` command bit is not used for an ISO-IN endpoint.

#### 41.4.8.8 High Bandwidth Isochronous Endpoint Handling: IN Example

For high bandwidth isochronous endpoints, the DMA can be programmed with the number of transactions (`BUFF_LENGTH` field in `UDPHS_DMACONTROLx`) and the system should provide

the required number of packets per microframe, otherwise, the host will notice a sequencing problem.

A response should be made to the first token IN recognized inside a microframe under the following conditions:

- If at least one bank has been validated, the correct DATA<sub>x</sub> corresponding to the programmed Number Of Transactions per Microframe (NB\_TRANS) should be answered. In case of a subsequent missed or corrupted token IN inside the microframe, the USB 2.0 Core available data bank(s) that should normally have been transmitted during that microframe shall be flushed at its end. If this flush occurs, an error condition is flagged (ERR\_FLUSH is set in UDPHS\_EPTSTAx).
- If no bank is validated yet, the default DATA0 ZLP is answered and underflow is flagged (ERR\_FL\_ISO is set in UDPHS\_EPTSTAx). Then, no data bank is flushed at microframe end.
- If no data bank has been validated at the time when a response should be made for the second transaction of NB\_TRANS = 3 transactions microframe, a DATA1 ZLP is answered and underflow is flagged (ERR\_FL\_ISO is set in UDPHS\_EPTSTAx). If and only if remaining untransmitted banks for that microframe are available at its end, they are flushed and an error condition is flagged (ERR\_FLUSH is set in UDPHS\_EPTSTAx).
- If no data bank has been validated at the time when a response should be made for the last programmed transaction of a microframe, a DATA0 ZLP is answered and underflow is flagged (ERR\_FL\_ISO is set in UDPHS\_EPTSTAx). If and only if the remaining untransmitted data bank for that microframe is available at its end, it is flushed and an error condition is flagged (ERR\_FLUSH is set in UDPHS\_EPTSTAx).
- If at the end of a microframe no valid token IN has been recognized, no data bank is flushed and no error condition is reported.

At the end of a microframe in which at least one data bank has been transmitted, if less than NB\_TRANS banks have been validated for that microframe, an error condition is flagged (ERR\_TRANS is set in UDPHS\_EPTSTAx).

Cases of Error (in UDPHS\_EPTSTAx)

- ERR\_FL\_ISO: There was no data to transmit inside a microframe, so a ZLP is answered by default.
- ERR\_FLUSH: At least one packet has been sent inside the microframe, but the number of token IN received is lesser than the number of transactions actually validated (TX\_BK\_RDY) and likewise with the NB\_TRANS programmed.
- ERR\_TRANS: At least one packet has been sent inside the microframe, but the number of token IN received is lesser than the number of programmed NB\_TRANS transactions and the packets not requested were not validated.
- ERR\_FL\_ISO + ERR\_FLUSH: At least one packet has been sent inside the microframe, but the data has not been validated in time to answer one of the following token IN.
- ERR\_FL\_ISO + ERR\_TRANS: At least one packet has been sent inside the microframe, but the data has not been validated in time to answer one of the following token IN and the data can be discarded at the microframe end.
- ERR\_FLUSH + ERR\_TRANS: The first token IN has been answered and it was the only one received, a second bank has been validated but not the third, whereas NB\_TRANS was waiting for three transactions.

- **ERR\_FL\_ISO + ERR\_FLUSH + ERR\_TRANS:** The first token IN has been treated, the data for the second Token IN was not available in time, but the second bank has been validated before the end of the microframe. The third bank has not been validated, but three transactions have been set in NB\_TRANS.

#### 41.4.8.9 Data OUT

#### 41.4.8.10 Bulk OUT or Interrupt OUT

Like data IN, data OUT packets are sent by the host during the data or the status stage of control transfer or during an interrupt/bulk/isochronous OUT transfer. Data buffers are sent packet by packet under the control of the application or under the control of the DMA channel.

#### 41.4.8.11 Bulk OUT or Interrupt OUT: Receiving a Packet Under Application Control (Host to Device)

Algorithm Description for Each Packet:

- The application enables an interrupt on RX\_BK\_RDY.
- When an interrupt on RX\_BK\_RDY is received, the application knows that UDPHS\_EPTSTAX register BYTE\_COUNT bytes have been received.
- The application reads the BYTE\_COUNT bytes from the endpoint.
- The application clears RX\_BK\_RDY.

**Note:** If the application does not know the size of the transfer, it may **not** be a good option to use AUTO\_VALID. Because if a zero-length-packet is received, the RX\_BK\_RDY is automatically cleared by the AUTO\_VALID hardware and if the endpoint interrupt is triggered, the software will not find its originating flag when reading the UDPHS\_EPTSTAx register.

Algorithm to Fill Several Packets:

- The application enables the interrupts of BUSY\_BANK and AUTO\_VALID.
- When a BUSY\_BANK interrupt is received, the application knows that all banks available for the endpoint have been filled. Thus, the application can read all banks available.

If the application doesn't know the size of the receive buffer, instead of using the BUSY\_BANK interrupt, the application must use RX\_BK\_RDY.

#### 41.4.8.12 Bulk OUT or Interrupt OUT: Sending a Buffer Using DMA (Host To Device)

To use the DMA setting, the AUTO\_VALID field is mandatory.

See [41.4.8.6 Bulk IN or Interrupt IN: Sending a Buffer Using DMA \(Device to Host\)](#) for more information.

DMA Configuration Example:

1. First program UDPHS\_DMAADDRESSx with the address of the buffer that should be transferred.
2. Enable the interrupt of the DMA in UDPHS\_IEN
3. Program the DMA Channelx Control Register:
  - Size of buffer to be sent.
  - END\_B\_EN: Can be used for OUT packet truncation (discarding of unbuffered packet data) at the end of DMA buffer.

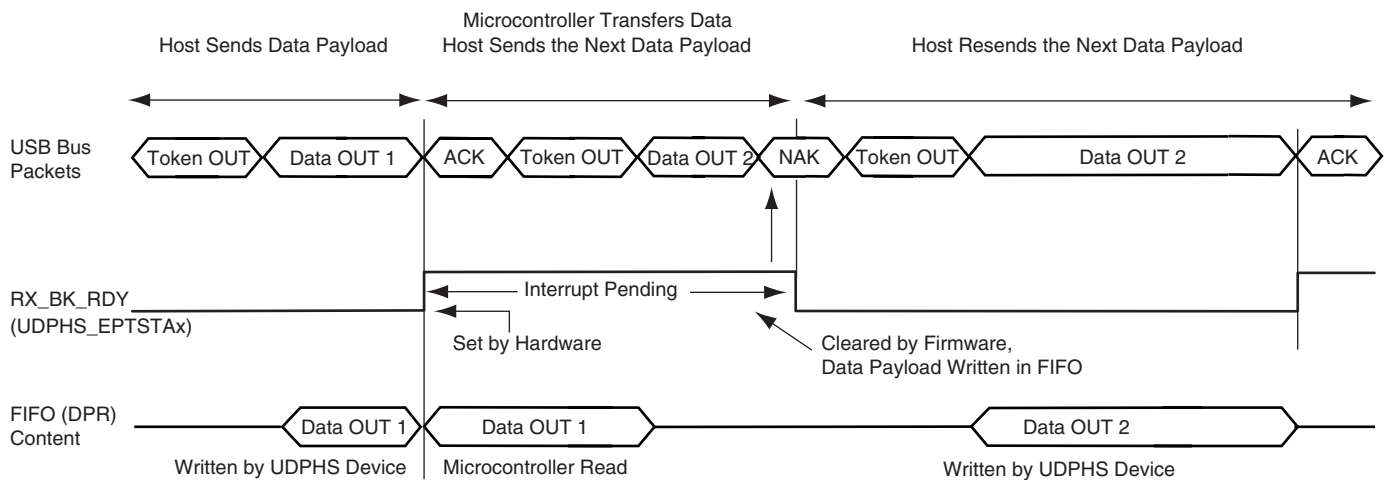
- END\_BUFFIT: Generate an interrupt when BUFF\_COUNT in the UDPHS\_DMASTATUSx register reaches 0.
- END\_TR\_EN: End of transfer enable, the UDPHS device can put an end to the current DMA transfer, in case of a short packet.
- END\_TR\_IT: End of transfer interrupt enable, an interrupt is sent after the last USB packet has been transferred by the DMA, if the USB transfer ended with a short packet. (Beneficial when the receive size is unknown.)
- CHANN\_ENB: Run and stop at end of buffer.

For OUT transfer, the bank will be automatically cleared by hardware when the application has read all the bytes in the bank (the bank is empty).

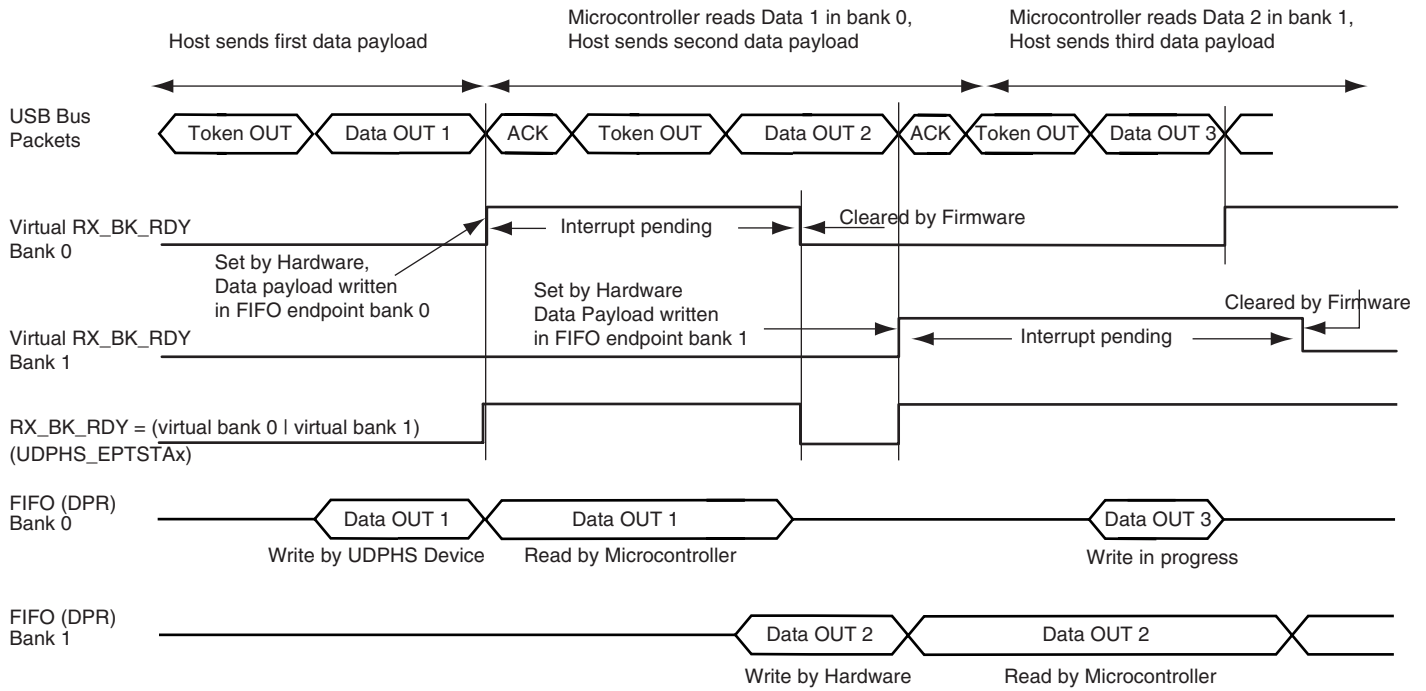
**Note:** When a zero-length-packet is received, RX\_BK\_RDY bit in UDPHS\_EPTSTAx is cleared automatically by AUTO\_VALID, and the application knows of the end of buffer by the presence of the END\_TR\_IT.

**Note:** If the host sends a zero-length packet, and the endpoint is free, then the device sends an ACK. No data is written in the endpoint, the RX\_BY\_RDY interrupt is generated, and the BYTE\_COUNT field in UDPHS\_EPTSTAx is null.

**Figure 41-12.** Data OUT Transfer for Endpoint with One Bank

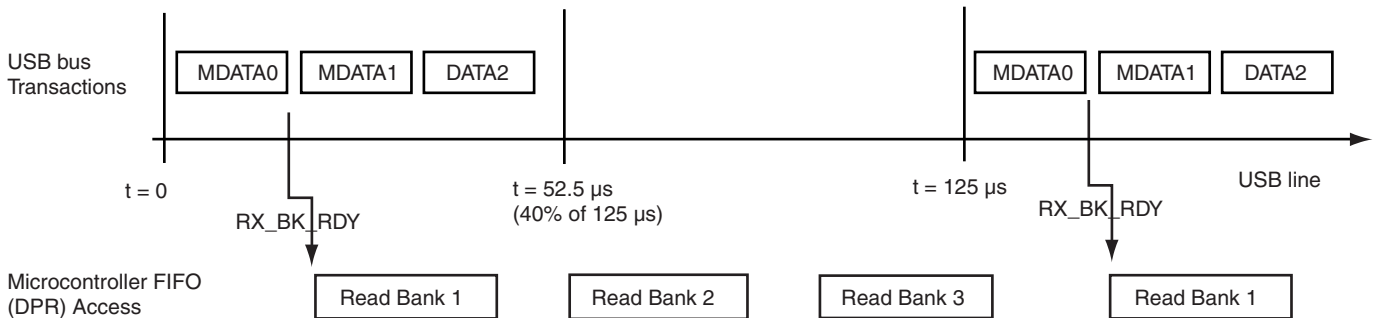


**Figure 41-13. Data OUT Transfer for an Endpoint with Two Banks**



#### 41.4.8.13 High Bandwidth Isochronous Endpoint OUT

**Figure 41-14. Bank Management, Example of Three Transactions per Microframe**



USB 2.0 supports individual High Speed isochronous endpoints that require data rates up to 192 Mb/s (24 MB/s): 3x1024 data bytes per microframe.

To support such a rate, two or three banks may be used to buffer the three consecutive data packets. The microcontroller (or the DMA) should be able to empty the banks very rapidly (at least 24 MB/s on average).

NB\_TRANS field in UDPHS\_EPTCFGx register = Number Of Transactions per Microframe.

If NB\_TRANS > 1 then it is High Bandwidth.

Example:

- If NB\_TRANS = 3, the sequence should be either
  - MData0
  - MData0/Data1
  - MData0/Data1/Data2
- If NB\_TRANS = 2, the sequence should be either
  - MData0
  - MData0/Data1
- If NB\_TRANS = 1, the sequence should be
  - Data0

#### 41.4.8.14 Isochronous Endpoint Handling: OUT Example

The user can ascertain the bank status (free or busy), and the toggle sequencing of the data packet for each bank with the UDPHS\_EPTSTAx register in the three bit fields as follows:

- TOGGLESQ\_STA: PID of the data stored in the current bank
- CURRENT\_BANK: Number of the bank currently being accessed by the microcontroller.
- BUSY\_BANK\_STA: Number of busy bank

This is particularly useful in case of a missing data packet.

If the inter-packet delay between the OUT token and the Data is greater than the USB standard, then the ISO-OUT transaction is ignored. (Payload data is not written, no interrupt is generated to the CPU.)

If there is a data CRC (Cyclic Redundancy Check) error, the payload is, none the less, written in the endpoint. The ERR\_CRISO flag is set in UDPHS\_EPTSTAx register.

If the endpoint is already full, the packet is not written in the DPRAM. The ERR\_FL\_ISO flag is set in UDPHS\_EPTSTAx.

If the payload data is greater than the maximum size of the endpoint, then the ERR\_OVFLW flag is set. It is the task of the CPU to manage this error. The data packet is written in the endpoint (except the extra data).

If the host sends a Zero Length Packet, and the endpoint is free, no data is written in the endpoint, the RX\_BK\_RDY flag is set, and the BYTE\_COUNT field in UDPHS\_EPTSTAx register is null.

The FRCESTALL command bit is unused for an isochronous endpoint.

Otherwise, payload data is written in the endpoint, the RX\_BK\_RDY interrupt is generated and the BYTE\_COUNT in UDPHS\_EPTSTAx register is updated.



## 41.4.8.15 STALL

STALL is returned by a function in response to an IN token or after the data phase of an OUT or in response to a PING transaction. STALL indicates that a function is unable to transmit or receive data, or that a control pipe request is not supported.

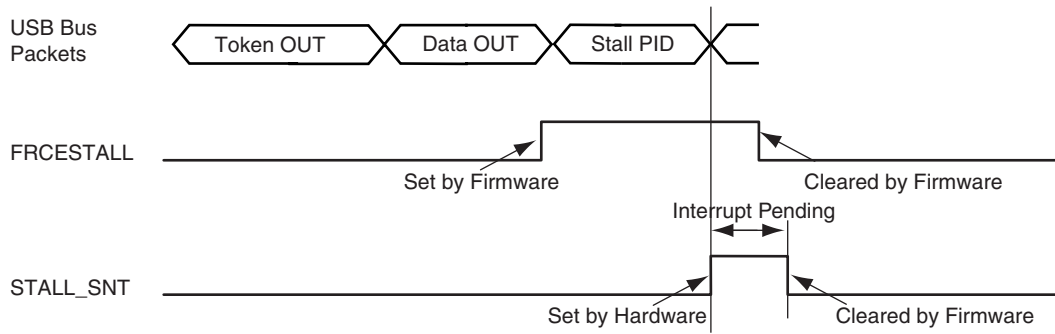
- OUT

To stall an endpoint, set the FRCESTALL bit in UDPHS\_EPTSETSTAx register and after the STALL\_SNT flag has been set, set the TOGGLE\_SEG bit in the UDPHS\_EPTCLRSTAx register.

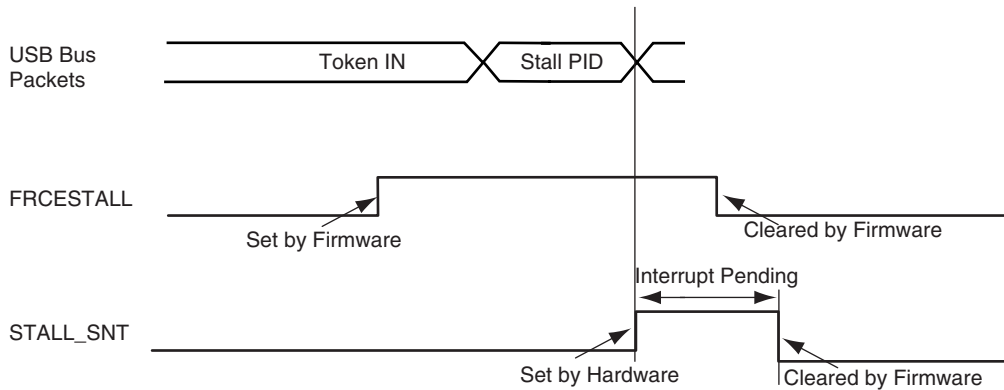
- IN

Set the FRCESTALL bit in UDPHS\_EPTSETSTAx register.

**Figure 41-15.** Stall Handshake Data OUT Transfer



**Figure 41-16.** Stall Handshake Data IN Transfer



#### 41.4.9 Speed Identification

The high speed reset is managed by the hardware.

At the connection, the host makes a reset which could be a classic reset (full speed) or a high speed reset.

At the end of the reset process (full or high), the ENDRESET interrupt is generated.

Then the CPU should read the SPEED bit in UDPHS\_INTSTAx to ascertain the speed mode of the device.

#### 41.4.10 USB V2.0 High Speed Global Interrupt

Interrupts are defined in [Section 41.5.3 "UDPHS Interrupt Enable Register"](#) (UDPHS\_IEN) and in [Section 41.5.4 "UDPHS Interrupt Status Register"](#) (UDPHS\_INTSTA).

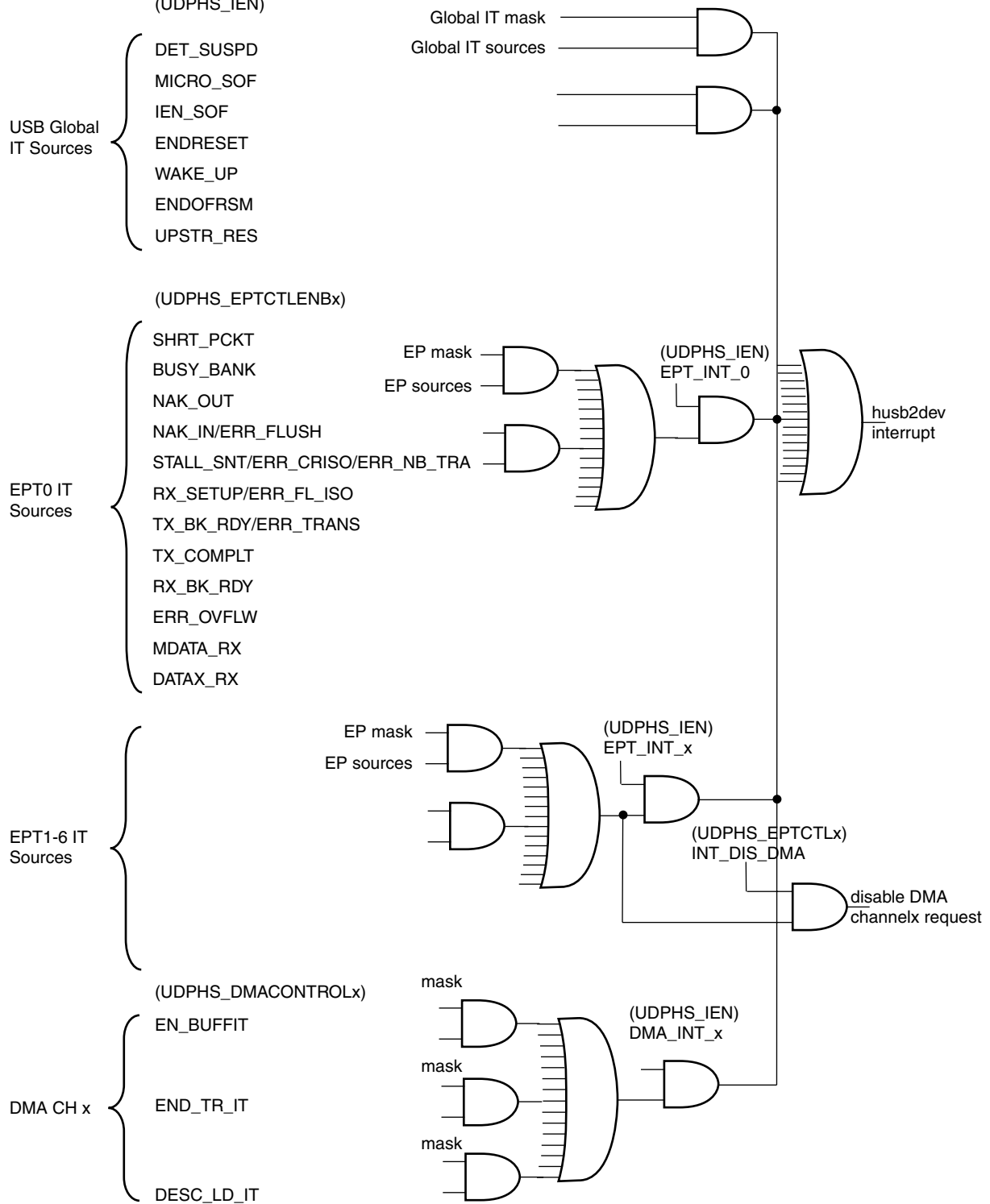
#### 41.4.11 Endpoint Interrupts

Interrupts are enabled in UDPHS\_IEN (see [Section 41.5.3 "UDPHS Interrupt Enable Register"](#)) and individually masked in UDPHS\_EPTCTLENBx (see [Section 41.5.9 "UDPHS Endpoint Control Enable Register"](#)).

**Table 41-4.** Endpoint Interrupt Source Masks

SHRT_PCKT	Short Packet Interrupt
BUSY_BANK	Busy Bank Interrupt
NAK_OUT	NAKOUT Interrupt
NAK_IN/ERR_FLUSH	NAKIN/Error Flush Interrupt
STALL_SNT/ERR_CRISO/ERR_NB_TRA	Stall Sent/CRC error/Number of Transaction Error Interrupt
RX_SETUP/ERR_FL_ISO	Received SETUP/Error Flow Interrupt
TX_PK_RD /ERR_TRANS	TX Packet Read/Transaction Error Interrupt
TX_COMPLT	Transmitted IN Data Complete Interrupt
RX_BK_RDY	Received OUT Data Interrupt
ERR_OVFLW	Overflow Error Interrupt
MDATA_RX	MDATA Interrupt
DATA_X_RX	DATAx Interrupt

**Figure 41-17. UDPHS Interrupt Control Interface**  
(UDPHS\_IEN)

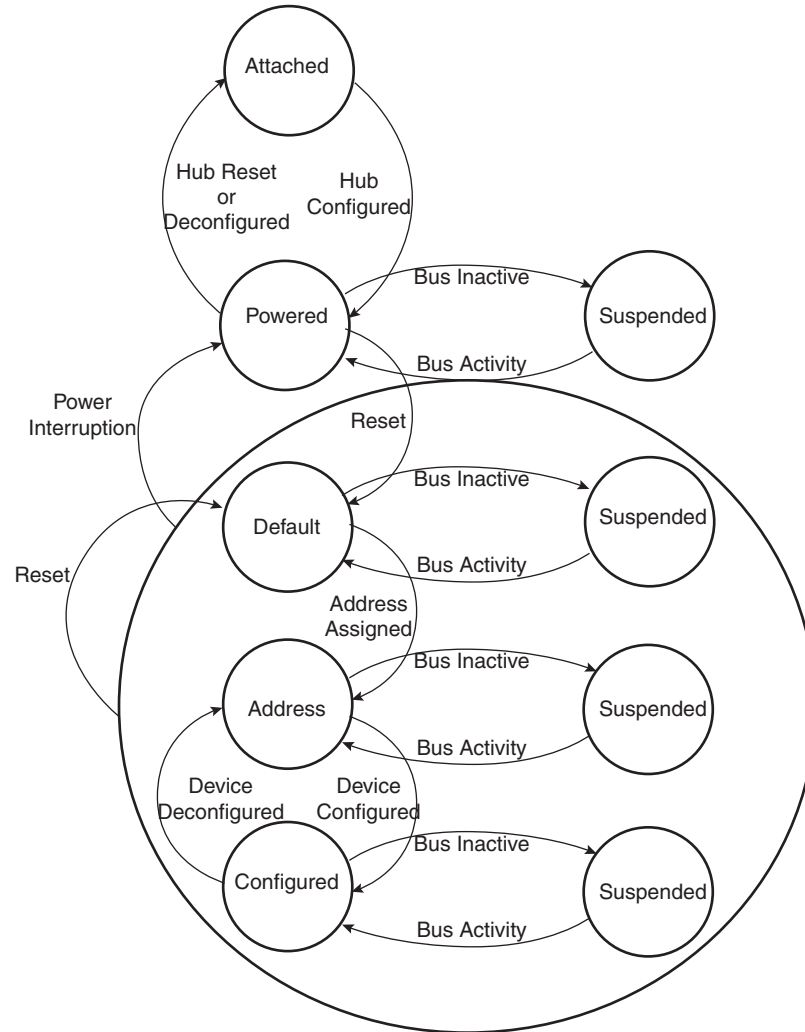


## 41.4.12 Power Modes

### 41.4.12.1 Controlling Device States

A USB device has several possible states. Refer to Chapter 9 (USB Device Framework) of the Universal Serial Bus Specification, Rev 2.0.

**Figure 41-18.** UDPHS Device State Diagram



Movement from one state to another depends on the USB bus state or on standard requests sent through control transactions via the default endpoint (endpoint 0).

After a period of bus inactivity, the 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  $\mu\text{A}$  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.

## 41.4.12.2 Not Powered State

Self powered devices can detect 5V VBUS using a PIO. When the device is not connected to a host, device power consumption can be reduced by the DETACH bit in UDPHS\_CTRL. Disabling the transceiver is automatically done. HSDM, HSDP, FSDP and FSDM lines are tied to GND pull-downs integrated in the hub downstream ports.

## 41.4.12.3 Entering Attached State

When no device is connected, the USB FSDP and FSDM signals are tied to GND by 15 K $\Omega$  pull-downs integrated in the hub downstream ports. When a device is attached to an hub downstream port, the device connects a 1.5 K $\Omega$  pull-up on FSDP. The USB bus line goes into IDLE state, FSDP is pulled-up by the device 1.5 K $\Omega$  resistor to 3.3V and FSDM is pulled-down by the 15 K $\Omega$  resistor to GND of the host.

After pull-up connection, the device enters the powered state. The transceiver remains disabled until bus activity is detected.

In case of low power consumption need, the device can be stopped. When the device detects the VBUS, the software must enable the USB transceiver by enabling the EN\_UDPHS bit in UDPHS\_CTRL register.

The software can detach the pull-up by setting DETACH bit in UDPHS\_CTRL register.

## 41.4.12.4 From Powered State to Default State (Reset)

After its connection to a USB host, the USB device waits for an end-of-bus reset. The unmasked flag ENDRESET is set in the UDPHS\_IEN register and an interrupt is triggered.

Once the ENDRESET interrupt has been triggered, the device enters Default State. In this state, the UDPHS software must:

- Enable the default endpoint, setting the EPT\_ENABL flag in the UDPHS\_EPTCTLENB[0] register and, optionally, enabling the interrupt for endpoint 0 by writing 1 in EPT\_INT\_0 of the UDPHS\_IEN 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.

In this state, the EN\_UDPHS bit in UDPHS\_CTRL register must be enabled.

## 41.4.12.5 From Default State to Address State (Address Assigned)

After a Set Address standard device request, the USB host peripheral enters the address state.

**Warning:** before the device enters address state, it must achieve the Status IN transaction of the control transfer, i.e., the UDPHS device sets its new address once the TX\_COMPLT flag in the UDPHS\_EPTCTL[0] register has been received and cleared.

To move to address state, the driver software sets the DEV\_ADDR field and the FADDR\_EN flag in the UDPHS\_CTRL register.

## 41.4.12.6 From Address State to Configured State (Device Configured)

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 BK\_NUMBER, EPT\_TYPE, EPT\_DIR and EPT\_SIZE fields in the UDPHS\_EPTCFGx registers and enabling them by setting the EPT\_ENABL flag in the UDPHS\_EPTCTLENBx registers, and, optionally, enabling corresponding interrupts in the UDPHS\_IEN register.

#### 41.4.12.7 *Entering Suspend State (Bus Activity)*

When a Suspend (no bus activity on the USB bus) is detected, the DET\_SUSPD signal in the UDPHS\_STA register is set. This triggers an interrupt if the corresponding bit is set in the UDPHS\_IEN register. This flag is cleared by writing to the UDPHS\_CLRINT register. Then the device enters Suspend Mode.

In this state bus powered devices must drain less than 500  $\mu$ A 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 UDPHS device peripheral clocks can be switched off. Resume event is asynchronously detected.

#### 41.4.12.8 *Receiving a Host Resume*

In Suspend mode, a resume event on the USB bus line is detected asynchronously, transceiver and clocks disabled (however the pull-up should not be removed).

Once the resume is detected on the bus, the signal WAKE\_UP in the UDPHS\_INTSTA is set. It may generate an interrupt if the corresponding bit in the UDPHS\_IEN register is set. This interrupt may be used to wake-up the core, enable PLL and main oscillators and configure clocks.

#### 41.4.12.9 *Sending an External Resume*

In Suspend State it is possible to wake-up the host by sending an external resume.

The device waits at least 5 ms after being entered in Suspend State before sending an external resume.

The device must force a K state from 1 to 15 ms to resume the host.

## 41.4.13 Test Mode

A device must support the TEST\_MODE feature when in the Default, Address or Configured High Speed device states.

TEST\_MODE can be:

- Test\_J
- Test\_K
- Test\_Packet
- Test\_SEO\_NAK

(See [Section 41.5.7 “UDPHS Test Register”](#) on page 787 for definitions of each test mode.)

```
const char test_packet_buffer[] = {
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // JKJKJKJK *
    9
    0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, // JJKKJJKK *
    8
    0xEE, 0xEE, 0xEE, 0xEE, 0xEE, 0xEE, 0xEE, 0xEE, // JJKKJJKK *
    8
    0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, //
    JJJJJJJKKKKKKK * 8
    0x7F, 0xBF, 0xDF, 0xEF, 0xF7, 0xFB, 0xFD, // JJJJJJJK * 8
    0xFC, 0x7E, 0xBF, 0xDF, 0xEF, 0xF7, 0xFB, 0xFD, 0x7E // {JKKKKKKK
    * 10}, JK
};
```

## 41.5 USB High Speed Device Port (UDPHS) User Interface

**Table 41-5.** Register Mapping

Offset	Register	Name	Access	Reset
0x00	<a href="#">UDPHS Control Register</a>	UDPHS_CTRL	Read/Write	0x0000_0200
0x04	<a href="#">UDPHS Frame Number Register</a>	UDPHS_FNUM	Read	0x0000_0000
0x08 - 0x0C	Reserved	–	–	–
0x10	<a href="#">UDPHS Interrupt Enable Register</a>	UDPHS_IEN	Read/Write	0x0000_0010
0x14	<a href="#">UDPHS Interrupt Status Register</a>	UDPHS_INTSTA	Read	0x0000_0000
0x18	<a href="#">UDPHS Clear Interrupt Register</a>	UDPHS_CLRINT	Write	–
0x1C	<a href="#">UDPHS Endpoints Reset Register</a>	UDPHS_EPTRST	Write	–
0x20 - 0xCC	Reserved	–	–	–
0xE0	<a href="#">UDPHS Test Register</a>	UDPHS_TST	Read/Write	0x0000_0000
0xE4 - 0xE8	Reserved	–	–	–
0x100	UDPHS Endpoint0 Configuration Register	UDPHS_EPTCFG0	Read/Write	0x0000_0000
0x104	UDPHS Endpoint0 Control Enable Register	UDPHS_EPTCTLENB0	Write	–
0x108	UDPHS Endpoint0 Control Disable Register	UDPHS_EPTCTLDIS0	Write	–
0x10C	UDPHS Endpoint0 Control Register	UDPHS_EPTCTL0	Read	0x0000_0000 <sup>(1)</sup>
0x110	Reserved (for endpoint 0)	–	–	–
0x114	UDPHS Endpoint0 Set Status Register	UDPHS_EPTSETSTA0	Write	–
0x118	UDPHS Endpoint0 Clear Status Register	UDPHS_EPTCLRSTA0	Write	–
0x11C	UDPHS Endpoint0 Status Register	UDPHS_EPTSTA0	Read	0x0000_0040
0x120 - 0x1DC	UDPHS Endpoint1 to 6 <sup>(2)</sup> Registers			
0x300 - 0x30C	Reserved	–	–	–
0x310	UDPHS DMA Next Descriptor1 Address Register	UDPHS_DMANXTDSC1	Read/Write	0x0000_0000
0x314	UDPHS DMA Channel1 Address Register	UDPHS_DMAADDRESS1	Read/Write	0x0000_0000
0x318	UDPHS DMA Channel1 Control Register	UDPHS_DMACONTROL1	Read/Write	0x0000_0000
0x31C	UDPHS DMA Channel1 Status Register	UDPHS_DMASTATUS1	Read/Write	0x0000_0000
0x320 - 0x370	DMA Channel2 to 5 <sup>(3)</sup> Registers			

- Notes:
1. The reset value for UDPHS\_EPTCTL0 is 0x0000\_0001.
  2. The addresses for the UDPHS Endpoint registers shown here are for UDPHS Endpoint0. The structure of this group of registers is repeated successively for each endpoint according to the consecution of endpoint registers located between 0x120 and 0x1DC.
  3. The addresses for the UDPHS DMA registers shown here are for UDPHS DMA Channel1. (There is no Channel0) The structure of this group of registers is repeated successively for each DMA channel according to the consecution of DMA registers located between 0x320 and 0x370.



## 41.5.1 UDPHS Control Register

**Name:** UDPHS\_CTRL

**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
–	–	–	–	PULLD_DIS	REWAKEUP	DETACH	EN_UDPHS
7	6	5	4	3	2	1	0
FADDR_EN	DEV_ADDR						

- **DEV\_ADDR: UDPHS Address**

Read:

This field contains the default address (0) after power-up or UDPHS bus reset.

Write:

This field is written with the value set by a SET\_ADDRESS request received by the device firmware.

- **FADDR\_EN: Function Address Enable**

Read:

0 = Device is not in address state.

1 = Device is in address state.

Write:

0 = only the default function address is used (0).

1 = this bit is set by the device firmware after a successful status phase of a SET\_ADDRESS transaction. When set, the only address accepted by the UDPHS controller is the one stored in the UDPHS Address field. It will not be cleared afterwards by the device firmware. It is cleared by hardware on hardware reset, or when UDPHS bus reset is received (see above).

- **EN\_UDPHS: UDPHS Enable**

Read:

0 = UDPHS is disabled.

1 = UDPHS is enabled.

Write:

0 = disable and reset the UDPHS controller, disable the UDPHS transceiver.

1 = enables the UDPHS controller.

- **DETACH: Detach Command**

Read:



0 = UDPHS is attached.

1 = UDPHS is detached, UTMI transceiver is suspended.

Write:

0 = pull up the DP line (attach command).

1 = simulate a detach on the UDPHS line and force the UTMI transceiver into suspend state (Suspend M = 0).

- **REWAKEUP: Send Remote Wake Up**

Read:

0 = Remote Wake Up is disabled.

1 = Remote Wake Up is enabled.

Write:

0 = no effect.

1 = force an external interrupt on the UDPHS controller for Remote Wake UP purposes.

An Upstream Resume is sent only after the UDPHS bus has been in SUSPEND state for at least 5 ms.

This bit is automatically cleared by hardware at the end of the Upstream Resume.

- **PULLD\_DIS: Pull-Down Disable**

When set, there is no pull-down on DP & DM. (DM Pull-Down = DP Pull-Down = 0).

Note: If the DETACH bit is also set, device DP & DM are left in high impedance state.

## 41.5.2 UDPHS Frame Number Register

**Name:** UDPHS\_FNUM

**Access Type:** Read

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

- **MICRO\_FRAME\_NUM: Microframe Number**

Number of the received microframe (0 to 7) in one frame. This field is reset at the beginning of each new frame (1 ms).

One microframe is received each 125 microseconds (1 ms/8).

- **FRAME\_NUMBER: Frame Number as defined in the Packet Field Formats**

This field is provided in the last received SOF packet (see INT\_SOF in the [UDPHS Interrupt Status Register](#)).

- **FNUM\_ERR: Frame Number CRC Error**

This bit is set by hardware when a corrupted Frame Number in Start of Frame packet (or Micro SOF) is received.

This bit and the INT\_SOF (or MICRO\_SOF) interrupt are updated at the same time.

### 41.5.3 UDPHS Interrupt Enable Register

**Name:** UDPHS\_IEN

**Access Type:** Read/Write

31	30	29	28	27	26	25	24
–	DMA_6	DMA_5	DMA_4	DMA_3	DMA_2	DMA_1	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	EPT_6	EPT_5	EPT_4	EPT_3	EPT_2	EPT_1	EPT_0
7	6	5	4	3	2	1	0
UPSTR_RES	ENDOFRSM	WAKE_UP	ENDRESET	INT_SOF	MICRO_SOF	DET_SUSPD	–

• **DET\_SUSPD: Suspend Interrupt Enable**

Read:

0 = Suspend Interrupt is disabled.

1 = Suspend Interrupt is enabled.

Write

0 = disable Suspend Interrupt.

1 = enable Suspend Interrupt.

• **MICRO\_SOF: Micro-SOF Interrupt Enable**

Read:

0 = Micro-SOF Interrupt is disabled.

1 = Micro-SOF Interrupt is enabled.

Write

0 = disable Micro-SOF Interrupt.

1 = enable Micro-SOF Interrupt.

• **INT\_SOF: SOF Interrupt Enable**

Read:

0 = SOF Interrupt is disabled.

1 = SOF Interrupt is enabled.

Write

0 = disable SOF Interrupt.

1 = enable SOF Interrupt.

- **ENDRESET: End Of Reset Interrupt Enable**

Read:

0 = End Of Reset Interrupt is disabled.

1 = End Of Reset Interrupt is enabled.

Write

0 = disable End Of Reset Interrupt.

1 = enable End Of Reset Interrupt. Automatically enabled after USB reset.

- **WAKE\_UP: Wake Up CPU Interrupt Enable**

Read:

0 = Wake Up CPU Interrupt is disabled.

1 = Wake Up CPU Interrupt is enabled.

Write

0 = disable Wake Up CPU Interrupt.

1 = enable Wake Up CPU Interrupt.

- **ENDOFRESM: End Of Resume Interrupt Enable**

Read:

0 = Resume Interrupt is disabled.

1 = Resume Interrupt is enabled.

Write

0 = disable Resume Interrupt.

1 = enable Resume Interrupt.

- **UPSTR\_RES: Upstream Resume Interrupt Enable**

Read:

0 = Upstream Resume Interrupt is disabled.

1 = Upstream Resume Interrupt is enabled.

Write

0 = disable Upstream Resume Interrupt.

1 = enable Upstream Resume Interrupt.

- **EPT\_x: Endpoint x Interrupt Enable**

Read:

0 = the interrupts for this endpoint are disabled.

1 = the interrupts for this endpoint are enabled.

Write

0 = disable the interrupts for this endpoint.



1 = enable the interrupts for this endpoint.

- **DMA\_INT\_x: DMA Channel x Interrupt Enable**

Read:

0 = the interrupts for this channel are disabled.

1 = the interrupts for this channel are enabled.

Write

0 = disable the interrupts for this channel.

1 = enable the interrupts for this channel.

## 41.5.4 UDPHS Interrupt Status Register

**Name:** UDPHS\_INTSTA

**Access Type:** Read-only

31	30	29	28	27	26	25	24
–	DMA_6	DMA_5	DMA_4	DMA_3	DMA_2	DMA_1	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	EPT_6	EPT_5	EPT_4	EPT_3	EPT_2	EPT_1	EPT_0
7	6	5	4	3	2	1	0
UPSTR_RES	ENDOFRSM	WAKE_UP	ENDRESET	INT_SOF	MICRO_SOF	DET_SUSPD	SPEED

- **SPEED: Speed Status**

0 = reset by hardware when the hardware is in Full Speed mode.

1 = set by hardware when the hardware is in High Speed mode

- **DET\_SUSPD: Suspend Interrupt**

0 = cleared by setting the DET\_SUSPD bit in UDPHS\_CLRINT register

1 = set by hardware when a UDPHS Suspend (Idle bus for three frame periods, a J state for 3 ms) is detected. This triggers a UDPHS interrupt when the DET\_SUSPD bit is set in UDPHS\_IEN register.

- **MICRO\_SOF: Micro Start Of Frame Interrupt**

0 = cleared by setting the MICRO\_SOF bit in UDPHS\_CLRINT register.

1 = set by hardware when an UDPHS micro start of frame PID (SOF) has been detected (every 125 us) or synthesized by the macro. This triggers a UDPHS interrupt when the MICRO\_SOF bit is set in UDPHS\_IEN. In case of detected SOF, the MICRO\_FRAME\_NUM field in UDPHS\_FNUM register is incremented and the FRAME\_NUMBER field doesn't change.

Note: The Micro Start Of Frame Interrupt (MICRO\_SOF), and the Start Of Frame Interrupt (INT\_SOF) are not generated at the same time.

- **INT\_SOF: Start Of Frame Interrupt**

0 = cleared by setting the INT\_SOF bit in UDPHS\_CLRINT.

1 = set by hardware when an UDPHS Start Of Frame PID (SOF) has been detected (every 1 ms) or synthesized by the macro. This triggers a UDPHS interrupt when the INT\_SOF bit is set in UDPHS\_IEN register. In case of detected SOF, in High Speed mode, the MICRO\_FRAME\_NUMBER field is cleared in UDPHS\_FNUM register and the FRAME\_NUMBER field is updated.

- **ENDRESET: End Of Reset Interrupt**

0 = cleared by setting the ENDRESET bit in UDPHS\_CLRINT.

1 = set by hardware when an End Of Reset has been detected by the UDPHS controller. This triggers a UDPHS interrupt when the ENDRESET bit is set in UDPHS\_IEN.

- **WAKE\_UP: Wake Up CPU Interrupt**

0 = cleared by setting the WAKE\_UP bit in UDPHS\_CLRINT.

1 = set by hardware when the UDPHS controller is in SUSPEND state and is re-activated by a filtered non-idle signal from the UDPHS line (not by an upstream resume). This triggers a UDPHS interrupt when the WAKE\_UP bit is set in UDPHS\_IEN register. When receiving this interrupt, the user has to enable the device controller clock prior to operation.

Note: this interrupt is generated even if the device controller clock is disabled.

- **ENDOFRSM: End Of Resume Interrupt**

0 = cleared by setting the ENDOFRSM bit in UDPHS\_CLRINT.

1 = set by hardware when the UDPHS controller detects a good end of resume signal initiated by the host. This triggers a UDPHS interrupt when the ENDOFRSM bit is set in UDPHS\_IEN.

- **UPSTR\_RES: Upstream Resume Interrupt**

0 = cleared by setting the UPSTR\_RES bit in UDPHS\_CLRINT.

1 = set by hardware when the UDPHS controller is sending a resume signal called “upstream resume”. This triggers a UDPHS interrupt when the UPSTR\_RES bit is set in UDPHS\_IEN.

- **EPT\_x: Endpoint x Interrupt**

0 = reset when the UDPHS\_EPTSTAx interrupt source is cleared.

1 = set by hardware when an interrupt is triggered by the UDPHS\_EPTSTAx register and this endpoint interrupt is enabled by the EPT\_INT\_x bit in UDPHS\_IEN.

- **DMA\_INT\_x: DMA Channel x Interrupt**

0 = reset when the UDPHS\_DMASTATUSx interrupt source is cleared.

1 = set by hardware when an interrupt is triggered by the DMA Channelx and this endpoint interrupt is enabled by the DMA\_INT\_x bit in UDPHS\_IEN.



## 41.5.5 UDPHS Clear Interrupt Register

**Name:** UDPHS\_CLRINT

**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
UPSTR_RES	ENDOFRSM	WAKE_UP	ENDRESET	INT_SOF	MICRO_SOF	DET_SUSPD	–

- **DET\_SUSPD: Suspend Interrupt Clear**

0 = no effect.

1 = clear the DET\_SUSPD bit in UDPHS\_INTSTA.

- **MICRO\_SOF: Micro Start Of Frame Interrupt Clear**

0 = no effect.

1 = clear the MICRO\_SOF bit in UDPHS\_INTSTA.

- **INT\_SOF: Start Of Frame Interrupt Clear**

0 = no effect.

1 = clear the INT\_SOF bit in UDPHS\_INTSTA.

- **ENDRESET: End Of Reset Interrupt Clear**

0 = no effect.

1 = clear the ENDRESET bit in UDPHS\_INTSTA.

- **WAKE\_UP: Wake Up CPU Interrupt Clear**

0 = no effect.

1 = clear the WAKE\_UP bit in UDPHS\_INTSTA.

- **ENDOFRSM: End Of Resume Interrupt Clear**

0 = no effect.

1 = clear the ENDOFRSM bit in UDPHS\_INTSTA.

- **UPSTR\_RES: Upstream Resume Interrupt Clear**

0 = no effect.

1 = clear the UPSTR\_RES bit in UDPHS\_INTSTA.

### 41.5.6 UDPHS Endpoints Reset Register

**Name:** UDPHS\_EPTRST

**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
–	EPT_6	EPT_5	EPT_4	EPT_3	EPT_2	EPT_1	EPT_0

- **EPT\_x: Endpoint x Reset**

0 = no effect.

1 = reset the Endpointx state.

Setting this bit clears the Endpoint status UDPHS\_EPTSTAx register, except for the TOGGLESQ\_STA field.

## 41.5.7 UDPHS Test Register

**Name:** UDPHS\_TST

**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
–	–	OPMODE2	TST_PKT	TST_K	TST_J	SPEED_CFG	

- **SPEED\_CFG: Speed Configuration**

Read/Write:

Speed Configuration:

00	Normal Mode: The macro is in Full Speed mode, ready to make a High Speed identification, if the host supports it and then to automatically switch to High Speed mode
01	Reserved
10	Force High Speed: Set this value to force the hardware to work in High Speed mode. Only for debug or test purpose.
11	Force Full Speed: Set this value to force the hardware to work only in Full Speed mode. In this configuration, the macro will not respond to a High Speed reset handshake

- **TST\_J: Test J Mode**

Read and write:

0 = no effect.

1 = set to send the J state on the UDPHS line. This enables the testing of the high output drive level on the D+ line.

- **TST\_K: Test K Mode**

Read and write:

0 = no effect.

1 = set to send the K state on the UDPHS line. This enables the testing of the high output drive level on the D- line.

- **TST\_PKT: Test Packet Mode**

Read and write:

0 = no effect.

1 = set to repetitively transmit the packet stored in the current bank. This enables the testing of rise and fall times, eye patterns, jitter, and any other dynamic waveform specifications.

- **OPMODE2: OpMode2**

Read and write:





0 = no effect.

1 = set to force the OpMode signal (UTMI interface) to “10”, to disable the bit-stuffing and the NRZI encoding.

Note: For the Test mode, Test\_SE0\_NAK (see Universal Serial Bus Specification, Revision 2.0: 7.1.20, Test Mode Support). Force the device in High Speed mode, and configure a bulk-type endpoint. Do not fill this endpoint for sending NAK to the host.

Upon command, a port’s transceiver must enter the High Speed receive mode and remain in that mode until the exit action is taken. This enables the testing of output impedance, low level output voltage and loading characteristics. In addition, while in this mode, upstream facing ports (and only upstream facing ports) must respond to any IN token packet with a NAK handshake (only if the packet CRC is determined to be correct) within the normal allowed device response time. This enables testing of the device squelch level circuitry and, additionally, provides a general purpose stimulus/response test for basic functional testing.

## 41.5.8 UDPHS Endpoint Configuration Register

**Name:** UDPHS\_EPTCFGx [x=0..6]

**Access Type:** Read/Write

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

- **EPT\_SIZE: Endpoint Size**

Read and write:

Set this field according to the endpoint size in bytes (see [Section 41.4.5 "Endpoint Configuration"](#)).

Endpoint Size

000	8 bytes
001	16 bytes
010	32 bytes
011	64 bytes
100	128 bytes
101	256 bytes
110	512 bytes
111	1024 bytes <sup>(1)</sup>

Note: 1. 1024 bytes is only for isochronous endpoint.

- **EPT\_DIR: Endpoint Direction**

Read and write:

0 = Clear this bit to configure OUT direction for Bulk, Interrupt and Isochronous endpoints.

1 = set this bit to configure IN direction for Bulk, Interrupt and Isochronous endpoints.

For Control endpoints this bit has no effect and should be left at zero.

- **EPT\_TYPE: Endpoint Type**

Read and write:

Set this field according to the endpoint type (see [Section 41.4.5 "Endpoint Configuration"](#)).

(Endpoint 0 should always be configured as control)

:Endpoint Type

00	Control endpoint
01	Isochronous endpoint
10	Bulk endpoint
11	Interrupt endpoint

- **BK\_NUMBER: Number of Banks**

Read and write:

Set this field according to the endpoint's number of banks (see [Section 41.4.5 "Endpoint Configuration"](#)).

Number of Banks

00	Zero bank, the endpoint is not mapped in memory
01	One bank (bank 0)
10	Double bank (Ping-Pong: bank 0/bank 1)
11	Triple bank (bank 0/bank 1/bank 2)

- **NB\_TRANS: Number Of Transaction per Microframe**

Read and Write:

The Number of transactions per microframe is set by software.

Note: Meaningful for high bandwidth isochronous endpoint only.

- **EPT\_MAPD: Endpoint Mapped**

Read-only:

0 = the user should reprogram the register with correct values.

1 = set by hardware when the endpoint size (EPT\_SIZE) and the number of banks (BK\_NUMBER) are correct regarding:

- the fifo max capacity (FIFO\_MAX\_SIZE in UDPHS\_IPFEATURES register)
- the number of endpoints/banks already allocated
- the number of allowed banks for this endpoint

## 41.5.9 UDPHS Endpoint Control Enable Register

**Name:** UDPHS\_EPTCTLENBx [x=0..6]

**Access Type:** Write-only

31	30	29	28	27	26	25	24
SHRT_PCKT	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	BUSY_BANK	–	–
15	14	13	12	11	10	9	8
NAK_OUT	NAK_IN/ ERR_FLUSH	STALL_SNT/ ERR_CRISO/ ERR_NBTRA	RX_SETUP/ ERR_FL_ISO	TX_PK_RDY/ ERR_TRANS	TX_COMPLT	RX_BK_RDY	ERR_OVFLW
7	6	5	4	3	2	1	0
MDATA_RX	DATA_X_RX	–	NYET_DIS	INTDIS_DMA	–	AUTO_VALID	EPT_ENABL

For additional Information, see [“UDPHS Endpoint Control Register” on page 795](#).

- **EPT\_ENABL: Endpoint Enable**

0 = no effect.

1 = enable endpoint according to the device configuration.

- **AUTO\_VALID: Packet Auto-Valid Enable**

0 = no effect.

1 = enable this bit to automatically validate the current packet and switch to the next bank for both IN and OUT transfers.

- **INTDIS\_DMA: Interrupts Disable DMA**

0 = no effect.

1 = If set, when an enabled endpoint-originated interrupt is triggered, the DMA request is disabled.

- **NYET\_DIS: NYET Disable (Only for High Speed Bulk OUT endpoints)**

0 = no effect.

1 = forces an ACK response to the next High Speed Bulk OUT transfer instead of a NYET response.

- **DATA\_X\_RX: DATAx Interrupt Enable (Only for high bandwidth Isochronous OUT endpoints)**

0 = no effect.

1 = enable DATAx Interrupt.

- **MDATA\_RX: MDATA Interrupt Enable (Only for high bandwidth Isochronous OUT endpoints)**

0 = no effect.

1 = enable MDATA Interrupt.

- **ERR\_OVFLW: Overflow Error Interrupt Enable**

0 = no effect.

1 = enable Overflow Error Interrupt.

- **RX\_BK\_RDY: Received OUT Data Interrupt Enable**

0 = no effect.

1 = enable Received OUT Data Interrupt.

- **TX\_COMPLT: Transmitted IN Data Complete Interrupt Enable**

0 = no effect.

1 = enable Transmitted IN Data Complete Interrupt.

- **TX\_PK\_RDY/ERR\_TRANS: TX Packet Ready/Transaction Error Interrupt Enable**

0 = no effect.

1 = enable TX Packet Ready/Transaction Error Interrupt.

- **RX\_SETUP/ERR\_FL\_ISO: Received SETUP/Error Flow Interrupt Enable**

0 = no effect.

1 = enable RX\_SETUP/Error Flow ISO Interrupt.

- **STALL\_SNT/ERR\_CRISO/ERR\_NBTRA: Stall Sent /ISO CRC Error/Number of Transaction Error Interrupt Enable**

0 = no effect.

1 = enable Stall Sent/Error CRC ISO/Error Number of Transaction Interrupt.

- **NAK\_IN/ERR\_FLUSH: NAKIN/Bank Flush Error Interrupt Enable**

0 = no effect.

1 = enable NAKIN/Bank Flush Error Interrupt.

- **NAK\_OUT: NAKOUT Interrupt Enable**

0 = no effect.

1 = enable NAKOUT Interrupt.

- **BUSY\_BANK: Busy Bank Interrupt Enable**

0 = no effect.

1 = enable Busy Bank Interrupt.

- **SHRT\_PCKT: Short Packet Send/Short Packet Interrupt Enable**

For OUT endpoints:

0 = no effect.

1 = enable Short Packet Interrupt.

For IN endpoints:

Guarantees short packet at end of DMA Transfer if the UDPHS\_DMACONTROLx register END\_B\_EN and UDPHS\_EPTCTLx register AUTOVALID bits are also set.



## 41.5.10 UDPHS Endpoint Control Disable Register

**Name:** UDPHS\_EPTCTLDISx [x=0..6]

**Access Type:** Write-only

31	30	29	28	27	26	25	24
SHRT_PCKT	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	BUSY_BANK	–	–
15	14	13	12	11	10	9	8
NAK_OUT	NAK_IN/ ERR_FLUSH	STALL_SNT/ ERR_CRISO/ ERR_NBTRA	RX_SETUP/ ERR_FL_ISO	TX_PK_RDY/ ERR_TRANS	TX_COMPLT	RX_BK_RDY	ERR_OVFLW
7	6	5	4	3	2	1	0
MDATA_RX	DATA_RX	–	NYET_DIS	INTDIS_DMA	–	AUTO_VALID	EPT_DISABL

For additional Information, see [“UDPHS Endpoint Control Register” on page 795](#).

- **EPT\_DISABL: Endpoint Disable**

0 = no effect.

1 = disable endpoint.

- **AUTO\_VALID: Packet Auto-Valid Disable**

0 = no effect.

1 = disable this bit to not automatically validate the current packet.

- **INTDIS\_DMA: Interrupts Disable DMA**

0 = no effect.

1 = disable the “Interrupts Disable DMA”.

- **NYET\_DIS: NYET Enable (Only for High Speed Bulk OUT endpoints)**

0 = no effect.

1 = let the hardware handle the handshake response for the High Speed Bulk OUT transfer.

- **DATA\_RX: DATAx Interrupt Disable (Only for High Bandwidth Isochronous OUT endpoints)**

0 = no effect.

1 = disable DATAx Interrupt.

- **MDATA\_RX: MDATA Interrupt Disable (Only for High Bandwidth Isochronous OUT endpoints)**

0 = no effect.

1 = disable MDATA Interrupt.

- **ERR\_OVFLW: Overflow Error Interrupt Disable**

0 = no effect.

1 = disable Overflow Error Interrupt.

- **RX\_BK\_RDY: Received OUT Data Interrupt Disable**

0 = no effect.

1 = disable Received OUT Data Interrupt.

- **TX\_COMPLT: Transmitted IN Data Complete Interrupt Disable**

0 = no effect.

1 = disable Transmitted IN Data Complete Interrupt.

- **TX\_PK\_RDY/ERR\_TRANS: TX Packet Ready/Transaction Error Interrupt Disable**

0 = no effect.

1 = disable TX Packet Ready/Transaction Error Interrupt.

- **RX\_SETUP/ERR\_FL\_ISO: Received SETUP/Error Flow Interrupt Disable**

0 = no effect.

1 = disable RX\_SETUP/Error Flow ISO Interrupt.

- **STALL\_SNT/ERR\_CRISO/ERR\_NBTRA: Stall Sent/ISO CRC Error/Number of Transaction Error Interrupt Disable**

0 = no effect.

1 = disable Stall Sent/Error CRC ISO/Error Number of Transaction Interrupt.

- **NAK\_IN/ERR\_FLUSH: NAKIN/bank flush error Interrupt Disable**

0 = no effect.

1 = disable NAKIN/ Bank Flush Error Interrupt.

- **NAK\_OUT: NAKOUT Interrupt Disable**

0 = no effect.

1 = disable NAKOUT Interrupt.

- **BUSY\_BANK: Busy Bank Interrupt Disable**

0 = no effect.

1 = disable Busy Bank Interrupt.

- **SHRT\_PCKT: Short Packet Interrupt Disable**

For OUT endpoints:

0 = no effect.

1 = disable Short Packet Interrupt.

For IN endpoints:

Never automatically add a zero length packet at end of DMA transfer.

## 41.5.11 UDPHS Endpoint Control Register

**Name:** UDPHS\_EPTCTLx [x=0..6]

**Access Type:** Read-only

31	30	29	28	27	26	25	24
SHRT_PCKT	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	BUSY_BANK	–	–
15	14	13	12	11	10	9	8
NAK_OUT	NAK_IN/ ERR_FLUSH	STALL_SNT/ ERR_CRISO/ ERR_NBTRA	RX_SETUP/ ERR_FL_ISO	TX_PK_RDY/ ERR_TRANS	TX_COMPLT	RX_BK_RDY	ERR_OVFLW
7	6	5	4	3	2	1	0
MDATA_RX	DATA_RX	–	NYET_DIS	INTDIS_DMA	–	AUTO_VALID	EPT_ENABL

- **EPT\_ENABL: Endpoint Enable**

0 = If cleared, the endpoint is disabled according to the device configuration. Endpoint 0 should always be enabled after a hardware or UDPHS bus reset and participate in the device configuration.

1 = If set, the endpoint is enabled according to the device configuration.

- **AUTO\_VALID: Packet Auto-Valid Enabled (Not for CONTROL Endpoints)**

Set this bit to automatically validate the current packet and switch to the next bank for both IN and OUT endpoints.

**For IN Transfer:**

If this bit is set, then the UDPHS\_EPTSTAx register TX\_PK\_RDY bit is set automatically when the current bank is full and at the end of DMA buffer if the UDPHS\_DMACONTROLx register END\_B\_EN bit is set.

The user may still set the UDPHS\_EPTSTAx register TX\_PK\_RDY bit if the current bank is not full, unless the user wants to send a Zero Length Packet by software.

**For OUT Transfer:**

If this bit is set, then the UDPHS\_EPTSTAx register RX\_BK\_RDY bit is automatically reset for the current bank when the last packet byte has been read from the bank FIFO or at the end of DMA buffer if the UDPHS\_DMACONTROLx register END\_B\_EN bit is set. For example, to truncate a padded data packet when the actual data transfer size is reached.

The user may still clear the UDPHS\_EPTSTAx register RX\_BK\_RDY bit, for example, after completing a DMA buffer by software if UDPHS\_DMACONTROLx register END\_B\_EN bit was disabled or in order to cancel the read of the remaining data bank(s).

- **INTDIS\_DMA: Interrupt Disables DMA**

If set, when an enabled endpoint-originated interrupt is triggered, the DMA request is disabled regardless of the UDPHS\_IEN register EPT\_INT\_x bit for this endpoint. Then, the firmware will have to clear or disable the interrupt source or clear this bit if transfer completion is needed.

If the exception raised is associated with the new system bank packet, then the previous DMA packet transfer is normally completed, but the new DMA packet transfer is not started (not requested).

If the exception raised is not associated to a new system bank packet (NAK\_IN, NAK\_OUT, ERR\_FL\_ISO...), then the request cancellation may happen at any time and may immediately stop the current DMA transfer.

This may be used, for example, to identify or prevent an erroneous packet to be transferred into a buffer or to complete a DMA buffer by software after reception of a short packet, or to perform buffer truncation on ERR\_FL\_ISO interrupt for adaptive rate.

- **NYET\_DIS: NYET Disable (Only for High Speed Bulk OUT endpoints)**

0 = If clear, this bit lets the hardware handle the handshake response for the High Speed Bulk OUT transfer.

1 = If set, this bit forces an ACK response to the next High Speed Bulk OUT transfer instead of a NYET response.

Note: According to the *Universal Serial Bus Specification, Rev 2.0* (8.5.1.1 NAK Responses to OUT/DATA During PING Protocol), a NAK response to an HS Bulk OUT transfer is expected to be an unusual occurrence.

- **DATA\_RX: DATAx Interrupt Enabled (Only for High Bandwidth Isochronous OUT endpoints)**

0 = no effect.

1 = send an interrupt when a DATA2, DATA1 or DATA0 packet has been received meaning the whole microframe data payload has been received.

- **MDATA\_RX: MDATA Interrupt Enabled (Only for High Bandwidth Isochronous OUT endpoints)**

0 = no effect.

1 = send an interrupt when an MDATA packet has been received and so at least one packet of the microframe data payload has been received.

- **ERR\_OVFLW: Overflow Error Interrupt Enabled**

0 = Overflow Error Interrupt is masked.

1 = Overflow Error Interrupt is enabled.

- **RX\_BK\_RDY: Received OUT Data Interrupt Enabled**

0 = Received OUT Data Interrupt is masked.

1 = Received OUT Data Interrupt is enabled.

- **TX\_COMPLT: Transmitted IN Data Complete Interrupt Enabled**

0 = Transmitted IN Data Complete Interrupt is masked.

1 = Transmitted IN Data Complete Interrupt is enabled.

- **TX\_PK\_RDY/ERR\_TRANS: TX Packet Ready/Transaction Error Interrupt Enabled**

0 = TX Packet Ready/Transaction Error Interrupt is masked.

1 = TX Packet Ready/Transaction Error Interrupt is enabled.

**Caution:** Interrupt source is active as long as the corresponding UDPHS\_EPTSTAx register TX\_PK\_RDY flag remains low. If there are no more banks available for transmitting after the software has set UDPHS\_EPTSTAx/TX\_PK\_RDY for the last transmit packet, then the interrupt source remains inactive until the first bank becomes free again to transmit at UDPHS\_EPTSTAx/TX\_PK\_RDY hardware clear.

- **RX\_SETUP/ERR\_FL\_ISO: Received SETUP/Error Flow Interrupt Enabled**

0 = Received SETUP/Error Flow Interrupt is masked.

1 = Received SETUP/Error Flow Interrupt is enabled.

- **STALL\_SNT/ERR\_CRISO/ERR\_NBTRA: Stall Sent/ISO CRC Error/Number of Transaction Error Interrupt Enabled**

0 = Stall Sent/ISO CRC error/number of Transaction Error Interrupt is masked.

1 = Stall Sent /ISO CRC error/number of Transaction Error Interrupt is enabled.

- **NAK\_IN/ERR\_FLUSH: NAKIN/Bank Flush Error Interrupt Enabled**

0 = NAKIN Interrupt is masked.

1 = NAKIN/Bank Flush Error Interrupt is enabled.

- **NAK\_OUT: NAKOUT Interrupt Enabled**

0 = NAKOUT Interrupt is masked.

1 = NAKOUT Interrupt is enabled.

- **BUSY\_BANK: Busy Bank Interrupt Enabled**

0 = BUSY\_BANK Interrupt is masked.

1 = BUSY\_BANK Interrupt is enabled.

**For OUT endpoints:** an interrupt is sent when all banks are busy.

For IN endpoints: an interrupt is sent when all banks are free.

- **SHRT\_PCKT: Short Packet Interrupt Enabled**

**For OUT endpoints:** send an Interrupt when a Short Packet has been received.

0 = Short Packet Interrupt is masked.

1 = Short Packet Interrupt is enabled.

**For IN endpoints:** a Short Packet transmission is guaranteed upon end of the DMA Transfer, thus signaling a BULK or INTERRUPT end of transfer or an end of isochronous (micro-)frame data, but only if the UDPHS\_DMACONTROLx register END\_B\_EN and UDPHS\_EPTCTLx register AUTO\_VALID bits are also set.

### 41.5.12 UDPHS Endpoint Set Status Register

**Name:** UDPHS\_EPTSETSTAx [x=0..6]

**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
–	–	–	–	TX_PK_RDY	–	KILL_BANK	–
7	6	5	4	3	2	1	0
–	–	FRCESTALL	–	–	–	–	–

• **FRCESTALL: Stall Handshake Request Set**

0 = no effect.

1 = set this bit to request a STALL answer to the host for the next handshake

Refer to chapters 8.4.5 (Handshake Packets) and 9.4.5 (Get Status) of the *Universal Serial Bus Specification, Rev 2.0* for more information on the STALL handshake.

• **KILL\_BANK: KILL Bank Set (for IN Endpoint)**

0 = no effect.

1 = kill the last written bank.

• **TX\_PK\_RDY: TX Packet Ready Set**

0 = no effect.

1 = set this bit after a packet has been written into the endpoint FIFO for IN data transfers

- 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 TX\_PK\_RDY is cleared.
- Transfer to the FIFO is done by writing in the “Buffer Address” register.
- Once the data payload has been transferred to the FIFO, the firmware notifies the UDPHS device setting TX\_PK\_RDY to one.
- UDPHS bus transactions can start.
- TXCOMP is set once the data payload has been received by the host.
- Data should be written into the endpoint FIFO only after this bit has been cleared.
- Set this bit without writing data to the endpoint FIFO to send a Zero Length Packet.

## 41.5.13 UDPHS Endpoint Clear Status Register

**Name:** UDPHS\_EPTCLRSTAx [x=0..6]

**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
NAK_OUT	NAK_IN/ ERR_FLUSH	STALL_SNT/ ERR_NBTRA	RX_SETUP/ ERR_FL_ISO	–	TX_COMPLT	RX_BK_RDY	–
7	6	5	4	3	2	1	0
–	TOGGLESQ	FRCESTALL	–	–	–	–	–

- **FRCESTALL: Stall Handshake Request Clear**

0 = no effect.

1 = clear the STALL request. The next packets from host will not be STALLED.

- **TOGGLESQ: Data Toggle Clear**

0 = no effect.

1 = clear the PID data of the current bank

For OUT endpoints, the next received packet should be a DATA0.

For IN endpoints, the next packet will be sent with a DATA0 PID.

- **RX\_BK\_RDY: Received OUT Data Clear**

0 = no effect.

1 = clear the RX\_BK\_RDY flag of UDPHS\_EPTSTAx.

- **TX\_COMPLT: Transmitted IN Data Complete Clear**

0 = no effect.

1 = clear the TX\_COMPLT flag of UDPHS\_EPTSTAx.

- **RX\_SETUP/ERR\_FL\_ISO: Received SETUP/Error Flow Clear**

0 = no effect.

1 = clear the RX\_SETUP/ERR\_FL\_ISO flags of UDPHS\_EPTSTAx.

- **STALL\_SNT/ERR\_NBTRA: Stall Sent/Number of Transaction Error Clear**

0 = no effect.

1 = clear the STALL\_SNT/ERR\_NBTRA flags of UDPHS\_EPTSTAx.

- **NAK\_IN/ERR\_FLUSH: NAKIN/Bank Flush Error Clear**

0 = no effect.

1 = clear the NAK\_IN/ERR\_FLUSH flags of UDPHS\_EPTSTAx.



- **NAK\_OUT: NAKOUT Clear**

0 = no effect.

1 = clear the NAK\_OUT flag of UDPHS\_EPTSTAx.



## 41.5.14 UDPHS Endpoint Status Register

**Name:** UDPHS\_EPTSTAx [x=0..6]

**Access Type:** Read-only

31	30	29	28	27	26	25	24
SHRT_PCKT		BYTE_COUNT					
23	22	21	20	19	18	17	16
BYTE_COUNT				BUSY_BANK_STA		CURRENT_BANK/ CONTROL_DIR	
15	14	13	12	11	10	9	8
NAK_OUT	NAK_IN/ ERR_FLUSH	STALL_SNT/ ERR_CRISO/ ERR_NBTRA	RX_SETUP/ ERR_FL_ISO	TX_PK_RDY/ ERR_TRANS	TX_COMPLT	RX_BK_RDY/ KILL_BANK	ERR_OVFLW
7	6	5	4	3	2	1	0
TOGGLESQ_STA		FRCESTALL	–	–	–	–	–

- **FRCESTALL: Stall Handshake Request**

0 = no effect.

1= If set a STALL answer will be done to the host for the next handshake.

This bit is reset by hardware upon received SETUP.

- **TOGGLESQ\_STA: Toggle Sequencing**

Toggle Sequencing:

**IN endpoint:** it indicates the PID Data Toggle that will be used for the next packet sent. This is not relative to the current bank.

**CONTROL and OUT endpoint:**

These bits are set by hardware to indicate the PID data of the current bank:

00	Data0
01	Data1
10	Data2 (only for High Bandwidth Isochronous Endpoint)
11	MData (only for High Bandwidth Isochronous Endpoint)

**Note 1:** In OUT transfer, the Toggle information is meaningful only when the current bank is busy (Received OUT Data = 1).

**Note 2:** These bits are updated for OUT transfer:

- a new data has been written into the current bank.
- the user has just cleared the Received OUT Data bit to switch to the next bank.

**Note 3:** For High Bandwidth Isochronous Out endpoint, it is recommended to check the UDPHS\_EPTSTAx/ERR\_TRANS bit to know if the toggle sequencing is correct or not.

**Note 4:** This field is reset to DATA1 by the UDPHS\_EPTCLRSTAx register TOGGLESQ bit, and by UDPHS\_EPTCTLDISx (disable endpoint).

- **ERR\_OVFLW: Overflow Error**

This bit is set by hardware when a new too-long packet is received.

Example: If the user programs an endpoint 64 bytes wide and the host sends 128 bytes in an OUT transfer, then the Overflow Error bit is set.

This bit is updated at the same time as the BYTE\_COUNT field.

This bit is reset by UDPHS\_EPTRST register EPT\_x (reset endpoint) and by UDPHS\_EPTCTLDISx (disable endpoint).

- **RX\_BK\_RDY/KILL\_BANK: Received OUT Data/KILL Bank**

- **Received OUT Data:** (For OUT endpoint or Control endpoint)

This bit is set by hardware after a new packet has been stored in the endpoint FIFO.

This bit is cleared by the device firmware after reading the OUT data from the endpoint.

For multi-bank endpoints, this bit may remain active even when cleared by the device firmware, this if an other packet has been received meanwhile.

Hardware assertion of this bit may generate an interrupt if enabled by the UDPHS\_EPTCTLx register RX\_BK\_RDY bit.

This bit is reset by UDPHS\_EPTRST register EPT\_x (reset endpoint) and by UDPHS\_EPTCTLDISx (disable endpoint).

- **KILL Bank:** (For IN endpoint)
  - the bank is really cleared or the bank is sent, BUSY\_BANK\_STA is decremented.
  - the bank is not cleared but sent on the IN transfer, TX\_COMPLT
  - the bank is not cleared because it was empty. The user should wait that this bit is cleared before trying to clear another packet.

**Note:** “Kill a packet” may be refused if at the same time, an IN token is coming and the current packet is sent on the UDPHS line. In this case, the TX\_COMPLT bit is set. Take notice however, that if at least two banks are ready to be sent, there is no problem to kill a packet even if an IN token is coming. In fact, in that case, the current bank is sent (IN transfer) and the last bank is killed.

- **TX\_COMPLT: Transmitted IN Data Complete**

This bit is set by hardware after an IN packet has been transmitted for isochronous endpoints and after it has been accepted (ACK’ed) by the host for Control, Bulk and Interrupt endpoints.

This bit is reset by UDPHS\_EPTRST register EPT\_x (reset endpoint), and by UDPHS\_EPTCTLDISx (disable endpoint).

- **TX\_PK\_RDY/ERR\_TRANS: TX Packet Ready/Transaction Error**

- **TX Packet Ready:**

This bit is cleared by hardware, as soon as the packet has been sent for isochronous endpoints, or after the host has acknowledged the packet for Control, Bulk and Interrupt endpoints.

For Multi-bank endpoints, this bit may remain clear even after software is set if another bank is available to transmit.

Hardware clear of this bit may generate an interrupt if enabled by the UDPHS\_EPTCTLx register TX\_PK\_RDY bit.

This bit is reset by UDPHS\_EPTRST register EPT\_x (reset endpoint), and by UDPHS\_EPTCTLDISx (disable endpoint).

- **Transaction Error:** (For high bandwidth isochronous OUT endpoints) (Read-Only)

This bit is set by hardware when a transaction error occurs inside one microframe.

If one toggle sequencing problem occurs among the n-transactions ( $n = 1, 2$  or  $3$ ) inside a microframe, then this bit is still set as long as the current bank contains one “bad” n-transaction. (see “[CURRENT\\_BANK/CONTROL\\_DIR: Current Bank/Control Direction](#)” on page 804) As soon as the current bank is relative to a new “good” n-transactions, then this bit is reset.

**Note1:** A transaction error occurs when the toggle sequencing does not respect the *Universal Serial Bus Specification, Rev 2.0* (5.9.2 High Bandwidth Isochronous endpoints) (Bad PID, missing data....)

**Note2:** When a transaction error occurs, the user may empty all the “bad” transactions by clearing the Received OUT Data flag (RX\_BK\_RDY).

If this bit is reset, then the user should consider that a new n-transaction is coming.

This bit is reset by UDPHS\_EPTRST register EPT\_x (reset endpoint), and by UDPHS\_EPTCTLDISx (disable endpoint).

- **RX\_SETUP/ERR\_FL\_ISO: Received SETUP/Error Flow**

- **Received SETUP:** (for Control endpoint only)

This bit is set by hardware when a valid SETUP packet has been received from the host.

It is cleared by the device firmware after reading the SETUP data from the endpoint FIFO.

This bit is reset by UDPHS\_EPTRST register EPT\_x (reset endpoint), and by UDPHS\_EPTCTLDISx (disable endpoint).

- **Error Flow:** (for isochronous endpoint only)

This bit is set by hardware when a transaction error occurs.

- Isochronous IN transaction is missed, the micro has no time to fill the endpoint (underflow).

- Isochronous OUT data is dropped because the bank is busy (overflow).

This bit is reset by UDPHS\_EPTRST register EPT\_x (reset endpoint) and by UDPHS\_EPTCTLDISx (disable endpoint).

- **STALL\_SNT/ERR\_CRISO/ERR\_NBTRA: Stall Sent/CRC ISO Error/Number of Transaction Error**

- **STALL\_SNT:** (for Control, Bulk and Interrupt endpoints)

This bit is set by hardware after a STALL handshake has been sent as requested by the UDPHS\_EPTSTAx register FRCESTALL bit.

This bit is reset by UDPHS\_EPTRST register EPT\_x (reset endpoint) and by UDPHS\_EPTCTLDISx (disable endpoint).

- **ERR\_CRISO:** (for Isochronous OUT endpoints) (Read-only)

This bit is set by hardware if the last received data is corrupted (CRC error on data).

This bit is updated by hardware when new data is received (Received OUT Data bit).

- **ERR\_NBTRA:** (for High Bandwidth Isochronous IN endpoints)

This bit is set at the end of a microframe in which at least one data bank has been transmitted, if less than the number of transactions per micro-frame banks (UDPHS\_EPTCFGx register NB\_TRANS) have been validated for transmission inside this microframe.

This bit is reset by UDPHS\_EPTRST register EPT\_x (reset endpoint) and by UDPHS\_EPTCTLDISx (disable endpoint).

- **NAK\_IN/ERR\_FLUSH: NAK IN/Bank Flush Error**

- **NAK\_IN:**

This bit is set by hardware when a NAK handshake has been sent in response to an IN request from the Host.

This bit is cleared by software.

- **ERR\_FLUSH:** (for High Bandwidth Isochronous IN endpoints)

This bit is set when flushing unsent banks at the end of a microframe.

This bit is reset by UDPHS\_EPTRST register EPT\_x (reset endpoint) and by EPT\_CTL\_DISx (disable endpoint).

• **NAK\_OUT: NAK OUT**

This bit is set by hardware when a NAK handshake has been sent in response to an OUT or PING request from the Host.

This bit is reset by UDPHS\_EPTRST register EPT\_x (reset endpoint) and by EPT\_CTL\_DISx (disable endpoint).

• **CURRENT\_BANK/CONTROL\_DIR: Current Bank/Control Direction**

– **Current Bank:** (all endpoints except Control endpoint)

These bits are set by hardware to indicate the number of the current bank.

00	Bank 0 (or single bank)
01	Bank 1
10	Bank 2
11	Invalid

**Note:** the current bank is updated each time the user:

- Sets the TX Packet Ready bit to prepare the next IN transfer and to switch to the next bank.
- Clears the received OUT data bit to access the next bank.

This bit is reset by UDPHS\_EPTRST register EPT\_x (reset endpoint) and by UDPHS\_EPTCTLDISx (disable endpoint).

– **Control Direction:** (for Control endpoint only)

0 = a Control Write is requested by the Host.

1 = a Control Read is requested by the Host.

**Note1:** This bit corresponds with the 7th bit of the bmRequestType (Byte 0 of the Setup Data).

**Note2:** This bit is updated after receiving new setup data.

• **BUSY\_BANK\_STA: Busy Bank Number**

These bits are set by hardware to indicate the number of busy banks.

**IN endpoint:** it indicates the number of busy banks filled by the user, ready for IN transfer.

**OUT endpoint:** it indicates the number of busy banks filled by OUT transaction from the Host.

00	All banks are free
01	1 busy bank
10	2 busy banks
11	3 busy banks

• **BYTE\_COUNT: UDPHS Byte Count**

Byte count of a received data packet.

This field is incremented after each write into the endpoint (to prepare an IN transfer).

This field is decremented after each reading into the endpoint (OUT transfer).

This field is also updated at RX\_BK\_RDY flag clear with the next bank.

This field is also updated at TX\_PK\_RDY flag set with the next bank.

This field is reset by EPT\_x of UDPHS\_EPTRST register.

- **SHRT\_PCKT: Short Packet**

An OUT Short Packet is detected when the receive byte count is less than the configured UDPHS\_EPTCFGx register EPT\_Size.

This bit is updated at the same time as the BYTE\_COUNT field.

This bit is reset by UDPHS\_EPTRST register EPT\_x (reset endpoint) and by UDPHS\_EPTCTLDISx (disable endpoint).

#### 41.5.15 UDPHS DMA Channel Transfer Descriptor

The DMA channel transfer descriptor is loaded from the memory.

Be careful with the alignment of this buffer.

The structure of the DMA channel transfer descriptor is defined by three parameters as described below:

Offset 0:

The address must be aligned: 0xXXXX0

Next Descriptor Address Register: UDPHS\_DMANXTDSCx

Offset 4:

The address must be aligned: 0xXXXX4

DMA Channelx Address Register: UDPHS\_DMAADDRESSx

Offset 8:

The address must be aligned: 0xXXXX8

DMA Channelx Control Register: UDPHS\_DMACONTROLx

To use the DMA channel transfer descriptor, fill the structures with the correct value (as described in the following pages).

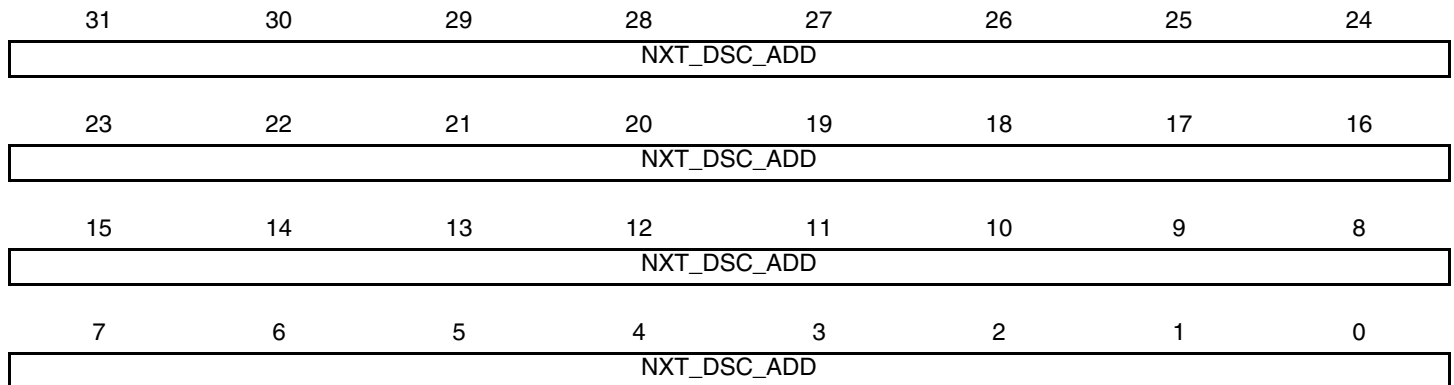
Then write directly in UDPHS\_DMANXTDSCx the address of the descriptor to be used first.

Then write 1 in the LDNXT\_DSC bit of UDPHS\_DMACONTROLx (load next channel transfer descriptor). The descriptor is automatically loaded upon Endpointx request for packet transfer.

## 41.5.16 UDPHS DMA Next Descriptor Address Register

**Name:** UDPHS\_DMANXTDSCx [x = 1..5]

**Access Type:** Read/Write



- **NXT\_DSC\_ADD**

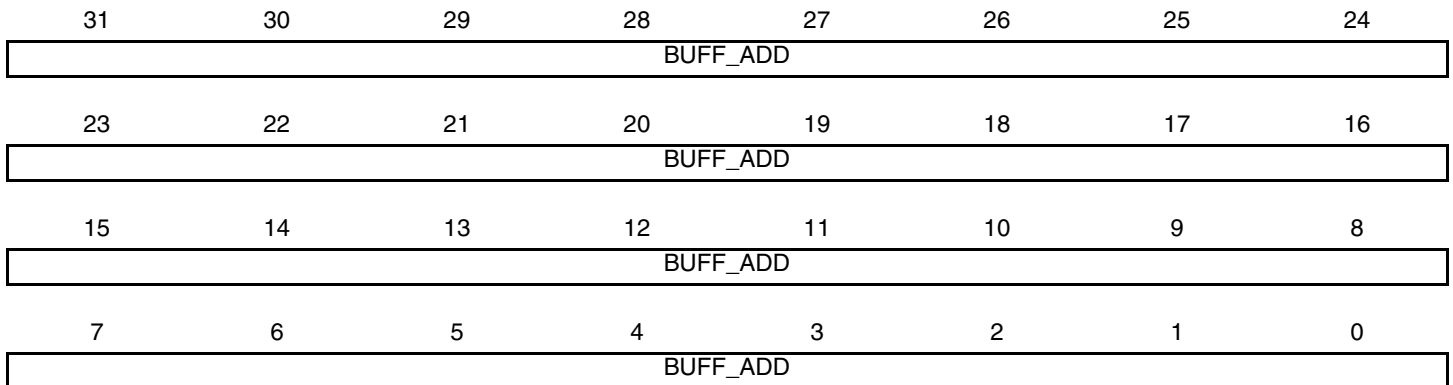
This field points to the next channel descriptor to be processed. This channel descriptor must be aligned, so bits 0 to 3 of the address must be equal to zero.



### 41.5.17 UDPHS DMA Channel Address Register

**Name:** UDPHS\_DMAADDRESSx [x = 1..5]

**Access Type:** Read/Write



• **BUFF\_ADD**

This field determines the AHB bus starting address of a DMA channel transfer.

Channel start and end addresses may be aligned on any byte boundary.

The firmware may write this field only when the UDPHS\_DMASTATUS register CHANN\_ENB bit is clear.

This field is updated at the end of the address phase of the current access to the AHB bus. It is incrementing of the access byte width. The access width is 4 bytes (or less) at packet start or end, if the start or end address is not aligned on a word boundary.

The packet start address is either the channel start address or the next channel address to be accessed in the channel buffer.

The packet end address is either the channel end address or the latest channel address accessed in the channel buffer.

The channel start address is written by software or loaded from the descriptor, whereas the channel end address is either determined by the end of buffer or the UDPHS device, USB end of transfer if the UDPHS\_DMACONTROLx register END\_TR\_EN bit is set.



## 41.5.18 UDPHS DMA Channel Control Register

**Name:** UDPHS\_DMACONTROLx [x = 1..5]

**Access Type:** Read/Write

31	30	29	28	27	26	25	24
BUFF_LENGTH							
23	22	21	20	19	18	17	16
BUFF_LENGTH							
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
BURST_LCK	DESC_LD_IT	END_BUFFIT	END_TR_IT	END_B_EN	END_TR_EN	LDNXT_DSC	CHANN_ENB

- **CHANN\_ENB (Channel Enable Command)**

0 = DMA channel is disabled at and no transfer will occur upon request. This bit is also cleared by hardware when the channel source bus is disabled at end of buffer.

If the UDPHS\_DMACONTROL register LDNXT\_DSC bit has been cleared by descriptor loading, the firmware will have to set the corresponding CHANN\_ENB bit to start the described transfer, if needed.

If the UDPHS\_DMACONTROL register LDNXT\_DSC bit is cleared, the channel is frozen and the channel registers may then be read and/or written reliably as soon as both UDPHS\_DMASTATUS register CHANN\_ENB and CHANN\_ACT flags read as 0.

If a channel request is currently serviced when this bit is cleared, the DMA FIFO buffer is drained until it is empty, then the UDPHS\_DMASTATUS register CHANN\_ENB bit is cleared.

If the LDNXT\_DSC bit is set at or after this bit clearing, then the currently loaded descriptor is skipped (no data transfer occurs) and the next descriptor is immediately loaded.

1 = UDPHS\_DMASTATUS register CHANN\_ENB bit will be set, thus enabling DMA channel data transfer. Then any pending request will start the transfer. This may be used to start or resume any requested transfer.

- **LDNXT\_DSC: Load Next Channel Transfer Descriptor Enable (Command)**

0 = no channel register is loaded after the end of the channel transfer.

1 = the channel controller loads the next descriptor after the end of the current transfer, i.e. when the UDPHS\_DMASTATUS/CHANN\_ENB bit is reset.

If the UDPHS\_DMA CONTROL/CHANN\_ENB bit is cleared, the next descriptor is immediately loaded upon transfer request.

LDNXT_DSC	CHANN_ENB	Description
0	0	Stop now
0	1	Run and stop at end of buffer
1	0	Load next descriptor now
1	1	Run and link at end of buffer

- **END\_TR\_EN: End of Transfer Enable (Control)**

Used for OUT transfers only.

0 = USB end of transfer is ignored.

1 = UDPHS device can put an end to the current buffer transfer.

When set, a BULK or INTERRUPT short packet or the last packet of an ISOCHRONOUS (micro) frame (DATAx) will close the current buffer and the UDPHS\_DMASTATUSx register END\_TR\_ST flag will be raised.

This is intended for UDPHS non-prenegotiated end of transfer (BULK or INTERRUPT) or ISOCHRONOUS microframe data buffer closure.

- **END\_B\_EN: End of Buffer Enable (Control)**

0 = DMA Buffer End has no impact on USB packet transfer.

1 = endpoint can validate the packet (according to the values programmed in the UDPHS\_EPTCTLx register AUTO\_VALID and SHRT\_PCKT fields) at DMA Buffer End, i.e. when the UDPHS\_DMASTATUS register BUFF\_COUNT reaches 0.

This is mainly for short packet IN validation initiated by the DMA reaching end of buffer, but could be used for OUT packet truncation (discarding of unwanted packet data) at the end of DMA buffer.

- **END\_TR\_IT: End of Transfer Interrupt Enable**

0 = UDPHS device initiated buffer transfer completion will not trigger any interrupt at UDPHS\_STATUSx/END\_TR\_ST rising.

1 = an interrupt is sent after the buffer transfer is complete, if the UDPHS device has ended the buffer transfer.

Use when the receive size is unknown.

- **END\_BUFFIT: End of Buffer Interrupt Enable**

0 = UDPHS\_DMA\_STATUSx/END\_BF\_ST rising will not trigger any interrupt.

1 = an interrupt is generated when the UDPHS\_DMASTATUSx register BUFF\_COUNT reaches zero.

- **DESC\_LD\_IT: Descriptor Loaded Interrupt Enable**

0 = UDPHS\_DMASTATUSx/DESC\_LDST rising will not trigger any interrupt.

1 = an interrupt is generated when a descriptor has been loaded from the bus.

- **BURST\_LCK: Burst Lock Enable**

0 = the DMA never locks bus access.

1 = USB packets AHB data bursts are locked for maximum optimization of the bus bandwidth usage and maximization of fly-by AHB burst duration.

- **BUFF\_LENGTH: Buffer Byte Length (Write-only)**

This field determines the number of bytes to be transferred until end of buffer. The maximum channel transfer size (64 KBytes) is reached when this field is 0 (default value). If the transfer size is unknown, this field should be set to 0, but the transfer end may occur earlier under UDPHS device control.

When this field is written, The UDPHS\_DMASTATUSx register BUFF\_COUNT field is updated with the write value.

Note: Bits [31:2] are only writable when issuing a channel Control Command other than “Stop Now”.

Note: For reliability it is highly recommended to wait for both UDPHS\_DMASTATUSx register CHAN\_ACT and CHAN\_ENB flags are at 0, thus ensuring the channel has been stopped before issuing a command other than “Stop Now”.

## 41.5.19 UDPHS DMA Channel Status Register

**Name:** UDPHS\_DMASTATUSx [x = 1..5]

**Access Type:** Read/Write

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

- **CHANN\_ENB: Channel Enable Status**

0 = if cleared, the DMA channel no longer transfers data, and may load the next descriptor if the UDPHS\_DMACONTROLx register LDNXT\_DSC bit is set.

When any transfer is ended either due to an elapsed byte count or a UDPHS device initiated transfer end, this bit is automatically reset.

1 = if set, the DMA channel is currently enabled and transfers data upon request.

This bit is normally set or cleared by writing into the UDPHS\_DMACONTROLx register CHANN\_ENB bit field either by software or descriptor loading.

If a channel request is currently serviced when the UDPHS\_DMACONTROLx register CHANN\_ENB bit is cleared, the DMA FIFO buffer is drained until it is empty, then this status bit is cleared.

- **CHANN\_ACT: Channel Active Status**

0 = the DMA channel is no longer trying to source the packet data.

When a packet transfer is ended this bit is automatically reset.

1 = the DMA channel is currently trying to source packet data, i.e. selected as the highest-priority requesting channel.

When a packet transfer cannot be completed due to an END\_BF\_ST, this flag stays set during the next channel descriptor load (if any) and potentially until UDPHS packet transfer completion, if allowed by the new descriptor.

- **END\_TR\_ST: End of Channel Transfer Status**

0 = cleared automatically when read by software.

1 = set by hardware when the last packet transfer is complete, if the UDPHS device has ended the transfer.

Valid until the CHANN\_ENB flag is cleared at the end of the next buffer transfer.

- **END\_BF\_ST: End of Channel Buffer Status**

0 = cleared automatically when read by software.

1 = set by hardware when the BUFF\_COUNT downcount reach zero.

Valid until the CHANN\_ENB flag is cleared at the end of the next buffer transfer.

- **DESC\_LDST: Descriptor Loaded Status**

0 = cleared automatically when read by software.

1 = set by hardware when a descriptor has been loaded from the system bus.

Valid until the CHANN\_ENB flag is cleared at the end of the next buffer transfer.

- **BUFF\_COUNT: Buffer Byte Count**

This field determines the current number of bytes still to be transferred for this buffer.

This field is decremented from the AHB source bus access byte width at the end of this bus address phase.

The access byte width is 4 by default, or less, at DMA start or end, if the start or end address is not aligned on a word boundary.

At the end of buffer, the DMA accesses the UDPHS device only for the number of bytes needed to complete it.

This field value is reliable (stable) only if the channel has been stopped or frozen (UDPHS\_EPTCTLx register NT\_DIS\_DMA bit is used to disable the channel request) and the channel is no longer active CHANN\_ACT flag is 0.

Note: For OUT endpoints, if the receive buffer byte length (BUFF\_LENGTH) has been defaulted to zero because the USB transfer length is unknown, the actual buffer byte length received will be 0x10000-BUFF\_COUNT.

## 42. Pulse Width Modulation (PWM) Controller

### 42.1 Description

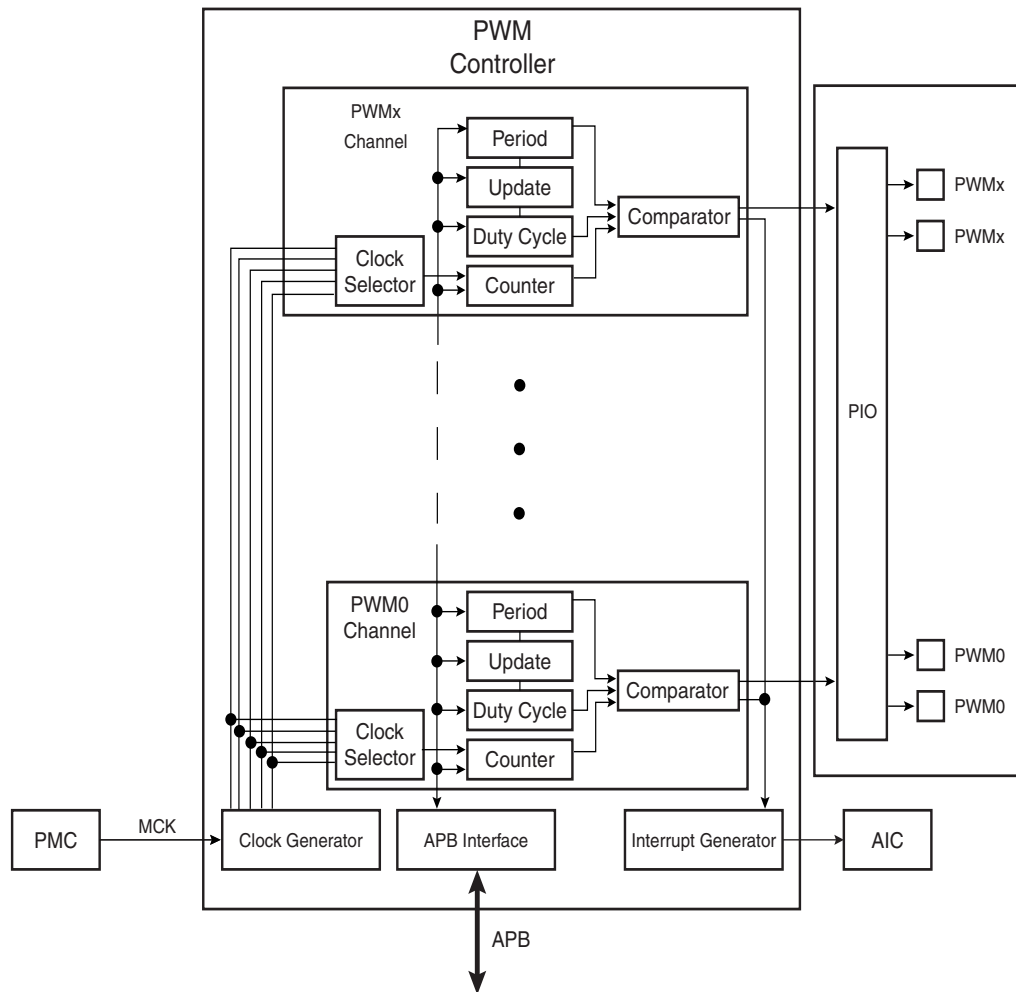
The PWM macrocell controls several channels independently. Each channel controls one square output waveform. Characteristics of the output waveform such as period, duty-cycle and polarity are configurable through the user interface. Each channel selects and uses one of the clocks provided by the clock generator. The clock generator provides several clocks resulting from the division of the PWM macrocell master clock.

All PWM macrocell accesses are made through APB mapped registers.

Channels can be synchronized, to generate non overlapped waveforms. All channels integrate a double buffering system in order to prevent an unexpected output waveform while modifying the period or the duty-cycle.

### 42.2 Block Diagram

Figure 42-1. Pulse Width Modulation Controller Block Diagram



## 42.3 I/O Lines Description

Each channel outputs one waveform on one external I/O line.

**Table 42-1.** I/O Line Description

Name	Description	Type
PWMx	PWM Waveform Output for Channel x	Output

## 42.4 Product Dependencies

### 42.4.1 I/O Lines

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

All of the PWM outputs may or may not be enabled. If an application requires only four channels, then only four PIO lines will be assigned to PWM outputs.

### 42.4.2 Power Management

The PWM is not continuously clocked. The programmer must first enable the PWM clock in the Power Management Controller (PMC) before using the PWM. However, if the application does not require PWM operations, the PWM clock can be stopped when not needed and be restarted later. In this case, the PWM will resume its operations where it left off.

Configuring the PWM does not require the PWM clock to be enabled.

### 42.4.3 Interrupt Sources

The PWM interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the PWM interrupt requires the AIC to be programmed first. Note that it is not recommended to use the PWM interrupt line in edge sensitive mode.

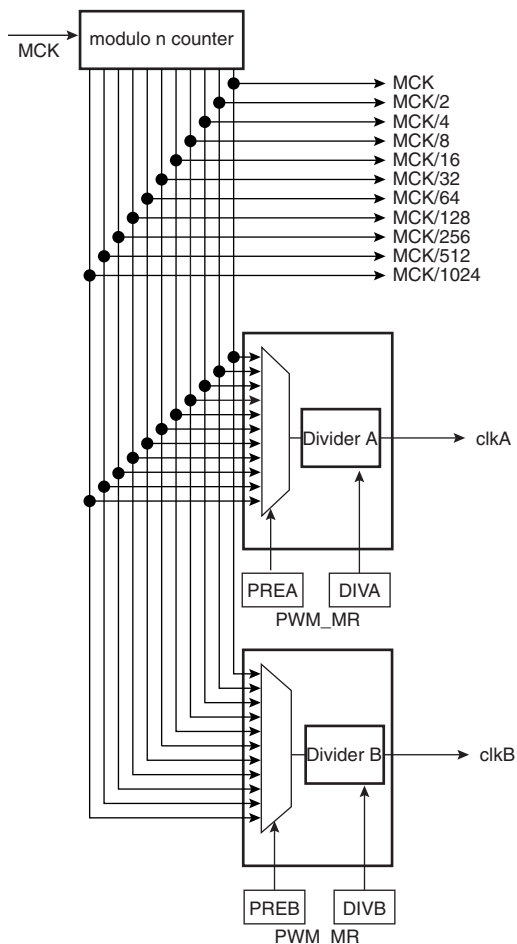
## 42.5 Functional Description

The PWM macrocell is primarily composed of a clock generator module and 4 channels.

- Clocked by the system clock, MCK, the clock generator module provides 13 clocks.
- Each channel can independently choose one of the clock generator outputs.
- Each channel generates an output waveform with attributes that can be defined independently for each channel through the user interface registers.

## 42.5.1 PWM Clock Generator

Figure 42-2. Functional View of the Clock Generator Block Diagram



**Caution:** Before using the PWM macrocell, the programmer must first enable the PWM clock in the Power Management Controller (PMC).

The PWM macrocell master clock, MCK, is divided in the clock generator module to provide different clocks available for all channels. Each channel can independently select one of the divided clocks.

The clock generator is divided in three blocks:

- a modulo n counter which provides 11 clocks:  $F_{MCK}$ ,  $F_{MCK}/2$ ,  $F_{MCK}/4$ ,  $F_{MCK}/8$ ,  $F_{MCK}/16$ ,  $F_{MCK}/32$ ,  $F_{MCK}/64$ ,  $F_{MCK}/128$ ,  $F_{MCK}/256$ ,  $F_{MCK}/512$ ,  $F_{MCK}/1024$
- two linear dividers (1, 1/2, 1/3, ... 1/255) that provide two separate clocks: clkA and clkB

Each linear divider can independently divide one of the clocks of the modulo n counter. The selection of the clock to be divided is made according to the PREA (PREB) field of the PWM Mode register (PWM\_MR). The resulting clock clkA (clkB) is the clock selected divided by DIVA (DIVB) field value in the PWM Mode register (PWM\_MR).

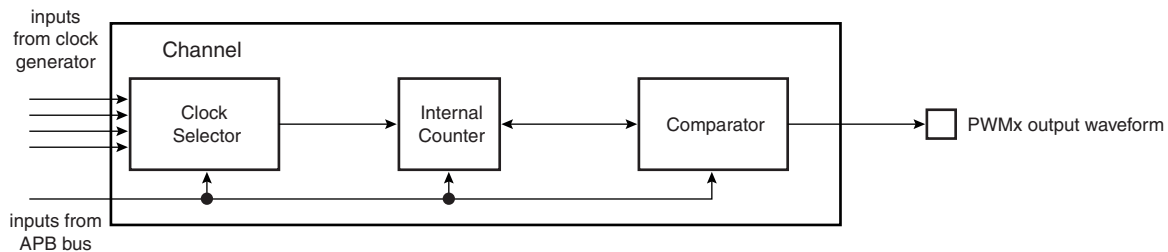
After a reset of the PWM controller, DIVA (DIVB) and PREA (PREB) in the PWM Mode register are set to 0. This implies that after reset clkA (clkB) are turned off.

At reset, all clocks provided by the modulo n counter are turned off except clock “clk”. This situation is also true when the PWM master clock is turned off through the Power Management Controller.

## 42.5.2 PWM Channel

### 42.5.2.1 Block Diagram

**Figure 42-3.** Functional View of the Channel Block Diagram



Each of the 4 channels is composed of three blocks:

- A clock selector which selects one of the clocks provided by the clock generator described in [Section 42.5.1 “PWM Clock Generator” on page 815](#).
- An internal counter clocked by the output of the clock selector. This internal counter is incremented or decremented according to the channel configuration and comparators events. The size of the internal counter is 16 bits.
- A comparator used to generate events according to the internal counter value. It also computes the PWMx output waveform according to the configuration.

### 42.5.2.2 Waveform Properties

The different properties of output waveforms are:

- the **internal clock selection**. The internal channel counter is clocked by one of the clocks provided by the clock generator described in the previous section. This channel parameter is defined in the CPRE field of the PWM\_CMRx register. This field is reset at 0.
- the **waveform period**. This channel parameter is defined in the CPRD field of the PWM\_CPRDx register.

- If the waveform is left aligned, then the output waveform period depends on the counter source clock and can be calculated:

By using the Master Clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024), the resulting period formula will be:

$$\frac{(X \times CPRD)}{MCK}$$

By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:



$$\frac{(CPRD \times DIVA)}{MCK} \text{ or } \frac{(CPRD \times DIVB)}{MCK}$$

If the waveform is center aligned then the output waveform period depends on the counter source clock and can be calculated:

By using the Master Clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(2 \times X \times CPRD)}{MCK}$$

By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(2 \times CPRD \times DIVA)}{MCK} \text{ or } \frac{(2 \times CPRD \times DIVB)}{MCK}$$

- the **waveform duty cycle**. This channel parameter is defined in the CDTY field of the PWM\_CDTYx register.

If the waveform is left aligned then:

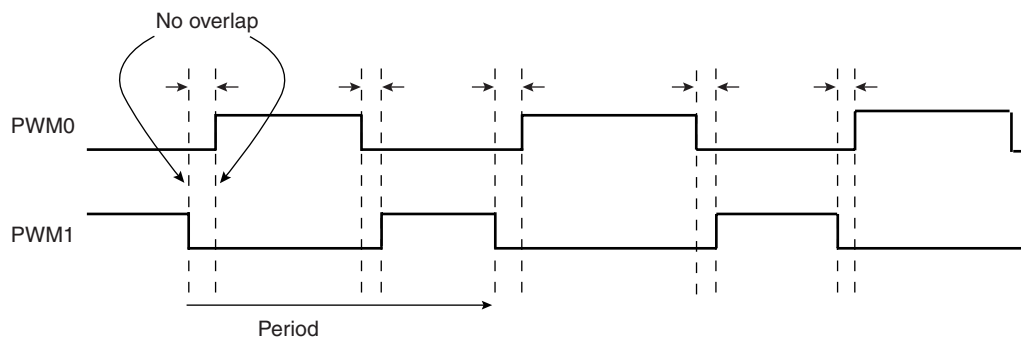
$$\text{duty cycle} = \frac{(\text{period} - 1 / \text{fchannel\_x\_clock} \times \text{CDTY})}{\text{period}}$$

If the waveform is center aligned, then:

$$\text{duty cycle} = \frac{((\text{period}/2) - 1 / \text{fchannel\_x\_clock} \times \text{CDTY})}{(\text{period}/2)}$$

- the **waveform polarity**. At the beginning of the period, the signal can be at high or low level. This property is defined in the CPOL field of the PWM\_CMRx register. By default the signal starts by a low level.
- the **waveform alignment**. The output waveform can be left or center aligned. Center aligned waveforms can be used to generate non overlapped waveforms. This property is defined in the CALG field of the PWM\_CMRx register. The default mode is left aligned.

**Figure 42-4.** Non Overlapped Center Aligned Waveforms



Note: 1. See [Figure 42-5 on page 819](#) for a detailed description of center aligned waveforms.

When center aligned, the internal channel counter increases up to CPRD and decreases down to 0. This ends the period.

When left aligned, the internal channel counter increases up to CPRD and is reset. This ends the period.



Thus, for the same CPRD value, the period for a center aligned channel is twice the period for a left aligned channel.

Waveforms are fixed at 0 when:

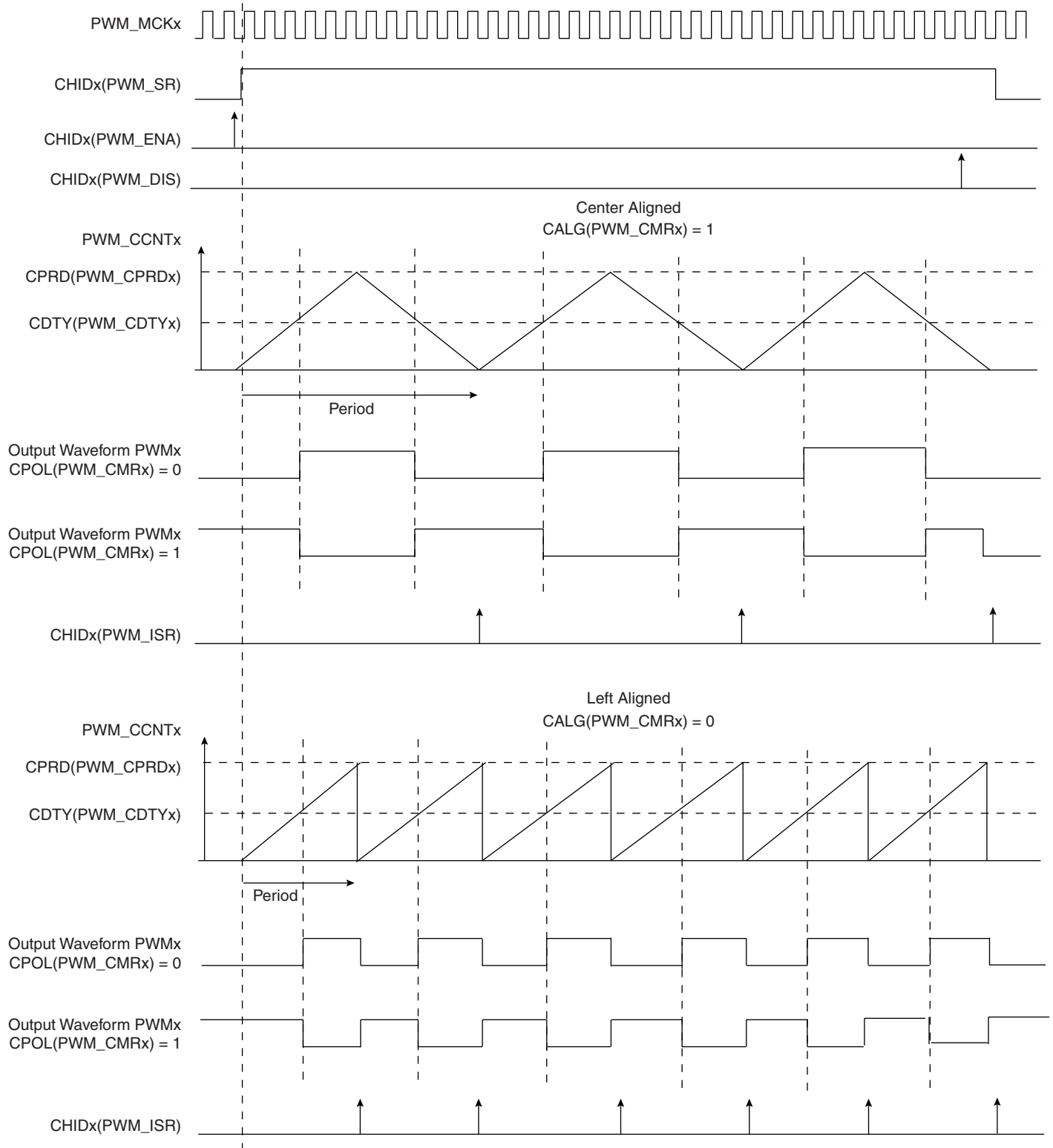
- CDTY = CPRD and CPOL = 0
- CDTY = 0 and CPOL = 1

Waveforms are fixed at 1 (once the channel is enabled) when:

- CDTY = 0 and CPOL = 0
- CDTY = CPRD and CPOL = 1

The waveform polarity must be set before enabling the channel. This immediately affects the channel output level. Changes on channel polarity are not taken into account while the channel is enabled.

**Figure 42-5. Waveform Properties**



## 42.5.3 PWM Controller Operations

### 42.5.3.1 Initialization

Before enabling the output channel, this channel must have been configured by the software application:

- Configuration of the clock generator if DIVA and DIVB are required
- Selection of the clock for each channel (CPRE field in the PWM\_CMRx register)
- Configuration of the waveform alignment for each channel (CALG field in the PWM\_CMRx register)
- Configuration of the period for each channel (CPRD in the PWM\_CPRDx register). Writing in PWM\_CPRDx Register is possible while the channel is disabled. After validation of the channel, the user must use PWM\_CUPDx Register to update PWM\_CPRDx as explained below.
- Configuration of the duty cycle for each channel (CDTY in the PWM\_CDTYx register). Writing in PWM\_CDTYx Register is possible while the channel is disabled. After validation of the channel, the user must use PWM\_CUPDx Register to update PWM\_CDTYx as explained below.
- Configuration of the output waveform polarity for each channel (CPOL in the PWM\_CMRx register)
- Enable Interrupts (Writing CHIDx in the PWM\_IER register)
- Enable the PWM channel (Writing CHIDx in the PWM\_ENA register)

It is possible to synchronize different channels by enabling them at the same time by means of writing simultaneously several CHIDx bits in the PWM\_ENA register.

- In such a situation, all channels may have the same clock selector configuration and the same period specified.

### 42.5.3.2 Source Clock Selection Criteria

The large number of source clocks can make selection difficult. The relationship between the value in the Period Register (PWM\_CPRDx) and the Duty Cycle Register (PWM\_CDTYx) can help the user in choosing. The event number written in the Period Register gives the PWM accuracy. The Duty Cycle quantum cannot be lower than  $1/PWM\_CPRDx$  value. The higher the value of PWM\_CPRDx, the greater the PWM accuracy.

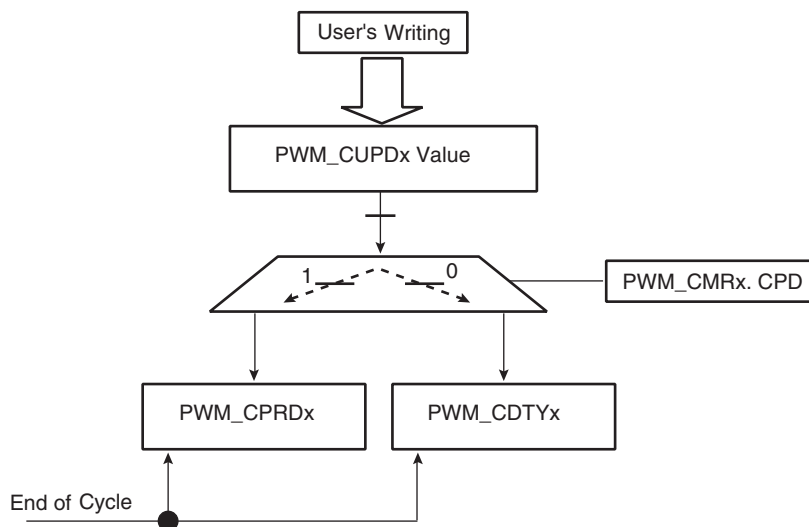
For example, if the user sets 15 (in decimal) in PWM\_CPRDx, the user is able to set a value between 1 up to 14 in PWM\_CDTYx Register. The resulting duty cycle quantum cannot be lower than  $1/15$  of the PWM period.

### 42.5.3.3 Changing the Duty Cycle or the Period

It is possible to modulate the output waveform duty cycle or period.

To prevent unexpected output waveform, the user must use the update register (PWM\_CUPDx) to change waveform parameters while the channel is still enabled. The user can write a new period value or duty cycle value in the update register (PWM\_CUPDx). This register holds the new value until the end of the current cycle and updates the value for the next cycle. Depending on the CPD field in the PWM\_CMRx register, PWM\_CUPDx either updates PWM\_CPRDx or PWM\_CDTYx. Note that even if the update register is used, the period must not be smaller than the duty cycle.

**Figure 42-6.** Synchronized Period or Duty Cycle Update



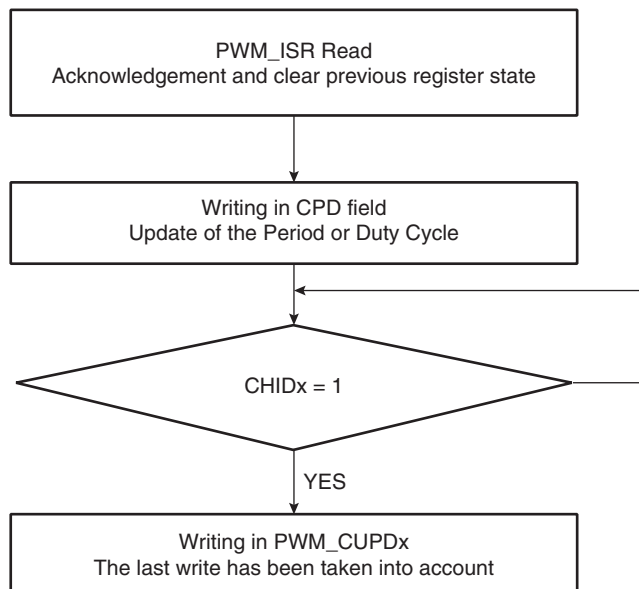
To prevent overwriting the PWM\_CUPDx by software, the user can use status events in order to synchronize his software. Two methods are possible. In both, the user must enable the dedicated interrupt in PWM\_IER at PWM Controller level.

The first method (polling method) consists of reading the relevant status bit in PWM\_ISR Register according to the enabled channel(s). See [Figure 42-7](#).

The second method uses an Interrupt Service Routine associated with the PWM channel.

Note: Reading the PWM\_ISR register automatically clears CHIDx flags.

**Figure 42-7.** Polling Method



Note: Polarity and alignment can be modified only when the channel is disabled.

#### 42.5.3.4 Interrupts

Depending on the interrupt mask in the PWM\_IMR register, an interrupt is generated at the end of the corresponding channel period. The interrupt remains active until a read operation in the PWM\_ISR register occurs.

A channel interrupt is enabled by setting the corresponding bit in the PWM\_IER register. A channel interrupt is disabled by setting the corresponding bit in the PWM\_IDR register.

## 42.6 Pulse Width Modulation (PWM) Controller User Interface

**Table 42-2.** PWM Controller Registers

Offset	Register	Name	Access	Peripheral Reset Value
0x00	PWM Mode Register	PWM_MR	Read/Write	0
0x04	PWM Enable Register	PWM_ENA	Write-only	-
0x08	PWM Disable Register	PWM_DIS	Write-only	-
0x0C	PWM Status Register	PWM_SR	Read-only	0
0x10	PWM Interrupt Enable Register	PWM_IER	Write-only	-
0x14	PWM Interrupt Disable Register	PWM_IDR	Write-only	-
0x18	PWM Interrupt Mask Register	PWM_IMR	Read-only	0
0x1C	PWM Interrupt Status Register	PWM_ISR	Read-only	0
0x4C - 0xFC	Reserved	—	—	—
0x100 - 0x1FC	Reserved			
0x200	Channel 0 Mode Register	PWM_CMR0	Read/Write	0x0
0x204	Channel 0 Duty Cycle Register	PWM_CDTY0	Read/Write	0x0
0x208	Channel 0 Period Register	PWM_CPRD0	Read/Write	0x0
0x20C	Channel 0 Counter Register	PWM_CCNT0	Read-only	0x0
0x210	Channel 0 Update Register	PWM_CUPD0	Write-only	-
...	Reserved			
0x220	Channel 1 Mode Register	PWM_CMR1	Read/Write	0x0
0x224	Channel 1 Duty Cycle Register	PWM_CDTY1	Read/Write	0x0
0x228	Channel 1 Period Register	PWM_CPRD1	Read/Write	0x0
0x22C	Channel 1 Counter Register	PWM_CCNT1	Read-only	0x0
0x230	Channel 1 Update Register	PWM_CUPD1	Write-only	-
...	...	...	...	...

## 42.6.1 PWM Mode Register

Register Name: PWM\_MR

Access Type: Read/Write

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

- **DIVA, DIVB: CLKA, CLKB Divide Factor**

DIVA, DIVB	CLKA, CLKB
0	CLKA, CLKB clock is turned off
1	CLKA, CLKB clock is clock selected by PREA, PREB
2-255	CLKA, CLKB clock is clock selected by PREA, PREB divided by DIVA, DIVB factor.

- **PREA, PREB**

PREA, PREB				Divider Input Clock
0	0	0	0	MCK.
0	0	0	1	MCK/2
0	0	1	0	MCK/4
0	0	1	1	MCK/8
0	1	0	0	MCK/16
0	1	0	1	MCK/32
0	1	1	0	MCK/64
0	1	1	1	MCK/128
1	0	0	0	MCK/256
1	0	0	1	MCK/512
1	0	1	0	MCK/1024
Other				Reserved

### 42.6.2 PWM Enable Register

Register Name: PWM\_ENA

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
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = No effect.

1 = Enable PWM output for channel x.

### 42.6.3 PWM Disable Register

Register Name: PWM\_DIS

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
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = No effect.

1 = Disable PWM output for channel x.



## 42.6.4 PWM Status Register

**Register Name:** PWM\_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
-	-	-	-	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = PWM output for channel x is disabled.

1 = PWM output for channel x is enabled.

#### 42.6.5 PWM Interrupt Enable Register

Register Name: PWM\_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
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID.**

0 = No effect.

1 = Enable interrupt for PWM channel x.

#### 42.6.6 PWM Interrupt Disable Register

Register Name: PWM\_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
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID.**

0 = No effect.

1 = Disable interrupt for PWM channel x.

## 42.6.7 PWM Interrupt Mask Register

**Register Name:** PWM\_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
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID.**

0 = Interrupt for PWM channel x is disabled.

1 = Interrupt for PWM channel x is enabled.

## 42.6.8 PWM Interrupt Status Register

**Register Name:** PWM\_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
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = No new channel period has been achieved since the last read of the PWM\_ISR register.

1 = At least one new channel period has been achieved since the last read of the PWM\_ISR register.

Note: Reading PWM\_ISR automatically clears CHIDx flags.



### 42.6.9 PWM Channel Mode Register

Register Name: PWM\_CMRx

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
–	–	–	–	–	CPD	CPOL	CALG
7	6	5	4	3	2	1	0
–	–	–	–	CPRE			

• **CPRE: Channel Pre-scaler**

CPRE				Channel Pre-scaler
0	0	0	0	MCK
0	0	0	1	MCK/2
0	0	1	0	MCK/4
0	0	1	1	MCK/8
0	1	0	0	MCK/16
0	1	0	1	MCK/32
0	1	1	0	MCK/64
0	1	1	1	MCK/128
1	0	0	0	MCK/256
1	0	0	1	MCK/512
1	0	1	0	MCK/1024
1	0	1	1	CLKA
1	1	0	0	CLKB
Other				Reserved

• **CALG: Channel Alignment**

- 0 = The period is left aligned.
- 1 = The period is center aligned.

• **CPOL: Channel Polarity**

- 0 = The output waveform starts at a low level.
- 1 = The output waveform starts at a high level.

• **CPD: Channel Update Period**

- 0 = Writing to the PWM\_CUPDx will modify the duty cycle at the next period start event.
- 1 = Writing to the PWM\_CUPDx will modify the period at the next period start event.

## 42.6.10 PWM Channel Duty Cycle Register

**Register Name:** PWM\_CDTYx

**Access Type:** Read/Write

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

Only the first 16 bits (internal channel counter size) are significant.

- **CDTY: Channel Duty Cycle**

Defines the waveform duty cycle. This value must be defined between 0 and CPRD (PWM\_CPRx).

### 42.6.11 PWM Channel Period Register

**Register Name:** PWM\_CPRDx

**Access Type:** Read/Write

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

Only the first 16 bits (internal channel counter size) are significant.

- **CPRD: Channel Period**

If the waveform is left-aligned, then the output waveform period depends on the counter source clock and can be calculated:

- By using the Master Clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(X \times CPRD)}{MCK}$$

- By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(CPRD \times DIVA)}{MCK} \text{ or } \frac{(CPRD \times DIVB)}{MCK}$$

If the waveform is center-aligned, then the output waveform period depends on the counter source clock and can be calculated:

- By using the Master Clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(2 \times X \times CPRD)}{MCK}$$

- By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(2 \times CPRD \times DIVA)}{MCK} \text{ or } \frac{(2 \times CPRD \times DIVB)}{MCK}$$

## 42.6.12 PWM Channel Counter Register

**Register Name:** PWM\_CCNTx

**Access Type:** Read-only

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

- **CNT: Channel Counter Register**

Internal counter value. This register is reset when:

- the channel is enabled (writing CHIDx in the PWM\_ENA register).
- the counter reaches CPRD value defined in the PWM\_CPRDx register if the waveform is left aligned.

## 42.6.13 PWM Channel Update Register

**Register Name:** PWM\_CUPDx

**Access Type:** Write-only

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

This register acts as a double buffer for the period or the duty cycle. This prevents an unexpected waveform when modifying the waveform period or duty-cycle.

Only the first 16 bits (internal channel counter size) are significant.

CPD (PWM_CMRx Register)	
0	The duty-cycle (CDTC in the PWM_CDRx register) is updated with the CUPD value at the beginning of the next period.
1	The period (CPRD in the PWM_CPRx register) is updated with the CUPD value at the beginning of the next period.





## 43. Touch Screen ADC Controller

### 43.1 Description

The Touch Screen ADC Controller is based on a Successive Approximation Register (SAR) 10-bit Analog-to-Digital Converter (ADC). It also integrates:

- a 6-to-1 analog multiplexer for analog-to-digital conversions of up to 6 analog lines
- 4 power switches that measure both axis positions on the resistive touch screen panel
- 1 additional power switch and an embedded resistor that detects pen-interrupt and pen loss

The conversions extend from 0V to TSADVREF.

The TSADCC 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.

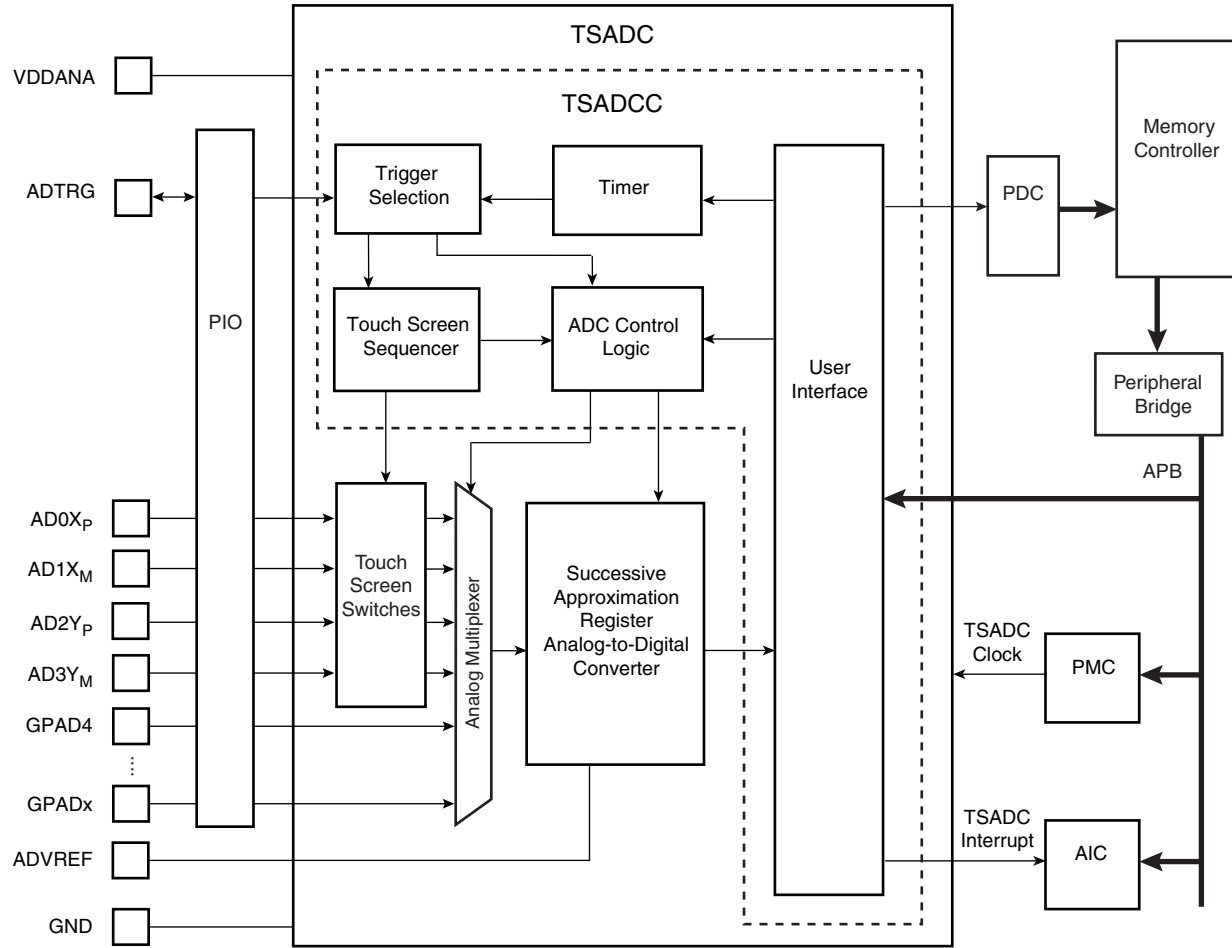
Conversions can be started for all enabled channels, either by a software trigger, by detection of a rising edge on the external trigger pin TSADTRG or by an integrated programmable timer. When the Touch Screen is enabled, a timer-triggered sequencer automatically configures the power switches, performs the conversions and stores the results in dedicated registers.

The TSADCC also integrates a Sleep Mode and a Pen-Detect Mode and connects with one PDC channel. These features reduce both power consumption and processor intervention.

The TSADCC timings, like the Startup Time and Sample and Hold Time, are fully configurable.

## 43.2 Block Diagram

Figure 43-1. TSADCC Block Diagram



GPADx: last general-purpose ADC channel defined by the number of channels

## 43.3 Signal Description

**Table 43-1.** TSADCC Pin Description

Pin Name	Description
VDDANA	Analog power supply
TSADVREF	Reference voltage
AD0X <sub>P</sub>	Analog input channel 0 or Touch Screen Top channel
AD1X <sub>M</sub>	Analog input channel 1 or Touch Screen Bottom channel
AD2Y <sub>P</sub>	Analog input channel 2 or Touch Screen Right channel
AD3Y <sub>M</sub>	Analog input channel 3 or Touch Screen Left channel
GPAD4 - GPAD5	General-purpose analog input channels 4 to 5
TSADTRG	External trigger

## 43.4 Product Dependencies

### 43.4.1 Power Management

The Analog-to-Digital Converter (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 TSADCC behavior.

### 43.4.2 Interrupt Sources

The TSADCC interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the TSADCC interrupt requires the AIC to be programmed first.

### 43.4.3 Analog Inputs

The analog input pins can be multiplexed with PIO lines. In this case, the assignment of the TSADCC input is automatically done as soon as the corresponding channel is enabled by writing the register TSADCC\_CHER. By default, after reset, the PIO lines are configured as input with its pull-up enabled and the TSADCC inputs are connected to the GND.

### 43.4.4 I/O Lines

The pin TSADTRG 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 TSADTRG to the TSADCC function.

### 43.4.5 Conversion Performances

For performance and electrical characteristics of the TSADCC, see the section “Electrical Characteristics” of the full datasheet.

## 43.5 Analog-to-digital Converter Functional Description

The TSADCC embeds a Successive Approximation Register (SAR) Analog-to-Digital Converter (ADC). The ADC supports 8-bit or 10-bit resolutions.

The conversion is performed on a full range between 0V and the reference voltage pin TSAD-VREF. Analog inputs between these voltages convert to values based on a linear conversion.

### 43.5.1 ADC Resolution

The ADC supports 8-bit or 10-bit resolutions. The 8-bit selection is performed by setting the bit LOWRES in the TSADCC Mode Register. See [Section 43.10.2 “TSADCC Mode Register” on page 848](#).

By default, after a reset, the resolution is the highest and the DATA field in the [“TSADCC Channel Data Register x \(x = 0..5\)”](#) are 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 TSADCC\_CDR register and of the LDATA field in the TSADCC\_LCDR register read 0.

Moreover, when a PDC channel is connected to the TSADCC, 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.

All the conversions for the Touch Screen forces the ADC in 10-bit resolution, regardless of the LOWRES setting. Further details are given in the section [“Operating Modes” on page 843](#).

### 43.5.2 ADC Clock

The TSADCC 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 [“TSADCC Mode Register”](#) and 10 ADC Clock cycles. The ADC Clock frequency is selected in the PRESCAL field of the [“TSADCC Mode Register”](#).

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 maximum sampling rate parameter given in the Electrical Characteristics section.

### 43.5.3 Sleep Mode

The TSADCC Sleep Mode maximizes power saving by automatically deactivating the Analog-to-Digital Converter cell when it is not being used for conversions. Sleep Mode is enabled by setting the bit SLEEP in [“TSADCC Mode Register”](#).

The SLEEP of the ADC is automatically managed by the conversion sequencer, which can automatically process the conversions of all channels at lowest power consumption.

When a trigger occurs, the Analog-to-Digital Converter cell is automatically activated. As the analog cell requires a start-up time, the logic waits during this time and then starts the conversion on the enabled channels. When all conversions are complete, the ADC is deactivated until the next trigger.

### 43.5.4 Startup Time

The Touch Screen ADC has a minimal Startup Time when it exits the Sleep Mode. As the ADC Clock depends on the application, the user has to program the field STARTUP in the [“TSADCC](#)

[Mode Register](#)", which defines how many ADC Clock cycles to wait before performing the first conversion of the sequence.

The field STARTUP can define a Startup Time between 8 and 1024 ADC Clock cycles by steps of 8.

The user must assure that ADC Startup Time given in the section "Electrical Characteristics" is covered by this wait time.

## 43.5.5 Sample and Hold Time

In the same way, a minimal Sample and Hold Time is necessary for the TSADCC to guarantee the best converted final value between selection of two channels. This time depends on the input impedance of the analog input, but also on the output impedance of the driver providing the signal to the analog input, as there is no input buffer amplifier.

The Sample and Hold time has to be programmed through the bitfields SHTIM in the ["TSADCC Mode Register"](#) and TSSHTIM in the ["TSADCC Touch Screen Register"](#).

The field SHTIM defines the number of ADC Clock cycles for an analog input, while the field TSSHTIM defines the number of ADC Clock cycles for a Touch Screen input.

These both fields can define a Sample and Hold time between 1 and 16 ADC Clock cycles.

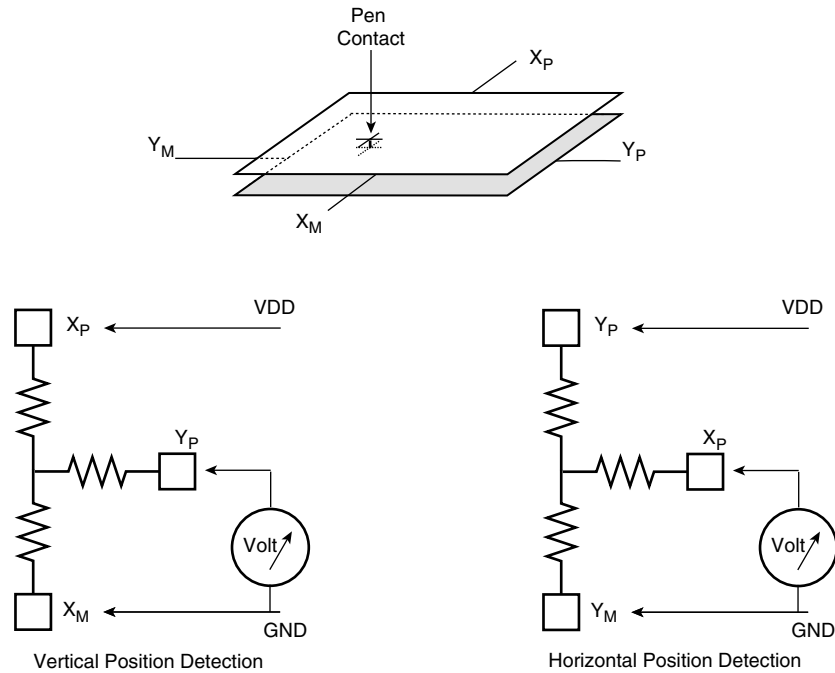
The field TSSHTIM defines also the time the power switches of the Touch Screen are closed when the TSADCC performs a conversion for the Touch Screen.

## 43.6 Touch Screen

### 43.6.1 Resistive Touch Screen Principles

A resistive touch screen is based on two resistive films, each one being fitted with a pair of electrodes, placed at the top and bottom on one film, and on the right and left on the other. Between the two, there is a layer that acts as an insulator, but also enables contact when you press the screen. This is illustrated in [Figure 43-2](#).

**Figure 43-2.** Touch Screen Position Measurement



**43.6.2 Position Measurement Method**

As shown in [Figure 43-2](#), to detect the position of a contact, a supply is first applied from top to bottom. Due to the linear resistance of the film, there is a voltage gradient from top to bottom. When a contact is performed on the screen, the voltage propagates at the point the two surfaces come into contact with the second film. If the input impedance on the right and left electrodes sense is high enough, the film does not affect this voltage, despite its resistive nature.

For the horizontal direction, the same method is used, but by applying supply from left to right. The range depends on the supply voltage and on the loss in the switches that connect to the top and bottom electrodes.

In an ideal world (linear, with no loss through switches), the horizontal position is equal to:

$$V_{Y_M} / V_{DD} \text{ or } V_{Y_P} / V_{DD}.$$

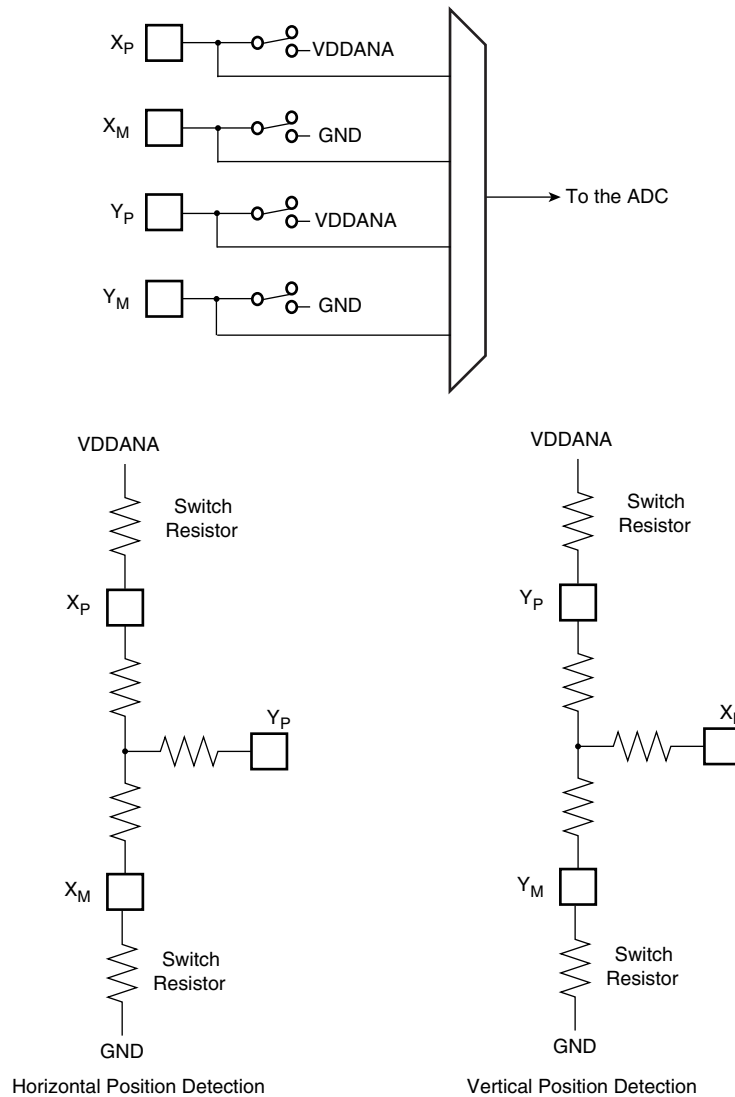
The proposed implementation with on-chip power switches is shown in [Figure 43-3](#). The voltage measurement at the output of the switch compensates for the switches loss.

It is possible to correct for the switch loss by performing the operation:

$$[V_{Y_P} - V_{X_M}] / [V_{X_P} - V_{X_M}].$$

This requires additional measurements, as shown in [Figure 43-3](#).

**Figure 43-3.** Touch Screen Switches Implementation



### 43.6.3 Pen Detect Method

When there is no contact, it is not necessary to perform conversion. However, it is important to detect a contact by keeping the power consumption as low as possible.

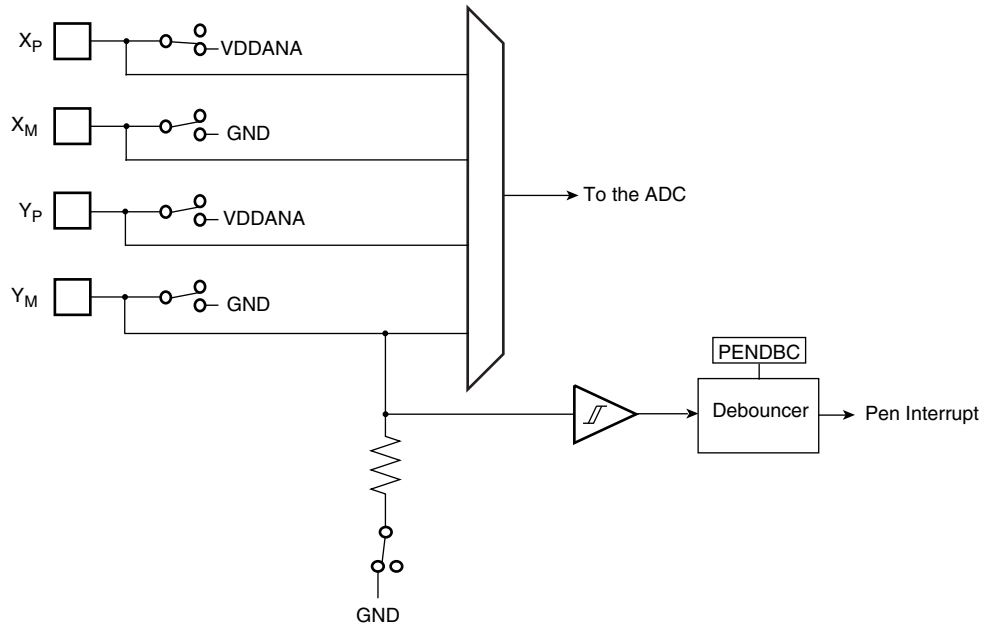
The proposed implementation polarizes the vertical panel by closing the switch on  $X_P$  and ties the horizontal panel by an embedded resistor connected to  $Y_M$ . This resistor is enabled by a fifth switch. Since there is no contact, no current is flowing and there is no related power consumption. As soon as a contact occurs, a current is flowing in the touch screen and a schmitt trigger detects the voltage in the resistor.

The Touch Screen Interrupt configuration is entered by programming the bit `PENDET` in the [“TSADCC Mode Register”](#). If this bit is written at 1, the switch on  $X_P$  and the switch on the resistor are both closed, except when a touch screen conversion is in progress.

To complete the circuit, a programmable debouncer is placed at the output of the schmitt trigger. This debouncer is programmable at 1 ADC Clock period, useful when the system is running at Slow Clock, or at up to  $2^{15}$  ADC Clock periods, but better used to filter noise on the Touch

Screen panel when the system is running at high speed. The debouncer length can be selected by programming the field PENDBC in “[TSADCC Mode Register](#)”.

**Figure 43-4.** Touch Screen Pen Detect



The Touch Screen Pen Detect can be used to generate a TSADCC interrupt to wake up the system or it can be programmed to trig a conversion, so that a position can be measured as soon as a contact is detected if the TSADCC is programmed for an operating mode involving the Touch Screen.

The Pen Detect generates two types of status, reported in the “[TSADCC Status Register](#)”:

- the bit PENCNT is set as soon as a current flows for a time over the debouncing time as defined by PENDBC and remains set until TSADCC\_SR is read.
- the bit NOCNT is set as soon as no current flows for a time over the debouncing time as defined by PENDBC and remains set until TSADCC\_SR is read.

Both bits are automatically cleared as soon as the Status Register TSADCC\_SR is read, and can generate an interrupt by writing accordingly the “[TSADCC Interrupt Enable Register](#)”.

### 43.7 Conversion Results

When a conversion is completed, the resulting 8-bit or 10-bit digital value is right-aligned and stored in the “[TSADCC Channel Data Register x \(x = 0..5\)](#)” of the current channel and in the “[TSADCC Last Converted Data Register](#)”.

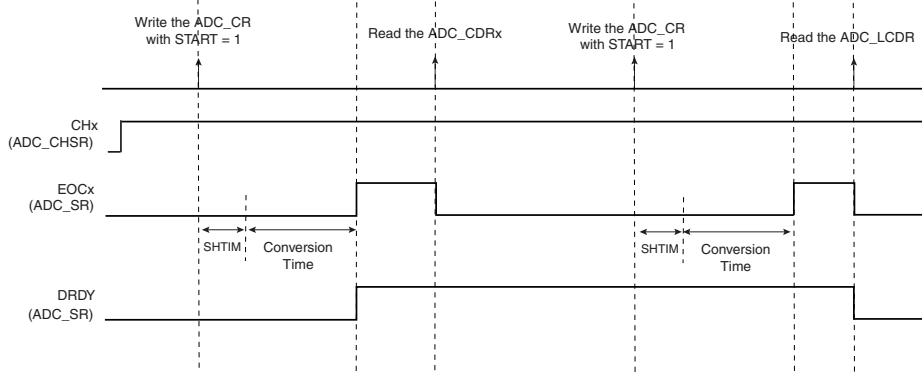
The channel EOC bit and the bit DRDY in the “[TSADCC Status Register](#)” are both set. If the PDC channel is enabled, DRDY rising triggers a data transfer. In any case, either EOC and DRDY can trigger an interrupt.

Reading one of the “[TSADCC Channel Data Register x \(x = 0..5\)](#)” registers clears the corresponding EOC bit.

Reading “[TSADCC Last Converted Data Register](#)” clears the DRDY bit and the EOC bit corresponding to the last converted channel.



**Figure 43-5. EOCx and DRDY Flag Behavior**

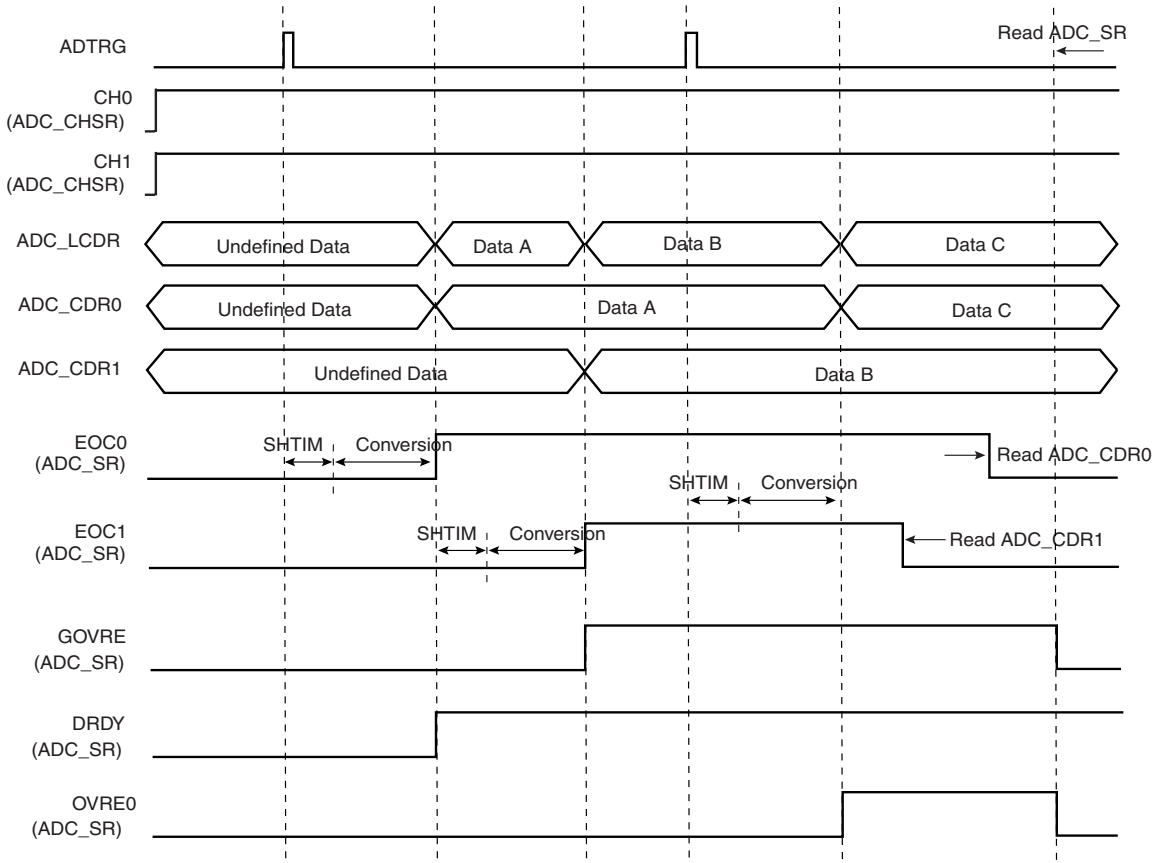


If the “[TSADCC Channel Data Register x \(x = 0..5\)](#)” is not read before further incoming data is converted, the corresponding Overrun Error (OVRE) flag is set in the “[TSADCC Status Register](#)”.

In the same way, new data converted when DRDY is high sets the bit GOVRE (General Overrun Error) in the “[TSADCC Status Register](#)”.

The OVRE and GOVRE flags are automatically cleared when the “[TSADCC Status Register](#)” is read.

**Figure 43-6. GOVRE and OVREx Flag Behavior**





**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 TSADCC\_SR are unpredictable.

## 43.8 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 “[TSADCC Control Register](#)” with the bit START at 1.

The hardware trigger can be selected by the filed TRGMOD in the TSADCC Trigger Register (TSADCC\_TRGR) between:

- an edge, either rising or falling or any, detected on the external trigger pin TSADTRG
- the Pen Detect, depending on how the PENDET bit is set in the “[TSADCC Mode Register](#)”
- a continuous trigger, meaning the TSADCC restarts the next sequence as soon as it finishes the current one, in this case, only one software trigger is required at the beginning
- a periodic trigger, which is defined by programming the field TRGPER in the “[TSADCC Trigger Register](#)”

Enabling hardware triggers does not disable the software trigger functionality. Thus, if a hardware trigger is selected, the start of a conversion can still be initiated by the software trigger.

## 43.9 Operating Modes

The Touch Screen ADC Controller features several operating modes, each defining a conversion sequence:

- The ADC Mode: at each trigger, all the enabled channels are converted
- The Touch Screen Mode: at each trigger, the touch screen inputs are converted with the switches accordingly set and the results are processed and stored in the corresponding data registers

The Operating Mode of the TSADCC is programmed in the field TSAMOD in the “[TSADCC Mode Register](#)”.

The conversion sequences for each Operating Mode are described in the following paragraphs.

The conversion sequencer, combined with the Sleep Modes, allows automatic processing with minimum processor intervention and optimized power consumption. In any case, the sequence starts with a trigger event.

Note: The reference voltage pins always remain connected in normal mode as in sleep mode.

### 43.9.1 ADC Mode

In the ADC Mode, the active channels are defined by the “[TSADCC Channel Status Register](#)”, which is defined by writing the “[TSADCC Channel Enable Register](#)” and “[TSADCC Channel Disable Register](#)”. The results are stored in the “[TSADCC Channel Data Register x \(x = 0..5\)](#)” and in the “[TSADCC Last Converted Data Register](#)”, so that data transfers by using the PDC are possible.

At each trigger, the following sequence is performed:

4. If SLEEP is set, wake up the ADC cell and wait for the Startup Time.
5. If Channel 0 is enabled, convert Channel 0 and store result in both TSADCC\_CDR0 and TSADCC\_LCDR.
6. If Channel 1 is enabled, convert Channel 1 and store result in both TSADCC\_CDR1 and TSADCC\_LCDR.

7. If Channel 2 is enabled, convert Channel 2 and store result in both TSADCC\_CDR2 and TSADCC\_LCDR.
8. If Channel 3 is enabled, convert Channel 3 and store result in both TSADCC\_CDR3 and TSADCC\_LCDR.
9. If Channel 4 to Channel 5 are enabled, convert the Channels and store result in the corresponding TSADCC\_CDRx and TSADCC\_LCDR.
10. If SLEEP is set, sleep down the ADC cell.

If the PDC is enabled, all the converted data are transferred contiguously in the memory buffer. The bit LOWRES defines which resolution is used, either 8-bit or 10-bit, and thus the width of the PDC memory buffer.

## 43.9.2 Touch Screen Mode

Writing TSAMOD to “Touch Screen Only Mode” automatically enables the touch screen pins as analog inputs, and thus disables the digital function of the corresponding pins.

In Touch Screen Mode, the channels 0 to 3 corresponding to the Touch Screen inputs are automatically activated and the bits CH0 to CH3 are automatically set in the “[TSADCC Channel Status Register](#)”.

The remaining channels can be either enabled or disabled by the user and their conversions are performed at the end of each touch screen sequence.

The resolution is forced to 10 bits, regardless of the LOWRES bit setting.

At each trigger, the following sequence is performed:

1. If SLEEP is set, wake up the ADC cell and wait for the Startup Time.
2. Close the switches on the inputs  $X_P$  and  $X_M$  during the Sample and Hold Time.
3. Convert Channel  $X_M$  and store the result in TSADCC\_CDR1.
4. Close the switches on the inputs  $X_P$  and  $X_M$  during the Sample and Hold Time.
5. Convert Channel  $X_P$  subtract TSADCC\_CDR1 from the result and store the subtraction result in both TSADCC\_CDR0 and TSADCC\_LCDR.
6. Close the switches on the inputs  $X_P$  and  $X_M$  during the Sample and Hold Time.
7. Convert Channel  $Y_P$  subtract TSADCC\_CDR1 from the result and store the subtraction result in both TSADCC\_CDR1 and TSADCC\_LCDR.
8. Close the switches on the inputs  $Y_P$  and  $Y_M$  during the Sample and Hold Time.
9. Convert Channel  $Y_M$  and store the result in TSADCC\_CDR3.
10. Close the switches on the inputs  $Y_P$  and  $Y_M$  during the Sample and Hold Time.
11. Convert Channel  $Y_P$  subtract TSADCC\_CDR3 from the result and store the subtraction result in both TSADCC\_CDR2 and TSADCC\_LCDR.
12. Close the switches on the inputs  $Y_P$  and  $Y_M$  during the Sample and Hold Time.
13. Convert Channel  $X_P$  subtract TSADCC\_CDR3 from the result and store the subtraction result in both TSADCC\_CDR3 and TSADCC\_LCDR.
14. If Channel 4 to Channel 5 are enabled, convert the Channels and store result in the corresponding TSADCC\_CDRx and TSADCC\_LCDR.
15. If SLEEP is set, sleep down the ADC cell.

The resulting buffer is 16 bits wide and its structure stored in memory is:

1.  $X_P - X_M$
2.  $Y_P - X_M$

3.  $Y_P - Y_M$
4.  $X_P - Y_M$
5. AD4 to AD5 if enabled.

The vertical position can be easily calculated by dividing the data at offset 0 ( $X_P - X_M$ ) by the data at offset 1 ( $Y_P - X_M$ ).

The horizontal position can be easily calculated by dividing the data at offset 2 ( $Y_P - Y_M$ ) by the data at offset 3 ( $X_P - Y_M$ ).

## 43.10 Touch Screen ADC Controller (TSADCC) User Interface

**Table 43-2.** Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	TSADCC_CR	Write-only	–
0x04	Mode Register	TSADCC_MR	Read-write	0x0000_0000
0x08	Trigger Register	TSADCC_TRGR	Read-write	0x0000_0000
0x0C	Touch Screen Register	TSADCC_TSR	Read-write	0x0000_0000
0x10	Channel Enable Register	TSADCC_CHER	Write-only	–
0x14	Channel Disable Register	TSADCC_CHDR	Write-only	–
0x18	Channel Status Register	TSADCC_CHSR	Read-only	0x0000_0000
0x1C	Status Register	TSADCC_SR	Read-only	0x000C_0000
0x20	Last Converted Data Register	TSADCC_LCDR	Read-only	0x0000_0000
0x24	Interrupt Enable Register	TSADCC_IER	Write-only	–
0x28	Interrupt Disable Register	TSADCC_IDR	Write-only	–
0x2C	Interrupt Mask Register	TSADCC_IMR	Read-only	0x0000_0000
0x30	Channel Data Register 0	TSADCC_CDR0	Read-only	0x0000_0000
0x34	Channel Data Register 1	TSADCC_CDR1	Read-only	0x0000_0000
0x38	Channel Data Register 2	TSADCC_CDR2	Read-only	0x0000_0000
0x3C	Channel Data Register 3	TSADCC_CDR3	Read-only	0x0000_0000
0x40	Channel Data Register 4	TSADCC_CDR4	Read-only	0x0000_0000
0x44	Channel Data Register 5	TSADCC_CDR5	Read-only	0x0000_0000
0x48 - 0xFC	Reserved	–	–	–

## 43.10.1 TSADCC Control Register

**Register Name:** TSADCC\_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 TSADCC simulating a hardware reset.

- **START: Start Conversion**

0 = No effect.

1 = Begins analog-to-digital conversion.

## 43.10.2 TSADCC Mode Register

Register Name: TSADCC\_MR

Access Type: Read/Write

31	30	29	28	27	26	25	24
PENDBC				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
–	PENDET	SLEEP	LOWRES	–	–	TSAMOD	

- **TSAMOD: Touch Screen ADC Mode**

TSAMOD	Touch Screen ADC Operating Mode
0	ADC Mode
1	Touch Screen Only Mode
2	Reserved
3	Reserved

- **LOWRES: Resolution Selection**

LOWRES	Selected Resolution
0	10-bit resolution
1	8-bit resolution

This option is only valid in ADC mode.

- **SLEEP: Sleep Mode**

SLEEP	Selected Mode
0	Normal Mode
1	Sleep Mode

- **PRESCAL: Prescaler Rate Selection**

$$ADCCLK = MCK / ((PRESCAL+1) * 2)$$

- **PENDET: Pen Detect Selection**

0: Disable the Touch screen pins as analog inputs

1: enable the Touch screen pins as analog inputs

- **STARTUP: Start Up Time**

$$\text{Startup Time} = (\text{STARTUP}+1) * 8 / \text{ADCCLK}$$

- **SHTIM: Sample & Hold Time for ADC Channels**



Programming 0 for SHTIM gives a Sample & Hold Time equal to  $(2/ADCCLK)$ .

$$\text{Sample \& Hold Time} = (\text{SHTIM}+1) / \text{ADCCLK}$$

- **PENDBC: Pen Detect debouncing period**

$$\text{Period} = 2^{\text{PENDBC}} / \text{ADCCLK}$$

## 43.10.3 TSADCC Trigger Register

**Register Name:** TSADCC\_TRGR

**Access Type:** Read/Write

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

- **TRGMOD: Trigger Mode**

TRGMOD			Selected Trigger Mode
0	0	0	No trigger, only software trigger can start conversions
0	0	1	External Trigger Rising Edge
0	1	0	External Trigger Falling Edge
0	1	1	External Trigger Any Edge
1	0	0	Pen Detect Trigger (shall be selected only if PENDET is set and TSAMOD = Touch Screen only mode)
1	0	1	Periodic Trigger (TRGPER shall be initiated appropriately)
1	1	0	Continuous Mode
1	1	1	Reserved

- **TRGPER: Trigger Period**

Effective only if TRGMOD defines a Periodic Trigger

Defines the periodic trigger period, with the following equations:

$$\text{Trigger Period} = (\text{TRGPER} + 1) / \text{ADCCLK}$$

## 43.10.4 TSADCC Touch Screen Register

**Register Name:** TSADCC\_TSR

**Access Type:** Read/Write

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

- **TSSHTIM: Sample & Hold Time for Touch Screen Channels**

Programming 0 for TSSHTIM gives a Touch Screen Sample & Hold Time equal to (2/ADCCLK).

$$\text{Touch Screen Sample \& Hold Time} = (\text{TSSHTIM} + 1) / \text{ADCCLK}$$

## 43.10.5 TSADCC Channel Enable Register

**Register Name:** TSADCC\_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
–	–	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Enable**

0 = No effect.

1 = Enables the corresponding channel.

## 43.10.6 TSADCC Channel Disable Register

**Register Name:** TSADCC\_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
–	–	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 TSADCC\_SR are unpredictable.

## 43.10.7 TSADCC Channel Status Register

**Register Name:** TSADCC\_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
–	–	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Status**

0 = Corresponding channel is disabled.

1 = Corresponding channel is enabled.

## 43.10.8 TSADCC Status Register

**Register Name:** TSADCC\_SR

**Access Type:** Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	NOCNT	PENCNT	RXBUFF	ENDRX	GOVRE	DRDY
15	14	13	12	11	10	9	8
–	–	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	6	5	4	3	2	1	0
–	–	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 TSADCC\_SR.

1 = There has been an overrun error on the corresponding channel since the last read of TSADCC\_SR.

- **DRDY: Data Ready**

0 = No data has been converted since the last read of TSADCC\_LCDR.

1 = At least one data has been converted and is available in TSADCC\_LCDR.

- **GOVRE: General Overrun Error**

0 = No General Overrun Error occurred since the last read of TSADCC\_SR.

1 = At least one General Overrun Error has occurred since the last read of TSADCC\_SR.

- **ENDRX: End of RX Buffer**

0 = The Receive Counter Register has not reached 0 since the last write in TSADCC\_RCR or TSADCC\_RNCR.

1 = The Receive Counter Register has reached 0 since the last write in TSADCC\_RCR or TSADCC\_RNCR.

- **RXBUFF: RX Buffer Full**

0 = TSADCC\_RCR or TSADCC\_RNCR have a value other than 0.

1 = Both TSADCC\_RCR and TSADCC\_RNCR have a value of 0.

- **PENCNT: Pen Contact**

0 = No contact has been detected since the last read of TSADCC\_SR or PENDET is at 0.

1 = At least one contact has been detected since the last read of TSADCC\_SR.

- **NOCNT: No Contact**

0 = No contact loss has been detected since the last read of TSADCC\_SR or PENDET is at 0.

1 = At least one contact loss has been detected since the last read of TSADCC\_SR.

## 43.10.9 TSADCC Channel Data Register x (x = 0..5)

**Register Name:** TSADCC\_CDR0..TSADCC\_CDR5

**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: Channel Data**

The analog-to-digital conversion data is placed into this register at the end of a conversion of the corresponding channel and remains until a new conversion on the same channel is completed.

## 43.10.10 TSADCC Last Converted Data Register

**Register Name:** TSADCC\_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 on any analog channel and remains until a new conversion on any analog channel is completed.



## 43.10.11 TSADCC Interrupt Enable Register

**Register Name:** TSADCC\_IER

**Access Type:** Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	NOCNT	PENCNT	RXBUFF	ENDRX	GOVRE	DRDY
15	14	13	12	11	10	9	8
–	–	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	6	5	4	3	2	1	0
–	–	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**
- **PENCNT: Pen Contact**
- **NOCNT: No Contact**

0 = No effect.

1 = Enables the corresponding interrupt.

## 43.10.12 TSADCC Interrupt Disable Register

**Register Name:** TSADCC\_IDR

**Access Type:** Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	NOCNT	PENCNT	RXBUFF	ENDRX	GOVRE	DRDY
15	14	13	12	11	10	9	8
–	–	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	6	5	4	3	2	1	0
–	–	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**
- **PENCNT: Pen Contact**
- **NOCNT: No Contact**

0 = No effect.

1 = Disables the corresponding interrupt.

## 43.10.13 TSADCC Interrupt Mask Register

**Register Name:** TSADCC\_IMR

**Access Type:** Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	NOCNT	PENCNT	RXBUFF	ENDRX	GOVRE	DRDY
15	14	13	12	11	10	9	8
–	–	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	6	5	4	3	2	1	0
–	–	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
- **PENCNT:** Pen Contact
- **NOCNT:** No Contact

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.



## 44. AT91SAM9R64/RL64 Electrical Characteristics

### 44.1 Absolute Maximum Ratings

**Table 44-1.** Absolute Maximum Ratings\*

Operating Temperature (Industrial).....	-40°C to +125°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 and VDDBU).....	1.5V
Maximum Operating Voltage (VDDOSC, VDDPLL, VDDIOMx and VDDIOPx).....	4.0V
Total DC Output Current on all I/O lines .....	500 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.

### 44.2 DC Characteristics

The following characteristics are applicable to the operating temperature range:  $T_A = -40^{\circ}\text{C}$  to  $85^{\circ}\text{C}$ , unless otherwise specified.

**Table 44-2.** DC Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{VDDCORE}$	DC Supply Core		1.08	1.2	1.32	V
$V_{VDDBU}$	DC Supply Backup		1.08	1.2	1.32	V
$V_{VDDPLLA}$	DC Supply PLLA		3.0	3.3	3.6	V
$V_{VDDPLLB}$	DC Supply PLLB		1.08	1.2	1.32	V
$\Delta V_{VDDPLLB}$	Ripple on $V_{VDDPLLB}$	rms value, from 10k Hz to 10 MHz			10	mV
$I_{VDDPLLB}$	Current Supply	Normal operating mode			30	mA
$V_{VDDIOM}$	DC Supply Memory I/Os	Selectable by software in Bus Matrix	1.65	1.8	1.95	V
			3.0	3.3	3.6	V
$V_{VDDIOP}$	DC Supply Peripheral I/Os		3.0	3.3	3.6	V
$V_{VDDUTMII}$	USB UTMI+ Interface Power Supply		3.0	3.3	3.6	V
$V_{VDDUTMIC}$	USB UTMI+ Core Power Supply		1.08	1.2	1.32	V
$V_{VDDANA}$	ADC Analog Power Supply		3.0	3.3	3.6	V
$V_{IL}$	Input Low-level Voltage	$V_{VDDIO}$ from 3.0V to 3.6V	-0.3		0.8	V
		$V_{VDDIO}$ from 1.65V to 1.95V	-0.3		$0.3 \times V_{VDDIO}$	V

**Table 44-2. DC Characteristics (Continued)**

Symbol	Parameter	Conditions	Min	Typ	Max	Units	
V <sub>IH</sub>	Input High-level Voltage	V <sub>VDDIO</sub> = V <sub>VDDIOM</sub> or V <sub>VDDIOP</sub> from 3.0V to 3.6V	2.0		V <sub>VDDIO</sub> +0.3V	V	
		V <sub>VDDIO</sub> = V <sub>VDDIOM</sub> or V <sub>VDDIOP</sub> from 1.65V to 1.95V	0.7 x V <sub>VDDIO</sub>		V <sub>VDDIO</sub> +0.3V	V	
V <sub>OL</sub>	Output Low-level Voltage	I <sub>O</sub> max, V <sub>VDDIO</sub> from 3.0V to 3.6V			0.4	V	
		I <sub>O</sub> max, V <sub>VDDIO</sub> from 1.65V to 1.95V			0.25 x V <sub>VDDIO</sub>	V	
V <sub>OH</sub>	Output High-level Voltage	I <sub>O</sub> max, V <sub>VDDIO</sub> from 3.0V to 3.6V	V <sub>VDDIO</sub> - 0.4			V	
		I <sub>O</sub> max, V <sub>VDDIO</sub> from 1.65V to 1.95V	0.75 x V <sub>VDDIO</sub>			V	
R <sub>PULLUP</sub>	Pull-up Resistance	PA0-PA31, PB0-PB31, PC0-PC31, PD0-PD21	70	100	175	kOhm	
I <sub>O</sub>	Output Current	PA0-PA31, PB0-PB31, PC0-PC31, PD0-PD21			8	mA	
I <sub>SC</sub>	Static Current	On V <sub>VDDCORE</sub> = 1.2V, MCK = 0 Hz, excluding POR	T <sub>A</sub> =25°C	175			μA
		All inputs driven TMS, TDI, TCK, NRST = 1	T <sub>A</sub> =85°C		1400		
		On V <sub>VDDBU</sub> = 1.2V, Logic cells consumption, excluding POR	T <sub>A</sub> =25°C	4			nA
		All inputs driven WKUP = 0	T <sub>A</sub> =85°C			16	

### 44.3 Power Consumption

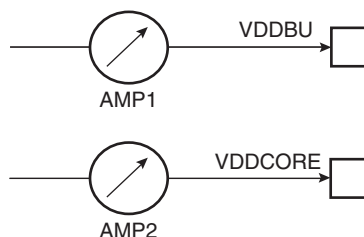
- Typical power consumption of PLLs, Slow Clock and Main Oscillator.
- Power consumption of power supply in four different modes: Active, Idle, Ultra Low-power and Backup.
- Power consumption by peripheral: calculated as the difference in current measurement after having enabled then disabled the corresponding clock.

#### 44.3.1 Power Consumption versus Modes

The values in [Table 44-3](#) and [Table 44-4 on page 863](#) are measured values of the power consumption with operating conditions as follows:

- V<sub>VDDIOM</sub> = V<sub>VDDIOP</sub> = 3.3 V
- V<sub>VDDPLLA</sub> = 3.3V
- V<sub>VDDCORE</sub> = V<sub>VDDBU</sub> = V<sub>VDDPLLB</sub> = 1.2V
- V<sub>VDDANA</sub> = 3.3V
- There is no consumption on the I/Os of the device

**Figure 44-1.** Measures Schematics



These figures represent the power consumption measured on the power supplies.

**Table 44-3.** Power Consumption for Different Modes

Mode	Conditions	Consumption	Unit
Active	ARM Core clock is 200MHz. MCK is 100MHz. All peripheral clocks activated. lcache enabled. on AMP2	47	mA
Idle	Idle state, waiting an interrupt. All peripheral clocks de-activated. on AMP2	12	mA
Ultra low power	ARM Core clock is 500 Hz. All peripheral clocks de-activated. on AMP2	175	$\mu$ A
Backup	Device only VDDDBU powered on AMP1	4	$\mu$ A

**Table 44-4.** Power Consumption by Peripheral in Active Mode on VDDCORE

Peripheral	Consumption	Unit
PIO Controller	5.5	$\mu$ A/MHz
USART	12	
UDPHS on AMP2	44	
TWI	6	
SPI	8.5	
MCI	15.5	
SSC	14	
Timer Counter Channels	4	
PWMC	6.5	
LCDC	4.5	
ADC/TSC	7.5	
AC97	11.5	

## 44.4 Crystal Oscillator Characteristics

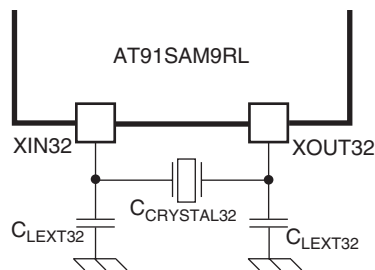
The following characteristics are applicable to the operating temperature range:  $T_A = -40^{\circ}\text{C}$  to  $85^{\circ}\text{C}$  and worst case of power supply, unless otherwise specified.

### 44.4.1 32 kHz Oscillator Characteristics

**Table 44-5.** 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	$C_{CRYSTAL32} = 6\text{pF}^{(3)}$		4		
		$C_{CRYSTAL32} = 12.5\text{pF}^{(3)}$		17		
	Duty Cycle		40		60	%
$t_{ST}$	Startup Time	$R_S = 50\text{ k}\Omega, C_L = 6\text{pF}^{(1)}$			400	ms
		$R_S = 50\text{ k}\Omega, C_L = 12.5\text{ pF}^{(1)}$			900	
		$R_S = 100\text{ k}\Omega, C_L = 6\text{pF}^{(1)}$			600	
		$R_S = 100\text{ k}\Omega, C_L = 12.5\text{ pF}^{(1)}$			1200	
$I_{DD\text{ ON}}$	Current Dissipation	$R_S = 50\text{ k}\Omega, C_L = 6\text{pF}^{(1)}$		0.65	1.6	$\mu\text{A}$
		$R_S = 50\text{ k}\Omega, C_L = 12.5\text{ pF}^{(1)}$		0.85	1.8	
		$R_S = 100\text{ k}\Omega, C_L = 6\text{pF}^{(1)}$		1	2.2	
		$R_S = 100\text{ k}\Omega, C_L = 12.5\text{ pF}^{(1)}$		1.2	2.4	

- 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 user load capacitance should be subtracted from  $C_{LEXT32}$ .



**Table 44-6.** 32 kHz Crystal Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
ESR	Equivalent Series Resistor $R_s$	Crystal @ 32.768 kHz		50	100	$\text{k}\Omega$
$C_M$	Motional Capacitance	Crystal @ 32.768 kHz	0.6		3	fF
$C_S$	Shunt Capacitance	Crystal @ 32.768 kHz	0.6		2	pF



## 44.4.2 RC Oscillator Characteristics

**Table 44-7.** RC Oscillator Characteristics

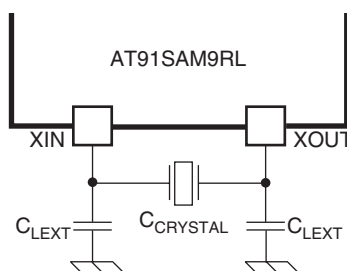
Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CPRCz})$	Crystal Oscillator Frequency		20	32	44	kHz
	Duty Cycle		45	50	55	%
$t_{ST}$	Startup Time				75	$\mu s$
$I_{DD}$	Current Consumption	After Startup Time		0.7	1.0	$\mu A$

## 44.4.3 12 MHz Main Oscillator Characteristics

**Table 44-8.** 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	$C_{CRYSTAL} = 15 \text{ pF}^{(1)}$		24		pF
		$C_{CRYSTAL} = 20 \text{ pF}^{(1)}$		34		
	Duty Cycle		40	50	60	%
$t_{ST}$	Startup Time				2	ms
$I_{DDST}$	Standby Current Consumption	Standby mode			1	$\mu A$
$P_{ON}$	Drive Level				150	$\mu W$
$I_{DD ON}$	Current Dissipation	@ 12MHz		550	950	$\mu A$
$I_{BYPASS}$	Bypass Current Dissipation			3.3	5	$\mu W/MHz$

Notes: 1. Additional user load capacitance should be subtracted from  $C_{LEXT}$ .



#### 44.4.4 XIN Clock Characteristics

**Table 44-9.** XIN Clock Electrical Characteristics

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPXIN})$	XIN Clock Frequency			50.0	MHz
$t_{CPXIN}$	XIN Clock Period		20.0		ns
$t_{CHXIN}$	XIN Clock High Half-period		$0.4 \times t_{CPXIN}$	$0.6 \times t_{CPXIN}$	
$t_{CLXIN}$	XIN Clock Low Half-period		$0.4 \times t_{CPXIN}$	$0.6 \times t_{CPXIN}$	
$C_{IN}$	XIN Input Capacitance	(1)		25	pF
$R_{IN}$	XIN Pulldown Resistor	(1)		500	k $\Omega$

Note: 1. These characteristics apply only when the Main Oscillator is in bypass mode (i.e., when MOSCEN = 0 and OSCBYPASS = 1 in the CKGR\_MOR register).

#### 44.4.5 Crystal Characteristics

**Table 44-10.** Crystal Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
ESR	Equivalent Series Resistor Rs				80	$\Omega$
$C_M$	Motional Capacitance		5		9	fF
$C_S$	Shunt Capacitance				7	pF

#### 44.4.6 PLLA Characteristics

**Table 44-11.** Phase Lock Loop Characteristics<sup>(1)</sup>

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$F_{IN}$	Input Frequency		1		32	MHz
$F_{OUT}$	Output Frequency	Field OUT of CKGR_PLL is 00	80		200	MHz
		Field OUT of CKGR_PLL is 10	190		240	MHz
$I_{PLL}$	Current Consumption	active mode			3	mA
		standby mode			1	$\mu$ A

Note: 1. Startup time depends on PLL RC filter. A calculation tool is provided by Atmel.

## 44.4.7 UTMI PLL Characteristics

**Table 44-12.** Phase Lock Loop Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$F_{IN}$	Input Frequency		4	12	32	MHz
$F_{OUT}$	Output Frequency		450	480	600	MHz
$I_{PLL}$	Current Consumption	active mode		5	8	mA
		standby mode			TBD	$\mu$ A

## 44.5 USB HS Characteristics

### 44.5.1 Electrical Characteristics

**Table 44-13.** Electrical Parameters

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$R_{PUI}$	Bus Pull-up Resistor on Upstream Port (idle bus)	in LS or FS Mode		1.5		kOhm
$R_{PUA}$	Bus Pull-up Resistor on Upstream Port (upstream port receiving)	in LS or FS Mode		15		kOhm
Settling time						
$T_{BIAS}$	Bias settling time				20	$\mu$ s
$T_{OSC}$	Oscillator settling time	With 12 MHz crystal			2	ms
$T_{SETTLING}$	Settling time	$F_{IN} = 12$ MHz		0.3	0.5	ms

### 44.5.2 Static Power Consumption

**Table 44-14.** Static Power Consumption

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$I_{BIAS}$	Bias current consumption on VBG				1	$\mu$ A
$I_{VDDUTMII}$	HS Transceiver and I/O current consumption				8	$\mu$ A
	LS / FS Transceiver and I/O current consumption	no connection <sup>(1)</sup>			3	$\mu$ A
$I_{VDDUTMIC}$	Core, PLL, and Oscillator current consumption				2	$\mu$ A

Note: 1. If cable is connected, add 200  $\mu$ A (Typical) due to Pull-up/Pull-down current consumption.

### 44.5.3 Dynamic Power Consumption

**Table 44-15.** Dynamic Power Consumption

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$I_{BIAS}$	Bias current consumption on VBG			0.7	0.8	mA
$I_{VDDUTMII}$	HS Transceiver current consumption	HS transmission		47	60	mA
	HS Transceiver current consumption	HS reception		18	27	mA
	LS / FS Transceiver current consumption	FS transmission 0m cable <sup>(1)</sup>		4	6	mA
	LS / FS Transceiver current consumption	FS transmission 5m cable <sup>(1)</sup>		26	30	mA
	LS / FS Transceiver current consumption	FS reception <sup>(1)</sup>		3	4.5	mA
$I_{VDDUTMIC}$	PLL, Core and Oscillator current consumption			5.5	9	mA

Note: 1. Including 1 mA due to Pull-up/Pull-down current consumption.

### 44.5.4 ADC

**Table 44-16.** Channel Conversion Time and ADC Clock

Parameter	Conditions	Min	Typ	Max	Units
ADC Clock Frequency	10-bit resolution mode			4	MHz
Startup Time	Return from Idle Mode			40	μs
Track and Hold Acquisition Time	ADC Clock = 4 MHz <sup>(1)</sup>	1.0			μs
Conversion Time	ADC Clock = 4 MHz <sup>(1)</sup>		3.33		μs
Throughput Rate	ADC Clock = 4 MHz <sup>(1)</sup>			220	kSPS

Note: 1. In worst case, the Track-and-Hold Acquisition Time is given by:

$$T_{TH} \text{ (ns)} = 500 + (0,08 \times Z_{IN})(\text{Ohm})$$

The full speed is obtained for  $t_{TH} = 1000$  ns with an input source impedance of 6250Ohm max, corresponding to 8 clock periods at maximum clock frequency.

To achieve optimal performance of the ADC, the analog power supply VDDANA and the ADVREF input voltage must be decoupled with a 4.7μF capacitor in parallel with a 100 nF capacitor.

**Table 44-17.** External Voltage Reference Input

Parameter	Conditions	Min	Typ	Max	Units
ADVREF Input Voltage Range		2.6	3.3	VDDANA	V
ADVREF Average Current				1100	μA
Current Consumption on VDDANA				200	μA

**Table 44-18.** Analog Inputs

Parameter	Min	Typ	Max	Units
Input Voltage Range	0		ADVREF	V
Input Leakage Current			1	μA
Input Capacitance		4	6	pF
Input Impedance		50	6250	Ohm

**Table 44-19.** Transfer Characteristics

Parameter	Conditions	Min	Typ	Max	Units
Resolution			10		bit
Integral Non-linearity				±2	LSB
Differential Non-linearity	No missing code			±0.9	LSB
Offset Error		-1.5	0.5	+2.5	LSB
Gain Error				±2	LSB

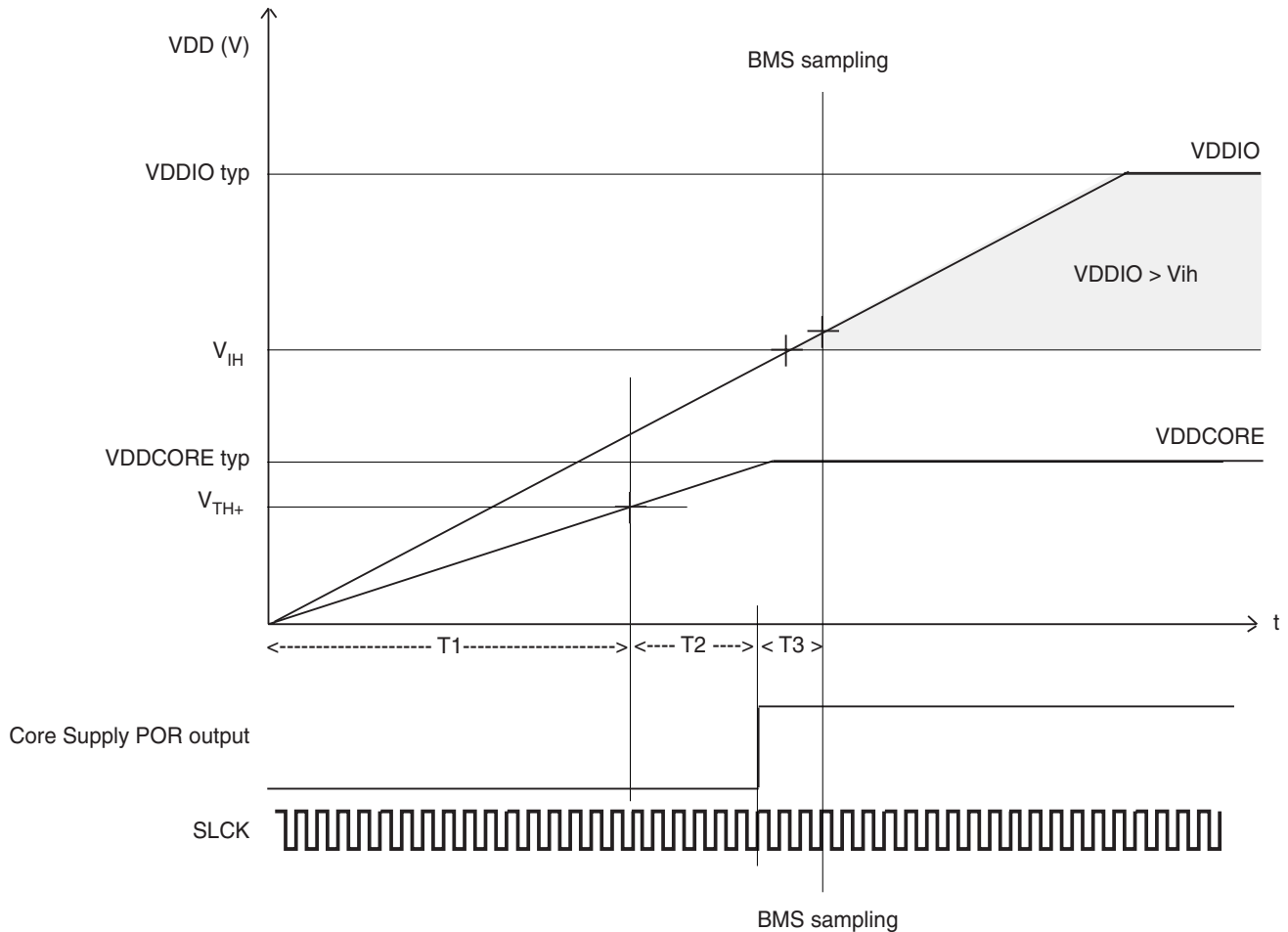
Note: 1. For more information on data converter terminology, refer to the Atmel application note [Data Converter Terminology, lit. no. 6022](#).

## 44.6 Core Power Supply POR Characteristics

**Table 44-20.** Power-On-Reset Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{TH+}$	Threshold Voltage Rising	Minimum Slope of +2.0V/30ms	0.70	0.85	1.08	V
$V_{TH-}$	Threshold Voltage Falling		0.60	0.80	1.00	V
$T_{RES}$	Reset Time		80	150	230	$\mu$ s

**Figure 44-2.**  $V_{VDDCORE}$  and  $V_{VDDIO}$  Constrains at Startup



Value of signals powered by VDDIO (such as BMS, NRST or I/O lines) will be unpredictable if read when  $V_{VDDIO}$  is lower than  $V_{IH}$ . For correct operations VDDIO must have a minimum slope at startup higher than  $V_{IH}$  when BMS is sampled. After this, all signals will be correctly read.

- Notes:
1. VDDIO stands for VDDIOPx or VDDIOMy.
  2. VDDCORE typ, VDDIO typ and  $V_{IH}$  are defined in [Table 44-2, "DC Characteristics"](#).
  3. T1 is given by the slope on VDDCORE power supply.
  4.  $T2 = t_{RES}$
  5.  $T3 = 3 \times t_{SLCK}$

## 44.7 Timings

### 44.7.1 Corner Definition

**Table 44-21.** Corner Definition

Corner	Process	Temp (External)	VDDCORE: 1.2V	VDDIO: 1.8V	VDDIO: 3.3V
MAX	Slow	85°C	1.10	1.65	3.0
STH	Slow	85°C	1.2	1.8	3.3
MIN	Fast	-40°C	1.32	1.95	3.6

### 44.7.2 Processor Clock

**Table 44-22.** Processor Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPPCK})$	Processor Clock Frequency	MAX corner		200	MHz
$1/(t_{CPPCK})$	Processor Clock Frequency	STH corner		243	MHz

### 44.7.3 Master Clock

**Table 44-23.** Master Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPMCK})$	Master Clock Frequency	MAX corner		100	MHz
$1/(t_{CPMCK})$	Master Clock Frequency	STH corner		121	MHz

### 44.7.4 I/Os

Criteria used to define the maximum frequency of the I/Os:

- output duty cycle (40%-60%)
- minimum output swing: 100 mV to VDDIO - 100 mV
- Addition of rising and falling time inferior to 75% of the period

**Table 44-24.** I/O Characteristics

Symbol	Parameter	Conditions	Min	Max	Units
FreqMax	VDDIOP powered pins frequency	3.3V domain <sup>(1)</sup>		TBD (100)	MHz

Note: 1. 3.3V domain:  $V_{VDDIOP}$  from 3.0V to 3.6V, maximum external capacitor = 40 pF.



#### 44.7.5 SMC

SMC timings are given in MAX (T = 85°C, VDDCORE = 1.08V) and STH (T = 85°C, VDDCORE = 1.2V) corners.

Timings are given assuming a capacitance load on data, control and address pads:

**Table 44-25.** Capacitance Load

IO Supply	Corner	
	MAX	STH
3.3V	50 pF	50 pF
1.8V	30 pF	30 pF

In following tables  $t_{CPMCK}$  is MCK period.

##### 44.7.5.1 Read Timings

**Table 44-26.** SMC Read Signals - NRD Controlled (READ\_MODE = 1)

Symbol	Parameter	Min in STH corner		Min in MAX corner		Units
		1.8V	3.3V	1.8V	3.3V	
NO HOLD SETTINGS (nrd hold = 0)						
SMC <sub>1</sub>	Data Setup before NRD High	10.9	10.1	12.1	11.4	ns
SMC <sub>2</sub>	Data Hold after NRD High	0	0	0	0	ns
HOLD SETTINGS (nrd hold ≠ 0)						
SMC <sub>3</sub>	Data Setup before NRD High	8.8	8.0	9.6	8.8	ns
SMC <sub>4</sub>	Data Hold after NRD High	0	0	0	0	ns
HOLD or NO HOLD SETTINGS (nrd hold ≠ 0, nrd hold = 0)						
SMC <sub>5</sub>	NBS0/A0, NBS1, NBS2/A1, NBS3, A2 - A25 Valid before NRD High	(nrd setup + nrd pulse)* $t_{CPMCK} - 1.9$	(nrd setup + nrd pulse)* $t_{CPMCK} - 1.9$	(nrd setup + nrd pulse)* $t_{CPMCK} - 2.3$	(nrd setup + nrd pulse)* $t_{CPMCK} - 2.3$	ns
SMC <sub>6</sub>	NCS low before NRD High	(nrd setup + nrd pulse - ncs rd setup)* $t_{CPMCK} - 1.6$	(nrd setup + nrd pulse - ncs rd setup)* $t_{CPMCK} - 3.2$	(nrd setup + nrd pulse - ncs rd setup)* $t_{CPMCK} - 2.2$	(nrd setup + nrd pulse - ncs rd setup)* $t_{CPMCK} - 3.6$	ns
SMC <sub>7</sub>	NRD Pulse Width	nrd pulse* $t_{CPMCK} - 0.2$	nrd pulse* $t_{CPMCK} - 0.3$	nrd pulse* $t_{CPMCK} - 0.3$	nrd pulse* $t_{CPMCK} - 0.4$	ns

**Table 44-27.** SMC Read Signals - NCS Controlled (Read\_MODE = 0)

Symbol	Parameter	Min in STH corner		Min in MAX corner		Units
		1.8V	3.3V	1.8V	3.3V	
NO HOLD SETTINGS (ncs rd hold = 0)						
SMC <sub>8</sub>	Data Setup before NCS High	11.2	10.4	12.5	12.6	ns
SMC <sub>9</sub>	Data Hold after NCS High	0	0	0	0	ns



**Table 44-27.** SMC Read Signals - NCS Controlled (Read\_MODE = 0) (Continued)

Symbol	Parameter	Min in STH corner		Min in MAX corner		Units
		1.8V	3.3V	1.8V	3.3V	
	VDDIOM supply	1.8V	3.3V	1.8V	3.3V	
HOLD SETTINGS (ncs rd hold ≠ 0)						
SMC <sub>10</sub>	Data Setup before NCS High	9.1	8.4	9.8	10.0	ns
SMC <sub>11</sub>	Data Hold after NCS High	0	0	0	0	ns
HOLD or NO HOLD SETTINGS (ncs rd hold ≠ 0, ncs rd hold = 0)						
SMC <sub>12</sub>	NBS0/A0, NBS1, NBS2/A1, NBS3, A2 - A25 valid before NCS High	(ncs rd setup + ncs rd pulse)* t <sub>CPMCK</sub> - 5.0	(ncs rd setup + ncs rd pulse)* t <sub>CPMCK</sub> - 4.8	(ncs rd setup + ncs rd pulse)* t <sub>CPMCK</sub> - 5.1	(ncs rd setup + ncs rd pulse)* t <sub>CPMCK</sub> - 4.8	ns
SMC <sub>13</sub>	NRD low before NCS High	(ncs rd setup + ncs rd pulse - nrd setup)* t <sub>CPMCK</sub> - 2.8	(ncs rd setup + ncs rd pulse - nrd setup)* t <sub>CPMCK</sub> - 2.6	(ncs rd setup + ncs rd pulse - nrd setup)* t <sub>CPMCK</sub> - 2.4	(ncs rd setup + ncs rd pulse - nrd setup)* t <sub>CPMCK</sub> - 2.1	ns
SMC <sub>14</sub>	NCS Pulse Width	ncs rd pulse length * t <sub>CPMCK</sub> - 1.0	ncs rd pulse length * t <sub>CPMCK</sub> - 1.0	ncs rd pulse length * t <sub>CPMCK</sub> - 1.1	ncs rd pulse length * t <sub>CPMCK</sub> - 1.1	ns

#### 44.7.5.2 Write Timings

**Table 44-28.** SMC Write Signals - NWE controlled (WRITE\_MODE = 1)

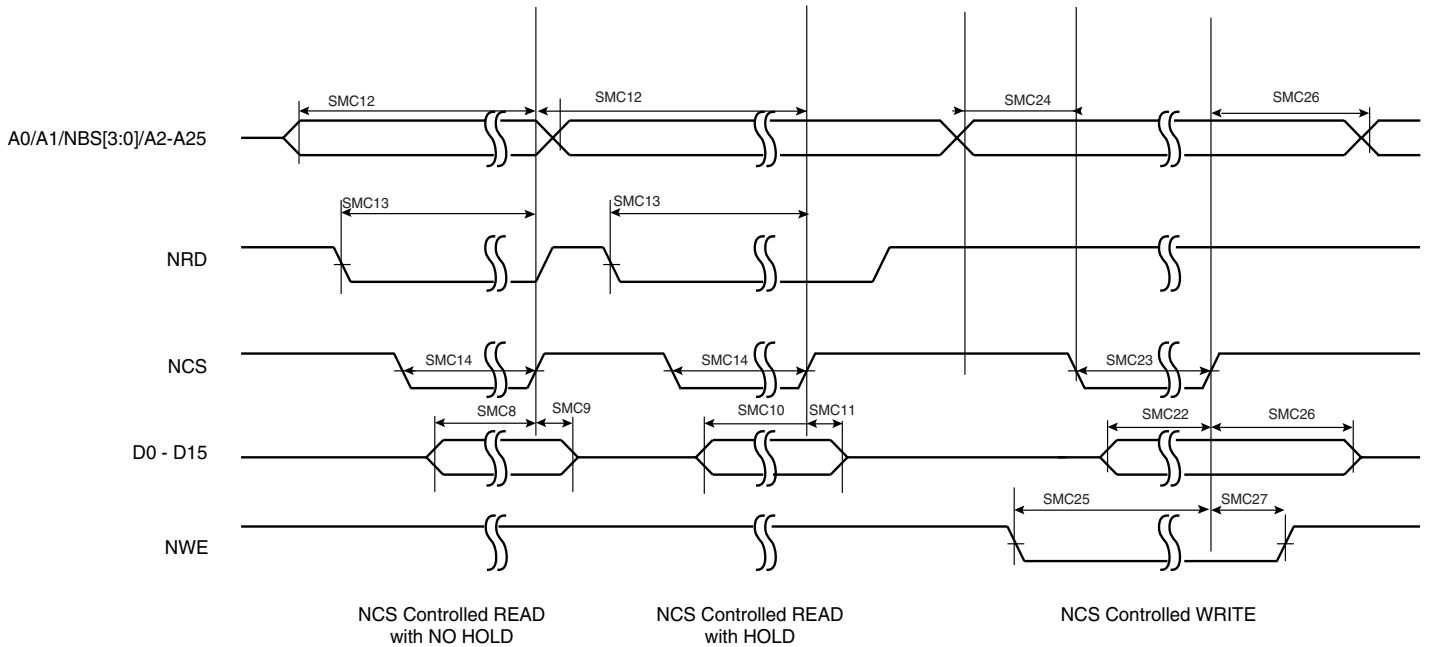
Symbol	Parameter	Min in STH corner		Min in MAX corner		Units
		1.8V Supply	3.3V Supply	1.8V Supply	3.3V Supply	
HOLD or NO HOLD SETTINGS (nwe hold ≠ 0, nwe hold = 0)						
SMC <sub>15</sub>	Data Out Valid before NWE High	nwe pulse * t <sub>CPMCK</sub> - 1.8	nwe pulse * t <sub>CPMCK</sub> - 2.9	nwe pulse * t <sub>CPMCK</sub> - 2.2	nwe pulse * t <sub>CPMCK</sub> - 3.4	ns
SMC <sub>16</sub>	NWE Pulse Width	nwe pulse * t <sub>CPMCK</sub> - 0.3	nwe pulse * t <sub>CPMCK</sub> - 0.3	nwe pulse * t <sub>CPMCK</sub> - 0.4	nwe pulse * t <sub>CPMCK</sub> - 9.9	ns
SMC <sub>17</sub>	NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25 valid before NWE low	nwe setup * t <sub>CPMCK</sub> - 2.2	nwe setup * t <sub>CPMCK</sub> - 3.8	nwe setup * t <sub>CPMCK</sub> - 2.9	nwe setup * t <sub>CPMCK</sub> - 4.4	ns
SMC <sub>18</sub>	NCS low before NWE high	(nwe setup - ncs rd setup + nwe pulse) * t <sub>CPMCK</sub> - 1.8	(nwe setup - ncs rd setup + nwe pulse) * t <sub>CPMCK</sub> - 3.4	(nwe setup - ncs rd setup + nwe pulse) * t <sub>CPMCK</sub> - 2.5	(nwe setup - ncs rd setup + nwe pulse) * t <sub>CPMCK</sub> - 3.9	ns
HOLD SETTINGS (nwe hold ≠ 0)						
SMC <sub>19</sub>	NWE High to Data OUT, NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25 change	nwe hold * t <sub>CPMCK</sub> - 3.7	nwe hold * t <sub>CPMCK</sub> - 3.4	nwe hold * t <sub>CPMCK</sub> - 3.5	nwe hold * t <sub>CPMCK</sub> - 3.2	ns
SMC <sub>20</sub>	NWE High to NCS Inactive <sup>(1)</sup>	(nwe hold - ncs wr hold)* t <sub>CPMCK</sub> - 3.3	(nwe hold - ncs wr hold)* t <sub>CPMCK</sub> - 3.1	(nwe hold - ncs wr hold)* t <sub>CPMCK</sub> - 3.0	(nwe hold - ncs wr hold)* t <sub>CPMCK</sub> - 2.8	ns
NO HOLD SETTINGS (nwe hold = 0)						
SMC <sub>21</sub>	NWE High to Data OUT, NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25, NCS change <sup>(1)</sup>	5.7	7.9	5.8	7.2	ns

Note: 1. hold length = total cycle duration - setup duration - pulse duration. "hold length" is for "ncs wr hold length" or "NWE hold length".

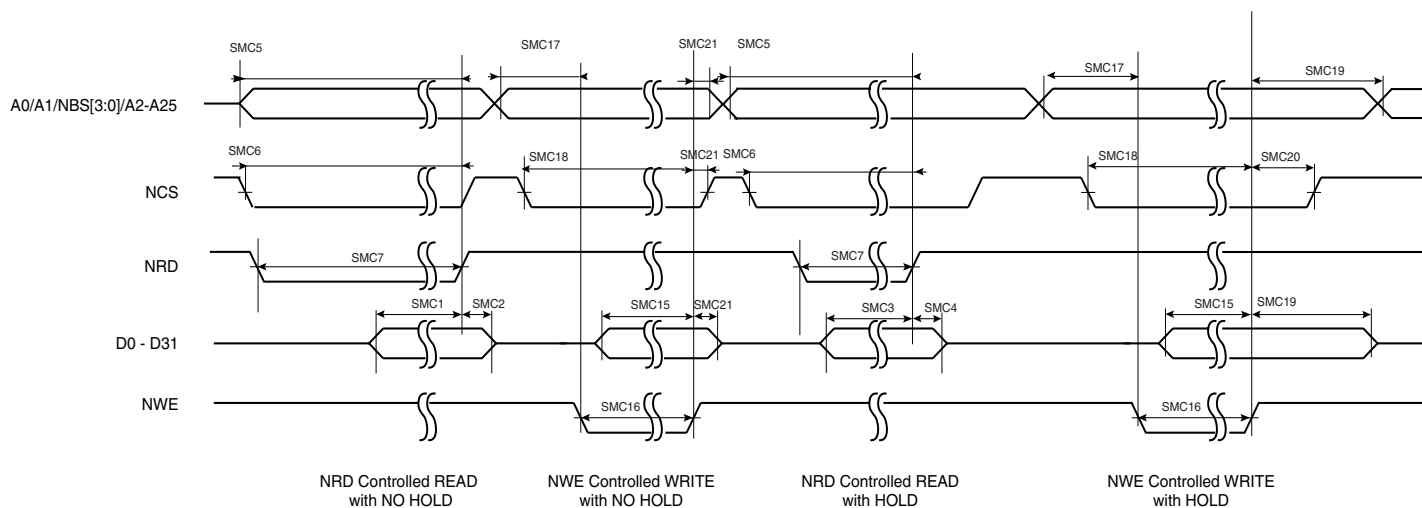
**Table 44-29.** SMC Write NCS Controlled (WRITE\_MODE = 0)

Symbol	Parameter	Min in STH corner		Min in MAX corner		Units
		1.8V Supply	3.3V Supply	1.8V Supply	3.3V Supply	
SMC <sub>22</sub>	Data Out Valid before NCS High	ncs wr pulse * t <sub>CPMCK</sub> - 4.6	ncs wr pulse * t <sub>CPMCK</sub> - 5.5	ncs wr pulse * t <sub>CPMCK</sub> - 4.6	ncs wr pulse * t <sub>CPMCK</sub> - 5.6	ns
SMC <sub>23</sub>	NCS Pulse Width	ncs wr pulse * t <sub>CPMCK</sub> - 4.6	ncs wr pulse * t <sub>CPMCK</sub> - 6.0	ncs wr pulse * t <sub>CPMCK</sub> - 4.9	ncs wr pulse * t <sub>CPMCK</sub> - 6.2	ns
SMC <sub>24</sub>	NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25 valid before NCS low	ncs wr setup * t <sub>CPMCK</sub> - 2.3	ncs wr setup * t <sub>CPMCK</sub> - 2.3	ncs wr setup * t <sub>CPMCK</sub> - 2.8	ncs wr setup * t <sub>CPMCK</sub> - 2.8	ns
SMC <sub>25</sub>	NWE low before NCS high	(ncs wr setup - nwe setup + ncs pulse)* t <sub>CPMCK</sub> - 3.1	(ncs wr setup - nwe setup + ncs pulse)* t <sub>CPMCK</sub> - 2.8	(ncs wr setup - nwe setup + ncs pulse)* t <sub>CPMCK</sub> - 2.8	(ncs wr setup - nwe setup + ncs pulse)* t <sub>CPMCK</sub> - 2.5	ns
SMC <sub>26</sub>	NCS High to Data Out, NBS0/A0, NBS1, NBS2/A1, NBS3, A2 - A25, change	ncs wr hold * t <sub>CPMCK</sub> - 3.7	ncs wr hold * t <sub>CPMCK</sub> - 3.5	ncs wr hold * t <sub>CPMCK</sub> - 3.6	ncs wr hold * t <sub>CPMCK</sub> - 4.1	ns
SMC <sub>27</sub>	NCS High to NWE Inactive	(ncs wr hold - nwe hold)* t <sub>CPMCK</sub> - 0.5	(ncs wr hold - nwe hold)* t <sub>CPMCK</sub> - 0.6	(ncs wr hold - nwe hold)* t <sub>CPMCK</sub> - 0.7	(ncs wr hold - nwe hold)* t <sub>CPMCK</sub> - 1.5	ns

**Figure 44-3.** SMC Timings - NCS Controlled Read and Write



**Figure 44-4.** SMC Timings - NRD Controlled Read and NWE Controlled Write



## 44.7.6 SDRAMC

Timings are given assuming a capacitance load on data, control and address pads:

**Table 44-30.** Capacitance Load on Data, Control and Address Pads

IO Supply	Corner	
	MAX	STH
3.3V	50 pF	50 pF
1.8V	30 pF	30 pF

**Table 44-31.** Capacitance Load on SDCK Pad

IO Supply	Corner	
	MAX	STH
3.3V	10 pF	10 pF
1.8V	10 pF	10 pF

The SDRAM Controller satisfies the timings of standard PC100, PC133 (3.3V supply) and Mobile SDRAM (1.8V supply) which are given in [Table 44-32](#), [Table 44-33](#) and [Table 44-34](#).

**Table 44-32.** SDRAM PC100 Characteristics

Parameter	Min	Max	Units
	3.3V Supply	3.3V Supply	
SDRAM Controller Clock Frequency		100	MHz
Control/Address/Data In Setup <sup>(1)(2)</sup>	2		ns
Control/Address/Data In Hold <sup>(1)(2)</sup>	1		ns
Data Out Access time after SDCK rising		6	ns
Data Out change time after SDCK rising	3		ns

**Table 44-33. SDRAM PC133 Characteristics**

Parameter	Min	Max	Units
	3.3V Supply	3.3V Supply	
SDRAM Controller Clock Frequency		<sup>(4)</sup> 133	MHz
Control/Address/Data In Setup <sup>(1)(2)</sup>	1.5		ns
Control/Address/Data In Hold <sup>(1)(2)</sup>	0.8		ns
Data Out Access time after SDCK rising		5.4	ns
Data Out change time after SDCK rising	3.0		ns

**Table 44-34. Mobile Characteristics**

Parameter	Min	Max	Units
	1.8V Supply	1.8V Supply	
SDRAM Controller Clock Frequency		<sup>(4)</sup> 133/100 <sup>(3)</sup>	MHz
Control/Address/Data In Setup <sup>(1)(2)</sup>	1.5		ns
Control/Address/Data In Hold <sup>(1)(2)</sup>	1		ns
Data Out Access time after SDCK rising		6/8 <sup>(3)</sup>	ns
Data Out change time after SDCK rising	2.5		ns

- Notes:
- Control is the set of following signals: SDCKE, SDCS, RAS, CAS, SDA10, BAx, DQMx, and SDWE.
  - Address is the set of A0-A9, A11-A13.
  - 133 MHz with CL=3, 100 MHz with CL=2.
  - Minimum of 133 MHz and MCK frequency for the operating conditions ([Table 44-2](#))

#### 44.7.7 EMAC

Timings are given assuming a capacitance load on data and clock:

**Table 44-35. Capacitance Load on Data, Clock Pads**

IO Supply	Corner	
	MAX	STH
3.3V	20 pF	20 pF
1.8V	20 pF	20 pF

The Ethernet Controller satisfies the timings of standard in MAX and STH corners.

**Table 44-36. EMAC Signals Relative to EMDC**

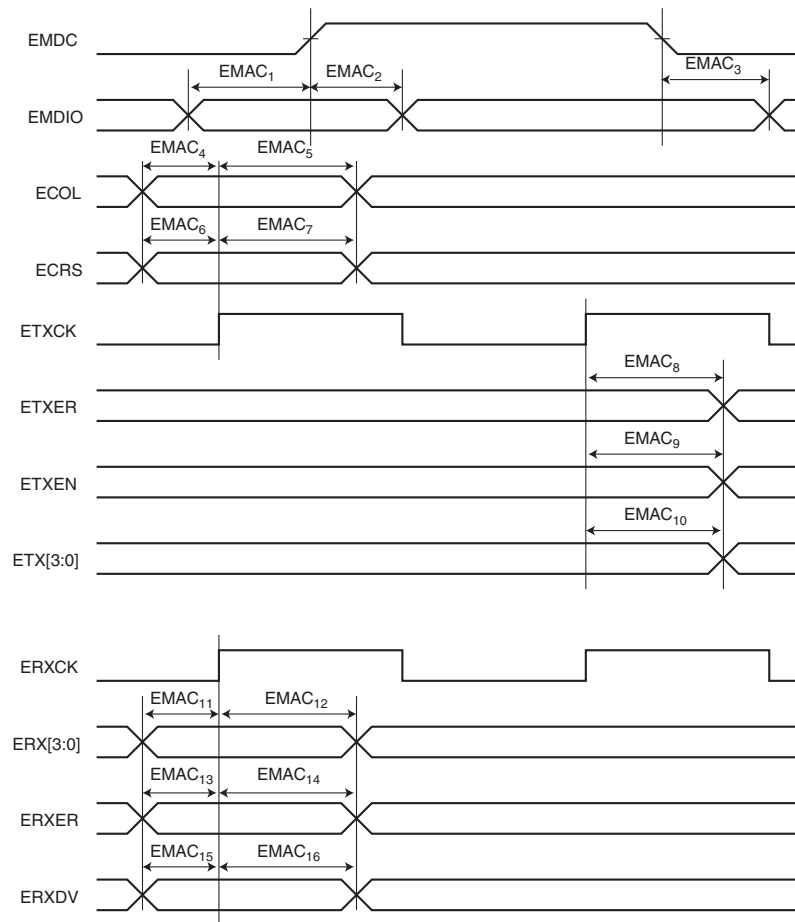
Symbol	Parameter	Min (ns)	Max (ns)
EMAC <sub>1</sub>	Setup for EMDIO from EMDC rising	10 ns	
EMAC <sub>2</sub>	Hold for EMDIO from EMDC rising	10 ns	
EMAC <sub>3</sub>	EMDIO toggling from EMDC rising	0 ns	300 ns

## 44.7.7.1 MII Mode

**Table 44-37.** EMAC MII Specific Signals

Symbol	Parameter	Min (ns)	Max (ns)
EMAC <sub>4</sub>	Setup for ECOL from ETXCK rising	10	
EMAC <sub>5</sub>	Hold for ECOL from ETXCK rising	10	
EMAC <sub>6</sub>	Setup for ECRS from ETXCK rising	10	
EMAC <sub>7</sub>	Hold for ECRS from ETXCK rising	10	
EMAC <sub>8</sub>	ETXER toggling from ETXCK rising	10	25
EMAC <sub>9</sub>	ETXEN toggling from ETXCK rising	10	25
EMAC <sub>10</sub>	ETX toggling from ETXCK rising	10	25
EMAC <sub>11</sub>	Setup for ERX from ERXCK	10	
EMAC <sub>12</sub>	Hold for ERX from ERXCK	10	
EMAC <sub>13</sub>	Setup for ERXER from ERXCK	10	
EMAC <sub>14</sub>	Hold for ERXER from ERXCK	10	
EMAC <sub>15</sub>	Setup for ERXDV from ERXCK	10	
EMAC <sub>16</sub>	Hold for ERXDV from ERXCK	10	

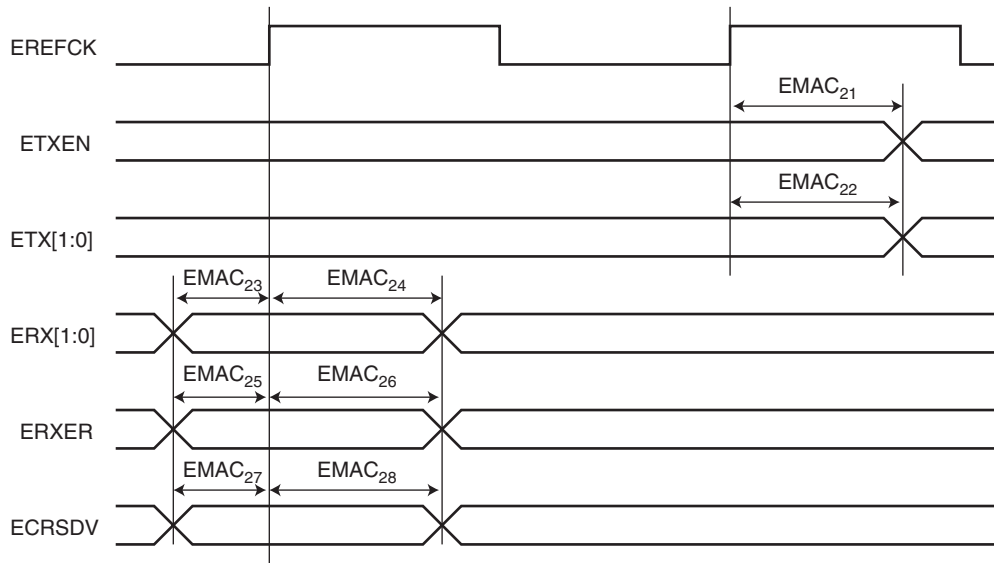
**Figure 44-5. EMAC MII Mode**



**Table 44-38. RMII Mode**

Symbol	Parameter	Min (ns)	Max (ns)
EMAC <sub>21</sub>	ETXEN toggling from EREFCK rising	2	16
EMAC <sub>22</sub>	ETX toggling from EREFCK rising	2	16
EMAC <sub>23</sub>	Setup for ERX from EREFCK rising	4	
EMAC <sub>24</sub>	Hold for ERX from EREFCK rising	2	
EMAC <sub>25</sub>	Setup for ERXER from EREFCK rising	4	
EMAC <sub>26</sub>	Hold for ERXER from EREFCK rising	2	
EMAC <sub>27</sub>	Setup for ECRSDV from EREFCK rising	4	
EMAC <sub>28</sub>	Hold for ECRSDV from EREFCK rising	2	

**Figure 44-6.** EMAC RMII Timings



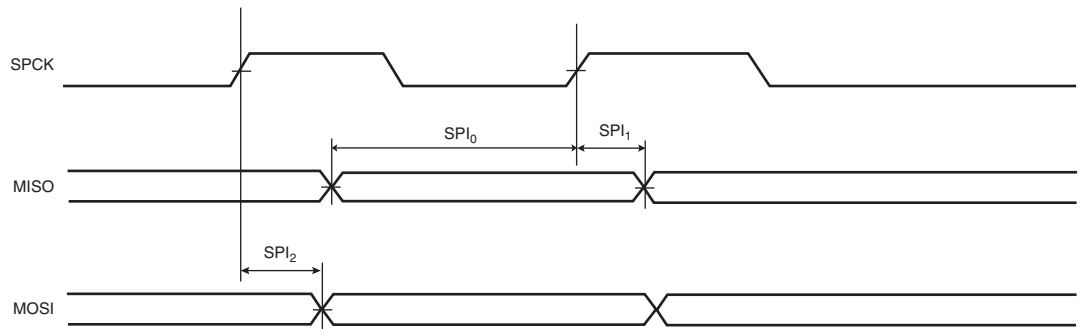
## 44.7.8 SPI

Timings are given assuming a capacitance load on MISO, SPCK and MOSI.

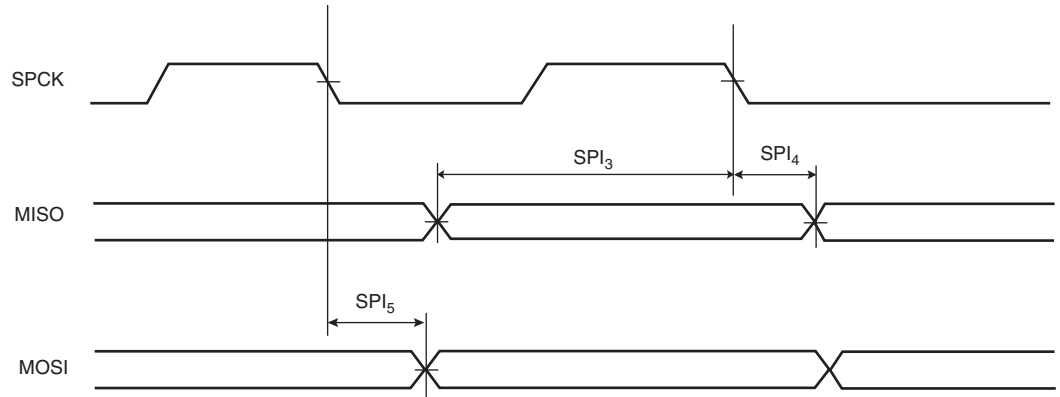
**Table 44-39.** Capacitance Load for MISO, SPCK and MOSI

IO Supply	Corner	
	MAX	STH
3.3V	40 pF	40 pF
1.8V	20 pF	20 pf

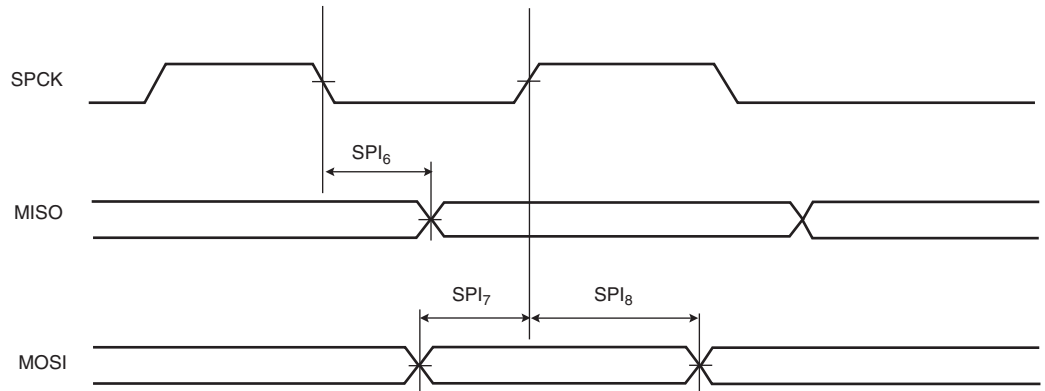
**Figure 44-7.** SPI Master Mode with (CPOL=0 and NCPHA=1) or (CPOL=1 and NCPHA=0)



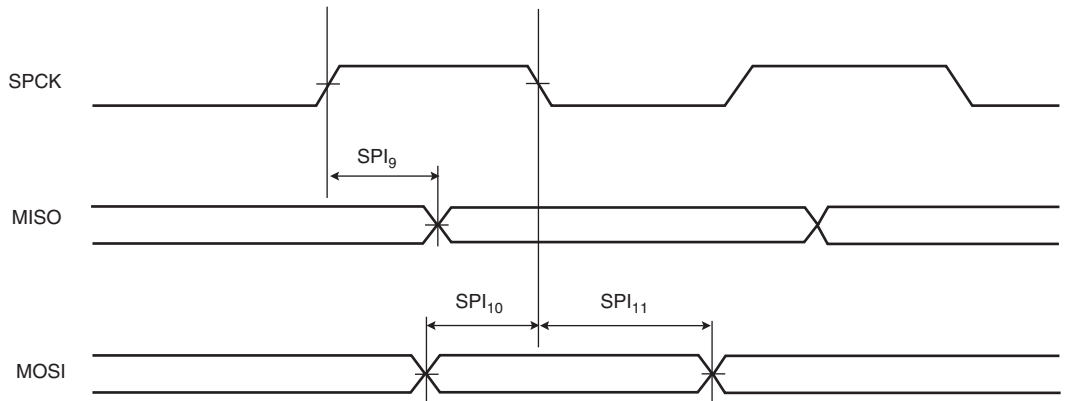
**Figure 44-8.** SPI Master Mode with (CPOL = NCPHA = 0) or (CPOL= NCPHA= 1)



**Figure 44-9.** SPI Slave Mode with (CPOL=0 and NCPHA=1) or (CPOL=1 and NCPHA=0)

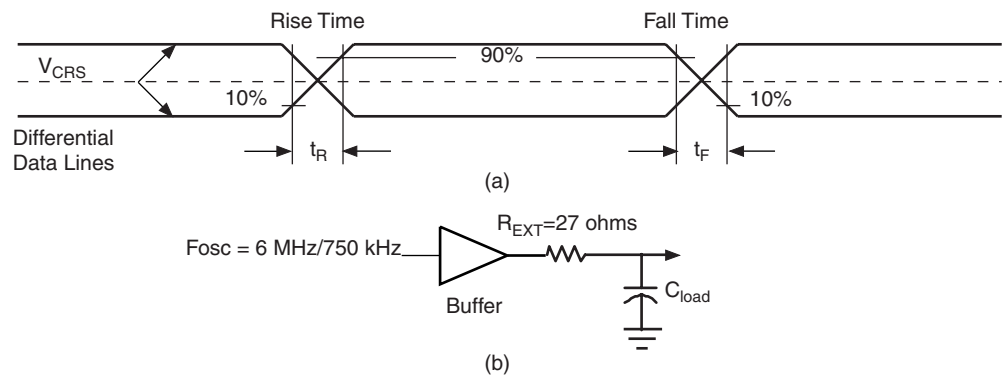


**Figure 44-10.** SPI Slave Mode with (CPOL = NCPHA = 0) or (CPOL= NCPHA= 1)





**Figure 44-11. SPI Slave Mode - NPCS Timings**



**Table 44-40. SPI Timings**

Symbol	Parameter	Cond	Min	Max	Units
Master Mode					
SPI <sub>0</sub>	MISO Setup time before SPCK rises	STH corner	$10.4 + 0.5 \cdot t_{CPMCK}$		ns
		MAX corner	$12.4 + 0.5 \cdot t_{CPMCK}$		ns
SPI <sub>1</sub>	MISO Hold time after SPCK rises	STH corner	$-8.6 - 0.5 \cdot t_{CPMCK}$		ns
		MAX corner	$-10.2 - 0.5 \cdot t_{CPMCK}$		ns
SPI <sub>2</sub>	SPCK rising to MOSI	STH corner	0.6		ns
		MAX corner	0.7		ns
SPI <sub>3</sub>	MISO Setup time before SPCK falls	STH corner	$11.1 + 0.5 \cdot t_{CPMCK}$		ns
		MAX corner	$13.0 + 0.5 \cdot t_{CPMCK}$		ns
SPI <sub>4</sub>	MISO Hold time after SPCK falls	STH corner	$-9 - 0.5 \cdot t_{CPMCK}$		ns
		MAX corner	$-10.6 - 0.5 \cdot t_{CPMCK}$		ns
SPI <sub>5</sub>	SPCK falling to MOSI	STH corner	0.1		ns
		MAX corner	0.3		ns
Slave Mode					
SPI <sub>6</sub>	SPCK falling to MISO	STH corner	11.5		ns
		MAX corner	13.5		ns
SPI <sub>7</sub>	MOSI Setup time before SPCK rises	STH corner	9.7		ns
		MAX corner	9.4		ns
SPI <sub>8</sub>	MOSI Hold time after SPCK rises	STH corner	-8.2		ns
		MAX corner	-7.9		ns
SPI <sub>9</sub>	SPCK rising to MISO	STH corner	9.2		ns
		MAX corner	10.8		ns
SPI <sub>10</sub>	MOSI Setup time before SPCK falls	STH corner	-10.3		ns
		MAX corner	-10.4		ns
SPI <sub>11</sub>	MOSI Hold time after SPCK falls	STH corner	11.8		ns
		MAX corner	12.0		ns

**Table 44-40. SPI Timings (Continued)**

Symbol	Parameter	Cond	Min	Max	Units
SPI <sub>12</sub>	NPCS0 setup to SPCK rising	STH corner	-7.0		ns
		MAX corner	-8.5		ns
SPI <sub>13</sub>	NPCS0 hold after SPCK falling	STH corner	10.2		ns
		MAX corner	12.5		ns
SPI <sub>14</sub>	NPCS0 setup to SPCK falling	STH corner	-9.6		ns
		MAX corner	-11.6		ns
SPI <sub>15</sub>	NPCS0 hold after SPCK rising	STH corner	10.1		ns
		MAX corner	12.5		ns
SPI <sub>16</sub>	NPCS0 falling to MISO valid	STH corner		9.2	ns
		MAX corner		10.7	ns

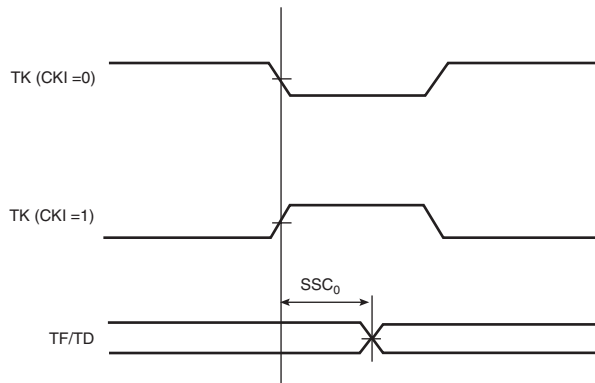
**44.7.9 SSC**

Timings are given assuming a capacitance load on TBC.

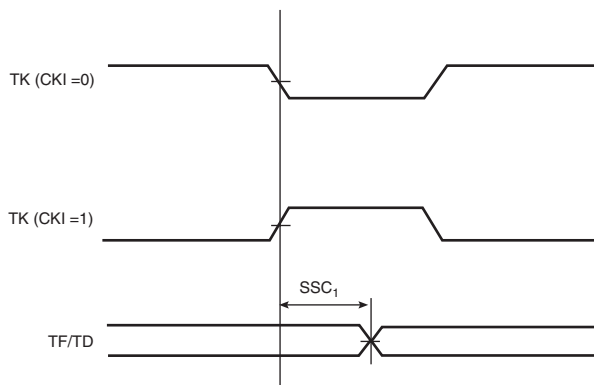
**Table 44-41. Capacitance Load**

IO Supply	Corner	
	MAX	STH
3.3V	30 pF	30 pF
1.8V	20 pF	20 pF

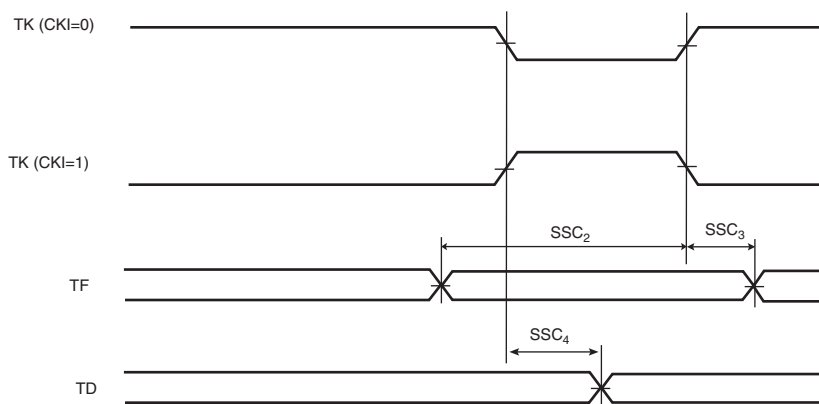
**Figure 44-12. SSC Transmitter, TK and TF in Output**



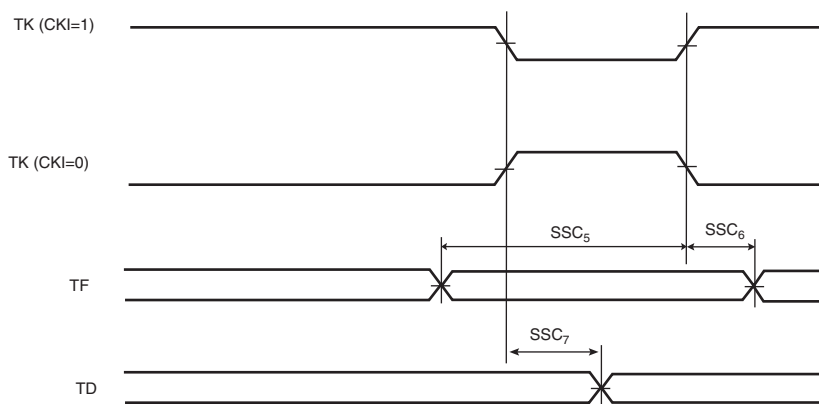
**Figure 44-13.** SSC Transmitter, TK in Input and TF in Output



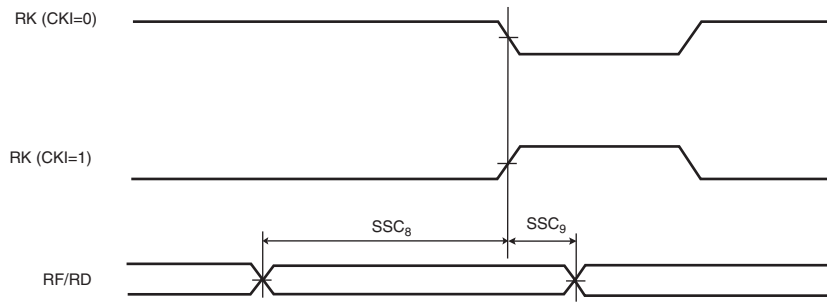
**Figure 44-14.** SSC Transmitter, TK in Output and TF in Input



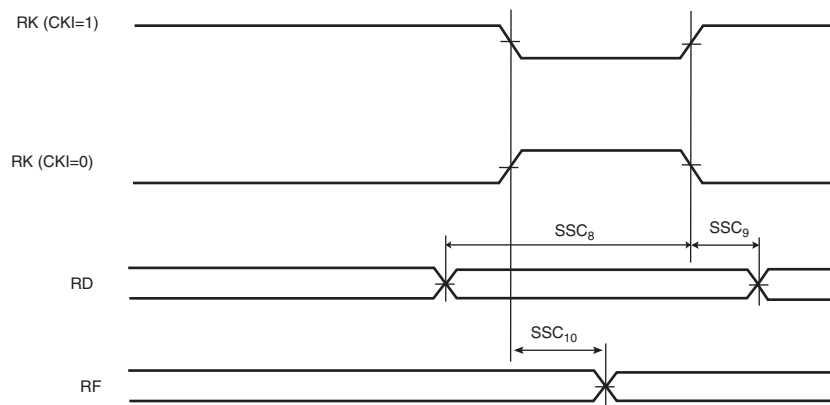
**Figure 44-15.** SSC Transmitter, TK and TF in Input



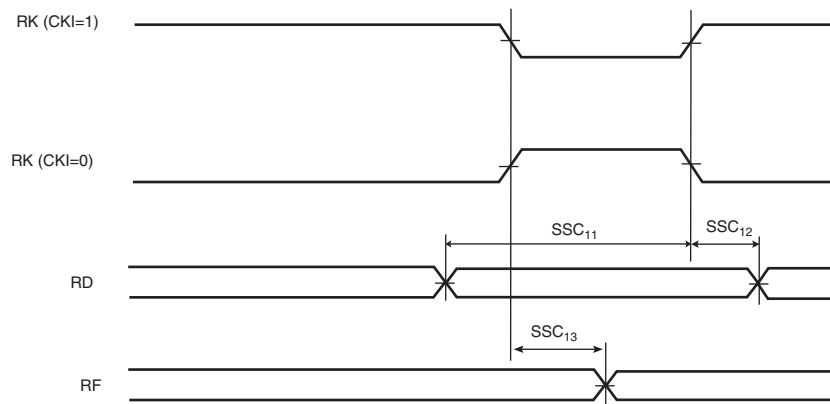
**Figure 44-16. SSC Receiver RK and RF in Input**



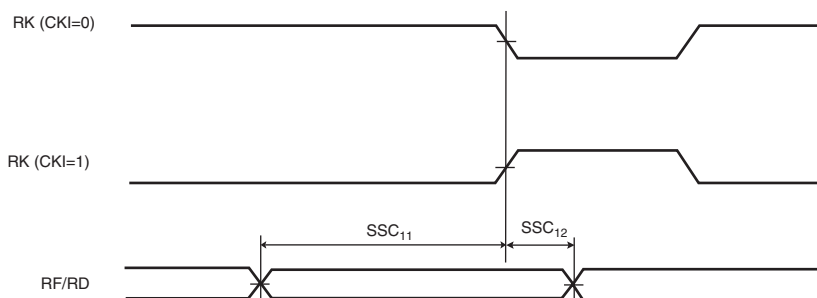
**Figure 44-17. SSC Receiver, RK in Input and RF in Output**



**Figure 44-18. SSC Receiver, RK and RF in Output**



**Figure 44-19.** SSC Receiver, RK in Output and RF in Input



**Table 44-42.** SSC Timings

Symbol	Parameter	Cond	Min	Max	Units
Transmitter					
SSC <sub>0</sub>	TK edge to TF/TD (TK output, TF output)	STH corner, VDDIO = 1.8V	-1.4 <sup>(2)</sup>	1.5 <sup>(2)</sup>	ns
		STH corner, VDDIO = 3.3V	-2.4 <sup>(2)</sup>	2.6 <sup>(2)</sup>	ns
		MAX corner, VDDIO = 1.8V	-1.3 <sup>(2)</sup>	1.5 <sup>(2)</sup>	ns
		MAX corner, VDDIO = 3.3V	-2.4 <sup>(2)</sup>	2.6 <sup>(2)</sup>	ns
SSC <sub>1</sub>	TK edge to TF/TD (TK input, TF output)	STH corner, VDDIO = 1.8V	-1.4 <sup>(2)</sup>	1.5 <sup>(2)</sup>	ns
		STH corner, VDDIO = 3.3V	-2.4 <sup>(2)</sup>	2.6 <sup>(2)</sup>	ns
		MAX corner, VDDIO = 1.8V	-1.3 <sup>(2)</sup>	1.5 <sup>(2)</sup>	ns
		MAX corner, VDDIO = 3.3V	-2.4 <sup>(2)</sup>	2.6 <sup>(2)</sup>	ns
SSC <sub>2</sub>	TF setup time before TK edge (TK output)	STH corner, VDDIO = 1.8V	10.0 - $t_{CPMCK}$		ns
		STH corner, VDDIO = 3.3V	11.0 - $t_{CPMCK}$		ns
		MAX corner, VDDIO = 1.8V	11.6 - $t_{CPMCK}$		ns
		MAX corner, VDDIO = 3.3V	12.8 - $t_{CPMCK}$		ns
SSC <sub>3</sub>	TF hold time after TK edge (TK output)	STH corner, VDDIO = 1.8V	$t_{CPMCK} - 8.7$		ns
		STH corner, VDDIO = 3.3V	$t_{CPMCK} - 8.7$		ns
		MAX corner, VDDIO = 1.8V	$t_{CPMCK} -$ 10.4		ns
		MAX corner, VDDIO = 3.3V	$t_{CPMCK} -$ 10.4		ns

**Table 44-42. SSC Timings (Continued)**

Symbol	Parameter	Cond	Min	Max	Units
SSC <sub>4</sub> <sup>(1)</sup>	TK edge to TF/TD (TK output, TF input)	STH corner, VDDIO = 1.8V	$2 * t_{CPMCK} - 0.7^{(1)(2)}$	$2 * t_{CPMCK} + 1.9^{(1)(2)}$	ns
		STH corner, VDDIO = 3.3V	$2 * t_{CPMCK} - 1.7^{(1)(2)}$	$2 * t_{CPMCK} + 2.9^{(1)(2)}$	ns
		MAX corner, VDDIO = 1.8V	$2 * t_{CPMCK} - 0.5^{(1)(2)}$	$2 * t_{CPMCK} + 1.9^{(1)(2)}$	ns
		MAX corner, VDDIO = 3.3V	$2 * t_{CPMCK} - 1.6^{(1)(2)}$	$2 * t_{CPMCK} + 3.1^{(1)(2)}$	ns
SSC <sub>5</sub>	TF setup time before TK edge (TK input)	STH corner, VDDIO = 1.8V	0		ns
		STH corner, VDDIO = 3.3V	0		ns
		MAX corner, VDDIO = 1.8V	0		ns
		MAX corner, VDDIO = 3.3V	0		ns
SSC <sub>6</sub>	TF hold time after TK edge (TK input)	STH corner, VDDIO = 1.8V	$t_{CPMCK}$		ns
		STH corner, VDDIO = 3.3V	$t_{CPMCK}$		ns
		MAX corner, VDDIO = 1.8V	$t_{CPMCK}$		ns
		MAX corner, VDDIO = 3.3V	$t_{CPMCK}$		ns
SSC <sub>7</sub> <sup>(1)</sup>	TK edge to TF/TD (TK input, TF input)	STH corner, VDDIO = 1.8V	10.8 <sup>(1)</sup>	$3 * t_{CPMCK} + 11^{(1)}$	ns
		STH corner, VDDIO = 3.3V	10.8 <sup>(1)</sup>	$3 * t_{CPMCK} + 12^{(1)}$	ns
		MAX corner, VDDIO = 1.8V	13.1 <sup>(1)</sup>	$3 * t_{CPMCK} + 12.8^{(1)}$	ns
		MAX corner, VDDIO = 3.3V	13.1 <sup>(1)</sup>	$3 * t_{CPMCK} + 13.9^{(1)}$	ns
Receiver					
SSC <sub>8</sub>	RF/RD setup time before RK edge (RK input)	STH corner, VDDIO = 1.8V	0		ns
		STH corner, VDDIO = 3.3V	0		ns
		MAX corner, VDDIO = 1.8V	0		ns
		MAX corner, VDDIO = 3.3V	0		ns
SSC <sub>9</sub>	RF/RD hold time after RK edge (RK input)	STH corner, VDDIO = 1.8V	$t_{CPMCK}$		ns
		STH corner, VDDIO = 3.3V	$t_{CPMCK}$		ns
		MAX corner, VDDIO = 1.8V	$t_{CPMCK}$		ns
		MAX corner, VDDIO = 3.3V	$t_{CPMCK}$		ns
SSC <sub>10</sub>	RK edge to RF (RK input)	STH corner, VDDIO = 1.8V	11.1 <sup>(2)</sup>	12.9 <sup>(2)</sup>	ns
		STH corner, VDDIO = 3.3V	11.1 <sup>(2)</sup>	13.9 <sup>(2)</sup>	ns
		MAX corner, VDDIO = 1.8V	13.3 <sup>(2)</sup>	15.2 <sup>(2)</sup>	ns
		MAX corner, VDDIO = 3.3V	13.3 <sup>(2)</sup>	16.3 <sup>(2)</sup>	ns

**Table 44-42. SSC Timings (Continued)**

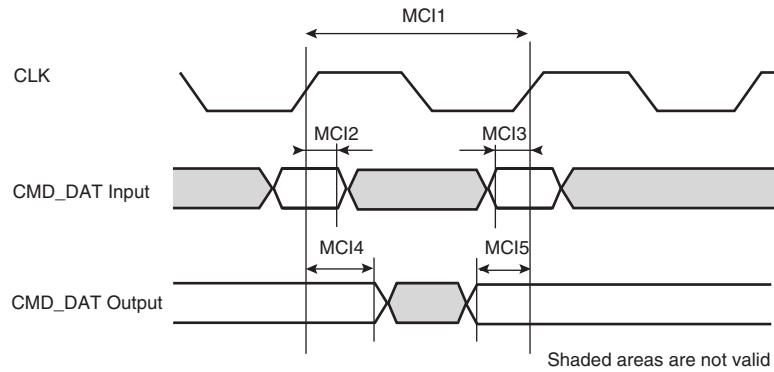
Symbol	Parameter	Cond	Min	Max	Units
SSC <sub>11</sub>	RF/RD setup time before RK edge (RK output)	STH corner, VDDIO = 1.8V	11.1 - $t_{CPMCK}$		ns
		STH corner, VDDIO = 3.3V	11.8 - $t_{CPMCK}$		ns
		MAX corner, VDDIO = 1.8V	12.6 - $t_{CPMCK}$		ns
		MAX corner, VDDIO = 3.3V	13.7 - $t_{CPMCK}$		ns
SSC <sub>12</sub>	RF/RD hold time after RK edge (RK output)	STH corner, VDDIO = 1.8V	$t_{CPMCK} - 8.8$		ns
		STH corner, VDDIO = 3.3V	$t_{CPMCK} - 8.8$		ns
		MAX corner, VDDIO = 1.8V	$t_{CPMCK} - 10.7$		ns
		MAX corner, VDDIO = 3.3V	$t_{CPMCK} - 10.7$		ns
SSC <sub>13</sub>	RK edge to RF (RK output)	STH corner, VDDIO = 1.8V	-1.0 <sup>(2)</sup>	1.8 <sup>(2)</sup>	ns
		STH corner, VDDIO = 3.3V	-2.1 <sup>(2)</sup>	2.8 <sup>(2)</sup>	ns
		MAX corner, VDDIO = 1.8V	-0.9 <sup>(2)</sup>	1.8 <sup>(2)</sup>	ns
		MAX corner, VDDIO = 3.3V	-2.0 <sup>(2)</sup>	2.9 <sup>(2)</sup>	ns

- Notes:
1. Timings SSC4 and SSC7 depend on the start condition. When STTDLY = 0 (Receive start delay) and START = 4, or 5 or 7 (Receive Start Selection), two Periods of the MCK must be added to timings.
  2. For output signals (TF, TD, RF), Min and Max access times are defined. The Min access time is the time between the TK (or RK) edge and the signal change. The Max access timing is the time between the TK edge and the signal stabilization. Figure 18 illustrates Min and Max accesses for SSC0. The same applies for SSC1, SSC4, and SSC7, SSC10 and SSC13.

### 44.7.10 MCI

Capacitance loads on data and clock are given in [Table 44-43](#).

**Figure 44-20.** MCI Timings



**Table 44-43.** MCI Timings

Symbol	Parameter	CLoad	Min	Max	Units
MCI <sub>1</sub>	CLK frequency at Data transfer Mode	C = 25 pF		25	MHz
		C = 100 pF		20	MHz
		C = 250 pF		20	MHz
	CLK frequency at Identification Mode			400	kHz
	CLK Low time	C = 100 pF	10		ns
	CLK High time	C = 100 pF	10		ns
	CLK Rise time	C = 100 pF		10	ns
	CLK Fall time	C = 100 pF		10	ns
	CLK Low time	C = 250 pF	50		ns
	CLK High time	C = 250 pF	50		ns
	CLK Rise time	C = 250 pF		50	ns
	CLK Fall time	C = 250 pF		50	ns
MCI <sub>2</sub>	Input hold time		3		ns
MCI <sub>3</sub>	Input setup time		3		ns
MCI <sub>4</sub>	Output change after CLK rising		5		ns
MCI <sub>5</sub>	Output valid before CLK rising		5		ns



## 44.7.11 UDP HS Transceiver

Figure 44-21. USB Control/Data Signals

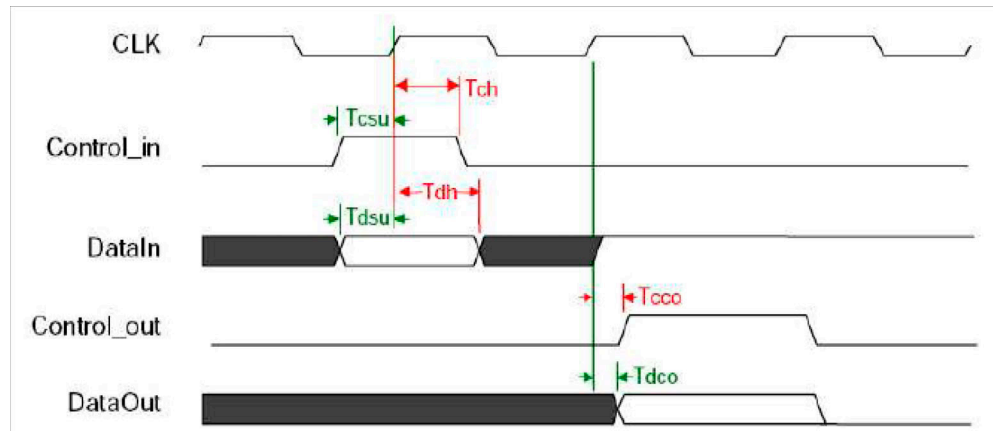


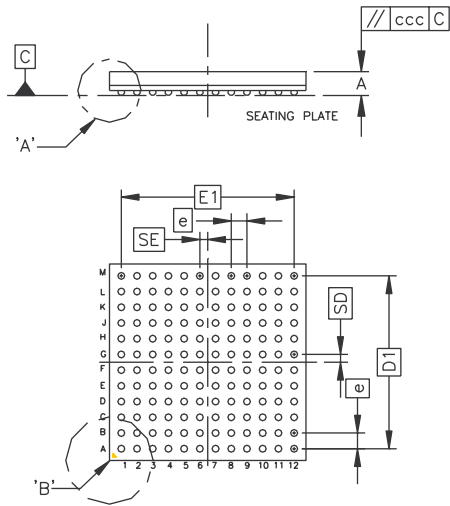
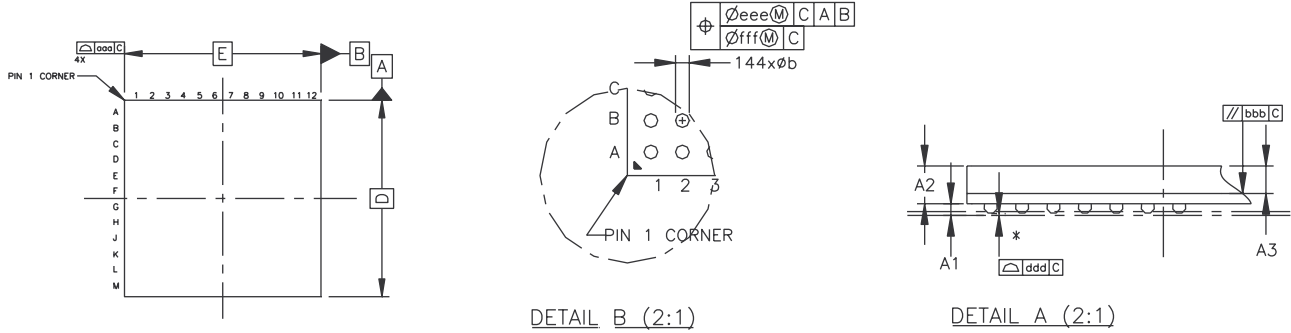
Table 44-44. UUSB Control/Data Timings

Symbol	Parameter	Min	Typ	Max	Unit
$T_{csu / dsu}$	Control/Data Signal Setup Time	7			ns
$T_{ch / dh}$	Control/Data Signal Hold Time	-3			ns
$T_{cco / dco}$	Control/Data Signal Clock to Out Time	4		8	ns

## 45. AT91SAM9R64/RL64 Mechanical Characteristics

### 45.1 Package Drawings

Figure 45-1. 144-ball BGA Package Drawing



ALL DIMENSIONS ARE IN MILLIMETERS.

SYMBOL	MILLIMETER			INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	---	---	1.40	---	---	0.0551
A1	0.20	0.25	0.30	0.0079	0.0098	0.0118
A2	0.92	0.96	1.00	0.0362	0.0378	0.0394
A3	0.70 BASIC			0.0276 BASIC		
D	9.95	10.00	10.05	0.3917	0.3937	0.3957
D1	8.80 BASIC			0.3465 BASIC		
E	9.95	10.00	10.05	0.3917	0.3937	0.3957
E1	8.80 BASIC			0.3465 BASIC		
SD	0.40 BASIC			0.0157 BASIC		
SE	0.40 BASIC			0.0157 BASIC		
e	0.80 BASIC			0.0315 BASIC		
b	0.30	0.35	0.40	0.0118	0.0138	0.0157
aaa	0.15			0.0059		
bbb	0.20			0.0079		
ccc	0.20			0.0079		
ddd	0.08			0.0031		
eee	0.15			0.0059		
fff	0.08			0.0031		

Table 45-1. Soldering Information

Ball Land	0.380 mm
Soldering Mask Opening	0.280 mm

Table 45-2. Device and 144-ball BGA Package Maximum Weight

300	mg
-----	----

Table 45-3. 144-ball BGA Package Characteristics

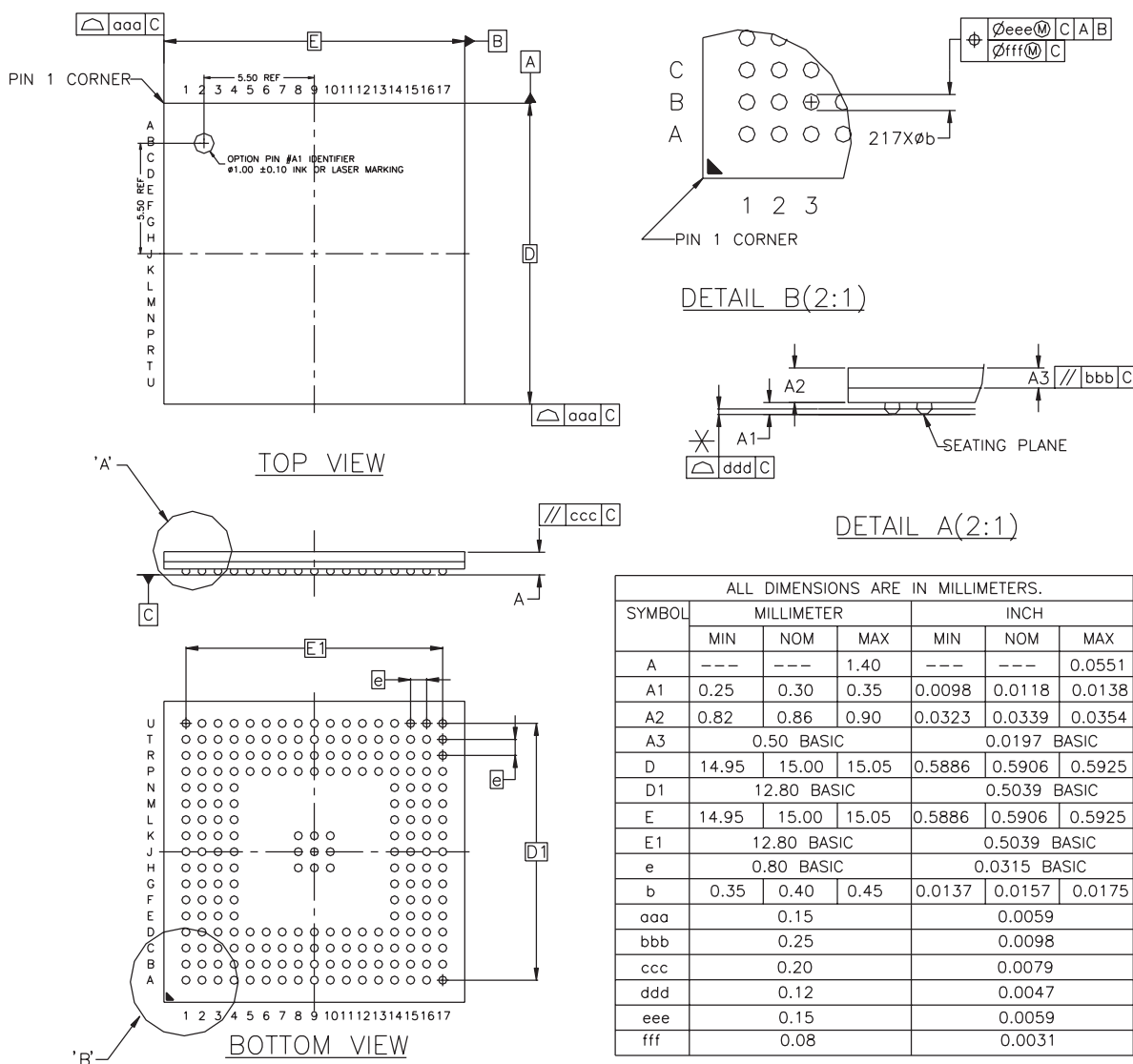
Moisture Sensitivity Level	3
----------------------------	---

Table 45-4. Package Reference

JEDEC Drawing Reference	none
JESD97 Classification	e1

This package respects the recommendations of the NEMI User Group.

**Figure 45-2.** 217-ball LFBGA Package Drawing



ALL DIMENSIONS ARE IN MILLIMETERS.

SYMBOL	MILLIMETER			INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	---	---	1.40	---	---	0.0551
A1	0.25	0.30	0.35	0.0098	0.0118	0.0138
A2	0.82	0.86	0.90	0.0323	0.0339	0.0354
A3	0.50 BASIC			0.0197 BASIC		
D	14.95	15.00	15.05	0.5886	0.5906	0.5925
D1	12.80 BASIC			0.5039 BASIC		
E	14.95	15.00	15.05	0.5886	0.5906	0.5925
E1	12.80 BASIC			0.5039 BASIC		
e	0.80 BASIC			0.0315 BASIC		
b	0.35	0.40	0.45	0.0137	0.0157	0.0175
aaa	0.15			0.0059		
bbb	0.25			0.0098		
ccc	0.20			0.0079		
ddd	0.12			0.0047		
eee	0.15			0.0059		
fff	0.08			0.0031		

**Table 45-5.** Soldering Information

Ball Land	0.43 mm ± 0.05
Solder Mask Opening	0.30 mm ± 0.05

**Table 45-6.** Device and 217-ball LFBGA Package Maximum Weight

450	mg
-----	----

**Table 45-7.** 217-ball LFBGA Package Characteristics

Moisture Sensitivity Level	3
----------------------------	---

**Table 45-8.** Package Reference

JEDEC Drawing Reference	MO-205
JESD97 Classification	e1

## 45.2 Soldering Profile

Table 45-9 gives the recommended soldering profile from J-STD-020C.

**Table 45-9.** Soldering Profile

Profile Feature	Green Package
Average Ramp-up Rate (217°C to Peak)	3°C/sec. max.
Preheat Temperature 175°C ±25°C	180 sec. max.
Temperature Maintained Above 217°C	60 sec. to 150 sec.
Time within 5°C of Actual Peak Temperature	20 sec. to 40 sec.
Peak Temperature Range	260 +0 °C
Ramp-down Rate	6°C/sec. max.
Time 25°C to Peak Temperature	8 min. max.

Note: It is recommended to apply a soldering temperature higher than 250°C

A maximum of three reflow passes is allowed per component.

## 46. AT91SAM9R64/RL64 Ordering Information

Table 46-1. AT91SAM9R64/RL64 Ordering Information

Ordering Code	Package	Package Type	Temperature Operating Range
AT91SAM9R64-CU	LFBGA144	Green	Industrial -40°C to 85°C
AT91SAM9RL64-CU	LFBGA217	Green	

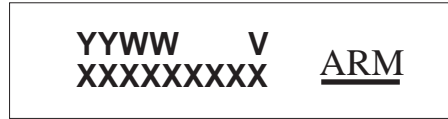


## 47. AT91SAM9R64/RL64 Errata

### 47.1 Marking

All devices are marked with the Atmel logo and the ordering code.

Additional marking has the following format:



where:

- “YY”: manufactory year
- “WW”: manufactory week
- “V”: revision
- “XXXXXXXXX”: lot number

## 47.2 AT91SAM9R64/RL64 Errata - Revision A Parts

Refer to [Section 47.1 “Marking” on page 895](#).

### 47.2.1 Bus Matrix

#### 47.2.1.1 *DEFAULT\_MASTER* functionality

The DEFAULT\_MASTER functionality is not available for the following slaves:

- Slave 2: USB Device High-Speed
- Slave 3: LCDC User Interface

#### **Problem Fix/ Workaround**

None.

#### 47.2.1.2 *FIXED\_PRIORITY* functionality

The FIXED\_PRIORITY arbitration scheme is not available for the following slaves:

- Slave 2: USB Device High-Speed
- Slave 3: LCDC User Interface
- Slave 5: Peripheral bridge

#### **Problem Fix/ Workaround**

None.

### 47.2.2 DMA Controller

#### 47.2.2.1 *Transfer size is limited in some conditions*

DMA Controller may be frozen if BTSIZE field is too big.

- BTSIZE is limited to 16383 (0x3FFF) when SRC\_WIDTH = WORD and DST\_WIDTH = BYTE
- BTSIZE is limited to 32767 (0x7FFF) when SRC\_WIDTH = HALFWORD and DST\_WIDTH = BYTE
- BTSIZE is limited to 32767 (0x7FFF) when SRC\_WIDTH = WORD and DST\_WIDTH = HALF\_WORD

#### **Problem Fix/Workaround**

Choose another transfer configuration.

### 47.2.3 MCI

#### 47.2.3.1 *Busy signal of R1b responses is not taken in account*

The busy status of the card during the response (R1b) is ignored for the commands CMD7, CMD28, CMD29, CMD38, CMD42, CMD56. Additionally, for commands CMD42 and CMD56 a conflict can occur on data line0 if the MCI sends data to the card while the card is still busy. The behavior is correct for CMD12 command (STOP\_TRANSFER).

#### **Problem Fix/Workaround**

None

#### 47.2.3.2 *SDIO interrupt does not work for slot different from A*

If 1-bit data bus width and on other slots than slot A, the SDIO interrupt can not be captured. The sample is made on the bad data line.



## Problem Fix/Workaround

None

### 47.2.3.3 Data Timeout Error Flag

As the data Timeout error flag checking the Naac timing cannot rise, the MCI can be stalled waiting indefinitely the Data start bit.

## Problem Fix/Workaround

A STOP command must be sent with a software timeout.

### 47.2.3.4 Data Write Operation and number of bytes

The Data Write operation with a number of bytes less than 12 is impossible.

## Problem Fix/Workaround

The PDC counters must always be equal to 12 bytes for data transfers lower than 12 bytes. The BLKLEN or BCNT field are used to specify the real count number.

### 47.2.3.5 Flag Reset is not correct in half duplex mode

In half duplex mode, the reset of the flags ENDRX, RXBUFF, ENDTX and TXBUFE can be incorrect. These flags are reset correctly after a PDC channel enable.

## Problem Fix/Workaround

Enable the interrupts related to ENDRX, ENDTX, RXBUFF and TXBUFE only after enabling the PDC channel by writing PDC\_TXTEN or PDC\_RXTEN.

## 47.2.4 Serial Synchronous Controller (SSC)

### 47.2.4.1 Transmitter Limitations in Slave Mode

If TK is programmed as output and TF is programmed as input, it is impossible to emit data when start of edge (rising or falling) of synchro with a Start Delay equal to zero.

## Problem Fix/Workaround

None.

### 47.2.4.2 Periodic Transmission Limitations in Master Mode

If Last Significant Bit is sent first (MSBF = 0) the first TAG during the frame synchro is not sent.

## Problem Fix/Workaround

None.

### 47.2.4.3 Unexpected RK clock cycle when RK outputs a clock during data transfer

When the SSC receiver is used in the following configuration:

- the internal clock divider is used (CKS =0 and DIV different from 0),
- RK pin set as output and provides the clock during data transfer (CKO=2)
- data sampled on RK falling edge (CKI =0)

then, at the end of the data, the RK pin is set in high impedance which may be interpreted as an unexpected clock cycle.

## Problem Fix/Workaround

Enable the pull-up on RK pin.

#### 47.2.4.4 *Incorrect first RK clock cycle when RK outputs a clock during data transfer*

When the SSC receiver is used in the following configuration:

- RX clock is divided clock (CKS =0 and DIV different from 0)
- RK pin set as output and provides the clock during data transfer (CKO=2)
- data sampled on RK falling edge (CKI =0)

then the first clock cycle time generated by the RK pin is equal to  $MCK/(2 \times (DIV + 1))$  instead of  $MCK/(2 \times DIV)$ .

##### **Problem Fix/Workaround**

None.

### 47.2.5 Serial Peripheral Interface (SPI)

#### 47.2.5.1 *Bad Serial Clock Generation on second chip\_select when SCBR = 1, CPOL = 1 and NCPHA = 0*

If the SPI is used in the following configuration:

- master mode
- CPOL=1 and NCPHA =0
- multiple chip selects used with one transfer with Baud rate (SCBR) equal to 1 (i.e., when serial clock frequency equals the system clock frequency) and the other transfers set with SCBR not equal to 1
- transmit with the slowest chip select and then with the fastest one

then an additional pulse will be generated on output PSCK during the second transfer.

##### **Problem Fix/Workaround**

Do not use a multiple Chip Select configuration where at least one SCR<sub>x</sub> register is configured with SCBR = 1 and the others differ from 1 if CPHA = 0 and CPOL = 1.

If all chip selects are configured with Baudrate = 1, the issue does not appear.

### 47.2.6 System Controller

#### 47.2.6.1 *Possible event loss when reading RTT\_SR*

If an event (RTTINC or ALMS) occurs within the same slow clock cycle the RTT\_SR is read, the corresponding bit might be cleared. This might lead in the loss of this event.

##### **Problem Fix/Workaround**

The software must handle RTT event as interrupt and should not poll RTT\_SR.

### 47.2.7 USART

#### 47.2.7.1 *RXBREAK problem when no timeguard*

The RXBREAK flag is not correctly handled (FRAME ERROR is set instead) when the time-guard is 0 and the break character is located just after STOP BIT.

##### **Problem Fix/Workaround**

If the NBSTOP = 1, => TIMEGUARD should be different from 0.

SYNCHRONOUS mode is not affected, only ASYNCHRONOUS.

## 47.2.7.2 *DCD is active High instead of Low*

DCD signal is active at high level in the USART block (Modem Mode).

DCD should be active at low level.

### **Problem Fix/Workaround**

Add an inverter.



## Table of Contents

	<b>Features .....</b>	<b>1</b>
<b>1</b>	<b>Description .....</b>	<b>3</b>
<b>2</b>	<b>Block Diagrams .....</b>	<b>5</b>
<b>3</b>	<b>Signal Description .....</b>	<b>7</b>
<b>4</b>	<b>Package and Pinout .....</b>	<b>12</b>
	4.1144-ball BGA Package Outline .....	12
	4.2Pinout .....	13
	4.3217-ball LFBGA Package Outline .....	14
	4.4Pinout .....	15
<b>5</b>	<b>Power Considerations .....</b>	<b>16</b>
	5.1Power Supplies .....	16
	5.2Power Consumption .....	16
	5.3Programmable I/O Lines Power Supplies .....	16
<b>6</b>	<b>I/O Line Considerations .....</b>	<b>17</b>
	6.1JTAG Port Pins .....	17
	6.2Test Pin .....	17
	6.3Reset Pins .....	17
	6.4PIO Controllers .....	17
	6.5Shutdown Logic Pins .....	18
<b>7</b>	<b>Processor and Architecture .....</b>	<b>18</b>
	7.1ARM926EJ-S Processor .....	18
	7.2Matrix Masters .....	19
	7.3Matrix Slaves .....	19
	7.4Master to Slave Access .....	19
	7.5Peripheral DMA Controller (PDC) .....	20
	7.6DMA Controller .....	20
	7.7Debug and Test Features .....	20
<b>8</b>	<b>Memories .....</b>	<b>22</b>
	8.1Embedded Memories .....	23
	8.2External Memories .....	26
<b>9</b>	<b>System Controller .....</b>	<b>28</b>

9.1	System Controller Mapping .....	28
9.2	Block Diagram .....	29
9.3	Reset Controller .....	30
9.4	Shutdown Controller .....	30
9.5	Clock Generator .....	30
9.6	Slow Clock Selection .....	31
9.7	Power Management Controller .....	31
9.8	Periodic Interval Timer .....	32
9.9	Watchdog Timer .....	32
9.10	Real-Time Timer .....	32
9.11	Real-Time Clock .....	32
9.12	General-Purpose Backed-up Registers .....	33
9.13	Advanced Interrupt Controller .....	33
9.14	Debug Unit .....	33
9.15	Chip Identification .....	34
9.16	PIO Controllers .....	34
<b>10</b>	<b><i>Peripherals .....</i></b>	<b>34</b>
10.1	Peripheral Mapping .....	34
10.2	Peripheral Identifiers .....	35
10.3	Peripheral Interrupts and Clock Control .....	36
10.4	Peripherals Signals Multiplexing on I/O Lines .....	36
<b>11</b>	<b><i>Embedded Peripherals Overview .....</i></b>	<b>44</b>
11.1	Serial Peripheral Interface (SPI) .....	44
11.2	Two-wire Interface (TWI) .....	44
11.3	USART .....	44
11.4	Serial Synchronous Controller (SSC) .....	45
11.5	AC97 Controller .....	45
11.6	Timer Counter (TC) .....	45
11.7	Pulse Width Modulation Controller (PWM) .....	46
11.8	Multimedia Card Interface (MCI) .....	46
11.9	USB High Speed Device Port (UDPHS) .....	46
11.10	LCD Controller (LCDC) .....	47
11.11	Touch Screen Analog-to-digital Converter (TSADCC) .....	47
<b>12</b>	<b><i>ARM926EJ-S Processor Overview .....</i></b>	<b>49</b>
12.1	Overview .....	49

12.2	Block Diagram	50
12.3	ARM9EJ-S Processor	50
12.4	CP15 Coprocessor	58
12.5	Memory Management Unit (MMU)	61
12.6	Caches and Write Buffer	62
12.7	Tightly-Coupled Memory Interface	64
12.8	Bus Interface Unit	65
<b>13</b>	<b>AT91SAM9R64/RL64 Debug and Test</b>	<b>67</b>
13.1	Description	67
13.2	Block Diagram	68
13.3	Application Examples	69
13.4	Debug and Test Pin Description	70
13.5	Functional Description	70
<b>14</b>	<b>AT91SAM9R64/RL64 Boot Program</b>	<b>73</b>
14.1	Description	73
14.2	Flow Diagram	73
14.3	Device Initialization	74
14.4	DataFlash Boot	75
14.5	SD Card Boot	78
14.6	NAND Flash Boot	78
14.7	SAM-BA Boot	79
14.8	Hardware and Software Constraints	82
<b>15</b>	<b>Reset Controller (RSTC)</b>	<b>85</b>
15.1	Description	85
15.2	Block Diagram	85
15.3	Functional Description	85
15.4	Reset Controller (RSTC) User Interface	94
<b>16</b>	<b>Real-time Timer (RTT)</b>	<b>99</b>
16.1	Overview	99
16.2	Block Diagram	99
16.3	Functional Description	99
16.4	Real-time Timer (RTT) User Interface	101
<b>17</b>	<b>Periodic Interval Timer (PIT)</b>	<b>105</b>
17.1	Overview	105

17.2	Block Diagram .....	105
17.3	Functional Description .....	106
17.4	Periodic Interval Timer (PIT) User Interface .....	108
<b>18</b>	<b><i>Watchdog Timer (WDT)</i></b> .....	<b>111</b>
18.1	Description .....	111
18.2	Block Diagram .....	111
18.3	Functional Description .....	112
18.4	Watchdog Timer (WDT) User Interface .....	114
<b>19</b>	<b><i>Shutdown Controller (SHDWC)</i></b> .....	<b>119</b>
19.1	Description .....	119
19.2	Block Diagram .....	119
19.3	I/O Lines Description .....	119
19.4	Product Dependencies .....	119
19.5	Functional Description .....	120
19.6	Shutdown Controller (SHDWC) User Interface .....	121
<b>20</b>	<b><i>Real-time Clock (RTC)</i></b> .....	<b>125</b>
20.1	Description .....	125
20.2	Block Diagram .....	125
20.3	Product Dependencies .....	125
20.4	Functional Description .....	125
20.5	Real-time Clock (RTC) User Interface .....	127
<b>21</b>	<b><i>AT91SAM9R64/RL64 Bus Matrix</i></b> .....	<b>141</b>
21.1	Description .....	141
21.2	Memory Mapping .....	141
21.3	Special Bus Granting Techniques .....	141
21.4	Arbitration .....	142
21.5	Bus Matrix User Interface .....	145
<b>22</b>	<b><i>External Bus Interface (EBI)</i></b> .....	<b>153</b>
22.1	Description .....	153
22.2	Block Diagram .....	154
22.3	I/O Lines Description .....	155
22.4	Application Example .....	157
22.5	Product Dependencies .....	160
22.6	Functional Description .....	160



22.7	Implementation Examples .....	168
<b>23</b>	<b><i>Static Memory Controller (SMC)</i></b> .....	<b>177</b>
23.1	Description .....	177
23.2	I/O Lines Description .....	177
23.3	Multiplexed Signals .....	177
23.4	Application Example .....	178
23.5	Product Dependencies .....	178
23.6	External Memory Mapping .....	179
23.7	Connection to External Devices .....	179
23.8	Standard Read and Write Protocols .....	183
23.9	Automatic Wait States .....	192
23.10	Data Float Wait States .....	197
23.11	External Wait .....	201
23.12	Slow Clock Mode .....	207
23.13	Asynchronous Page Mode .....	210
23.14	Static Memory Controller (SMC) User Interface .....	213
<b>24</b>	<b><i>SDRAM Controller (SDRAMC)</i></b> .....	<b>219</b>
24.1	Description .....	219
24.2	I/O Lines Description .....	219
24.3	Application Example .....	220
24.4	Product Dependencies .....	222
24.5	Functional Description .....	224
24.6	SDRAM Controller User Interface .....	232
<b>25</b>	<b><i>Error Corrected Code (ECC) Controller</i></b> .....	<b>243</b>
25.1	Description .....	243
25.2	Block Diagram .....	243
25.3	Functional Description .....	243
25.4	Error Corrected Code (ECC) Controller User Interface .....	248
<b>26</b>	<b><i>Peripheral DMA Controller (PDC)</i></b> .....	<b>253</b>
26.1	Description .....	253
26.2	Block Diagram .....	254
26.3	Functional Description .....	255
26.4	Peripheral DMA Controller (PDC) User Interface .....	258
<b>27</b>	<b><i>Clock Generator</i></b> .....	<b>269</b>

27.1	Description .....	269
27.2	Slow Clock Crystal Oscillator .....	269
27.3	Slow Clock RC Oscillator .....	269
27.4	Slow Clock Selection .....	269
27.5	Main Oscillator .....	272
27.6	Divider and PLL Block .....	273
<b>28</b>	<b><i>Power Management Controller (PMC)</i></b> .....	<b>276</b>
28.1	Description .....	276
28.2	Master Clock Controller .....	276
28.3	Processor Clock Controller .....	277
28.4	Peripheral Clock Controller .....	277
28.5	Programmable Clock Output Controller .....	277
28.6	Programming Sequence .....	278
28.7	Clock Switching Details .....	281
28.8	Power Management Controller (PMC) User Interface .....	285
<b>29</b>	<b><i>Advanced Interrupt Controller (AIC)</i></b> .....	<b>301</b>
29.1	Description .....	301
29.2	Block Diagram .....	302
29.3	Application Block Diagram .....	302
29.4	AIC Detailed Block Diagram .....	302
29.5	I/O Line Description .....	303
29.6	Product Dependencies .....	303
29.7	Functional Description .....	304
29.8	Advanced Interrupt Controller (AIC) User Interface .....	314
<b>30</b>	<b><i>Debug Unit (DBGU)</i></b> .....	<b>325</b>
30.1	Description .....	325
30.2	Block Diagram .....	326
30.3	Product Dependencies .....	327
30.4	UART Operations .....	327
30.5	Debug Unit User Interface .....	334
<b>31</b>	<b><i>Parallel Input/Output Controller (PIO)</i></b> .....	<b>349</b>
31.1	Description .....	349
31.2	Block Diagram .....	350
31.3	Product Dependencies .....	351
31.4	Functional Description .....	352

31.5	I/O Lines Programming Example .....	356
31.6	Parallel Input/Output (PIO) Controller User Interface .....	358
<b>32</b>	<b><i>Serial Peripheral Interface (SPI)</i></b> .....	<b>375</b>
32.1	Description .....	375
32.2	Block Diagram .....	376
32.3	Application Block Diagram .....	376
32.4	Signal Description .....	377
32.5	Product Dependencies .....	377
32.6	Functional Description .....	378
32.7	Serial Peripheral Interface (SPI) User Interface .....	387
<b>33</b>	<b><i>Two-wire Interface (TWI)</i></b> .....	<b>401</b>
33.1	Description .....	401
33.2	List of Abbreviations .....	402
33.3	Block Diagram .....	402
33.4	Application Block Diagram .....	403
33.5	Product Dependencies .....	403
33.6	Functional Description .....	404
33.7	Master Mode .....	405
33.8	Multi-master Mode .....	417
33.9	Slave Mode .....	420
33.10	Two-wire Interface (TWI) User Interface .....	428
<b>34</b>	<b><i>Universal Synchronous/Asynchronous Receiver/Transceiver</i></b> .....	<b>443</b>
34.1	Description .....	443
34.2	Block Diagram .....	444
34.3	Application Block Diagram .....	445
34.4	I/O Lines Description .....	445
34.5	Product Dependencies .....	446
34.6	Functional Description .....	447
34.7	USART User Interface .....	478
<b>35</b>	<b><i>Serial Synchronous Controller (SSC)</i></b> .....	<b>503</b>
35.1	Description .....	503
35.2	Block Diagram .....	504
35.3	Application Block Diagram .....	504
35.4	Pin Name List .....	505
35.5	Product Dependencies .....	505

35.6	Functional Description .....	505
35.7	SSC Application Examples .....	516
35.8	Synchronous Serial Controller (SSC) User Interface .....	518
<b>36</b>	<b><i>Timer Counter (TC)</i></b> .....	<b>541</b>
36.1	Description .....	541
36.2	Block Diagram .....	542
36.3	Pin Name List .....	543
36.4	Product Dependencies .....	543
36.5	Functional Description .....	544
36.6	Timer Counter (TC) User Interface .....	557
<b>37</b>	<b><i>DMA Controller (DMAC)</i></b> .....	<b>575</b>
37.1	Description .....	575
37.2	Block Diagram .....	576
37.3	Functional Description .....	577
37.4	DMAC Software Requirements .....	600
37.5	DMA Controller (DMAC) User Interface .....	602
<b>38</b>	<b><i>MultiMedia Card Interface (MCI)</i></b> .....	<b>623</b>
38.1	Description .....	623
38.2	Block Diagram .....	624
38.3	Application Block Diagram .....	625
38.4	Pin Name List .....	625
38.5	Product Dependencies .....	625
38.6	Bus Topology .....	626
38.7	MultiMedia Card Operations .....	629
38.8	SD/SDIO Card Operations .....	638
38.9	MultiMedia Card Interface (MCI) User Interface .....	639
<b>39</b>	<b><i>LCD Controller (LCDC)</i></b> .....	<b>663</b>
39.1	Description .....	663
39.2	Block Diagram .....	664
39.3	I/O Lines Description .....	665
39.4	Product Dependencies .....	665
39.5	Functional Description .....	665
39.6	Interrupts .....	686
39.7	Configuration Sequence .....	686
39.8	Double-buffer Technique .....	687

39.9	2D Memory Addressing .....	688
39.10	Register Configuration Guide .....	690
39.11	LCD Controller (LCDC) User Interface .....	691
<b>40</b>	<b><i>AC'97 Controller (AC'97C) .....</i></b>	<b>719</b>
40.1	Description .....	719
40.2	Block Diagram .....	720
40.3	Pin Name List .....	721
40.4	Application Block Diagram .....	721
40.5	Product Dependencies .....	722
40.6	Functional Description .....	723
40.7	AC'97 Controller (AC97C) User Interface .....	734
<b>41</b>	<b><i>USB High Speed Device Port (UDPHS) .....</i></b>	<b>749</b>
41.1	Description .....	749
41.2	Block Diagram .....	750
41.3	Typical Connection .....	751
41.4	Functional Description .....	751
41.5	USB High Speed Device Port (UDPHS) User Interface .....	776
<b>42</b>	<b><i>Pulse Width Modulation (PWM) Controller .....</i></b>	<b>813</b>
42.1	Description .....	813
42.2	Block Diagram .....	813
42.3	I/O Lines Description .....	814
42.4	Product Dependencies .....	814
42.5	Functional Description .....	814
42.6	Pulse Width Modulation (PWM) Controller User Interface .....	822
<b>43</b>	<b><i>Touch Screen ADC Controller .....</i></b>	<b>833</b>
43.1	Description .....	833
43.2	Block Diagram .....	834
43.3	Signal Description .....	835
43.4	Product Dependencies .....	835
43.5	Analog-to-digital Converter Functional Description .....	836
43.6	Touch Screen .....	837
43.7	Conversion Results .....	840
43.8	Conversion Triggers .....	843
43.9	Operating Modes .....	843
43.10	Touch Screen ADC Controller (TSADCC) User Interface .....	846





<b>44</b>	<b><i>AT91SAM9R64/RL64 Electrical Characteristics</i></b> .....	<b>861</b>
	44.1 Absolute Maximum Ratings .....	861
	44.2 DC Characteristics .....	861
	44.3 Power Consumption .....	862
	44.4 Crystal Oscillator Characteristics .....	864
	44.5 USB HS Characteristics .....	867
	44.6 Core Power Supply POR Characteristics .....	870
	44.7 Timings .....	871
<b>45</b>	<b><i>AT91SAM9R64/RL64 Mechanical Characteristics</i></b> .....	<b>890</b>
	45.1 Package Drawings .....	890
	45.2 Soldering Profile .....	892
<b>46</b>	<b><i>AT91SAM9R64/RL64 Ordering Information</i></b> .....	<b>893</b>
<b>47</b>	<b><i>AT91SAM9R64/RL64 Errata</i></b> .....	<b>895</b>
	47.1 Marking .....	895
	47.2 AT91SAM9R64/RL64 Errata - Revision A Parts .....	896
	<b><i>Table of Contents</i></b> .....	<b><i>i</i></b>



## 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](http://www.atmel.com)  
[www.atmel.com/AT91SAM](http://www.atmel.com/AT91SAM)

**Technical Support**  
AT91SAM Support

**Sales Contacts**  
[www.atmel.com/contacts/](http://www.atmel.com/contacts/)

**Literature Requests**  
[www.atmel.com/literature](http://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.



© 2008 Atmel Corporation. All rights reserved. Atmel®, logo and combinations thereof, DataFlash®, SAM-BA® and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. ARM®, the ARMPowered® logo, Thumb® and others are the registered trademarks or trademarks of ARM Ltd. Windows® and others are the registered trademarks of Microsoft Corporation in the US and/or other countries. Other terms and product names may be the trademarks of others.