Features

- Incorporates the ARM926EJ-S™ ARM® Thumb® Processor
  - DSP Instruction Extensions, ARM Jazelle® Technology for Java® Acceleration
  - 16 Kbyte Data Cache, 16 Kbyte Instruction Cache, Write Buffer
  - 220 MIPS at 200 MHz
  - Memory Management Unit
  - EmbeddedICE™ In-circuit Emulation, Debug Communication Channel Support
- Additional Embedded Memories
  - One 32 Kbyte Internal ROM, Two-cycle Access at Maximum Matrix Speed
  - One 32 Kbyte Internal SRAM, Single-cycle Access at Maximum Matrix Speed
- External Bus Interface (EBI)
  - EBI Supports Mobile DDR, SDRAM, Low Power SDRAM, Static Memory, Synchronous CellularRAM, ECC-enabled NAND Flash and CompactFlash™
- Metal Programmable (MP) Block
  - 500,000 Gates/250,000 Gates Metal Programmable Logic (through 5 Metal Layers) for AT91CAP9S500A/AT91CAP9S250A Respectively
  - Ten 512 x 36-bit Dual Port RAMs
  - Eight 512 x 72-bit Single Port RAMs
  - High Connectivity for Up to Three AHB Masters and Four AHB Slaves
  - Up to Seven AIC Interrupt Inputs
  - Up to Four DMA Hardware Handshake Interfaces
  - Delay Lines for Double Data Rate Interface
  - UTMI+ Full Connection
  - Up to 77 Dedicated I/Os
- LCD Controller
  - 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, Supports Wider Screen Buffers
- Image Sensor Interface
  - ITU-R BT. 601/656 External Interface, Programmable Frame Capture Rate
  - 12-bit Data Interface for Support of High Sensibility Sensors
  - SAV and EAV Synchronization, Preview Path with Scaler, YCbCr Format
- USB 2.0 Full Speed (12 Mbits per second) OHCI Host Double Port
  - Dual On-chip Transceivers
  - Integrated FIFOs and Dedicated DMA Channels
- USB 2.0 High Speed (480 Mbits per second) Device Port
  - On-chip Transceiver, 4 Kbyte Configurable Integrated DPRAM
  - Integrated FIFOs and Dedicated DMA Channels
  - Integrated UTMI+ Physical Interface
- Ethernet MAC 10/100 Base T
  - Media Independent Interface (MII) or Reduced Media Independent Interface (RMII)
  - 128-byte FIFOs and Dedicated DMA Channels for Receive and Transmit
- Multi-Layer Bus Matrix
  - Twelve 32-bit-layer Matrix, Allowing a Maximum of 38.4 Gbps of On-chip Bus Bandwidth at Maximum 100 MHz System Clock Speed
  - Boot Mode Select Option, Remap Command
- Fully-featured System Controller, Including
  - Reset Controller, Shutdown Controller

Customizable Microcontroller Processor

AT91CAP9S500A AT91CAP9S250A

Preliminary

6264C–CAP–24-Mar-09

- 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
- 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
  - 8 to 16 MHz On-chip Oscillator
  - Two PLLs up to 240 MHz
  - One USB 480 MHz PLL
- Power Management Controller (PMC)
  - Very Slow Clock Operating Mode, Software Programmable Power Optimization Capabilities
  - Four Programmable External Clock Signals
- Advanced Interrupt Controller (AIC)
  - Individually Maskable, Eight-level Priority, Vectored Interrupt Sources
  - Two External Interrupt Sources and One Fast Interrupt Source, Spurious Interrupt Protected
- Debug Unit (DBGU)
  - 2-wire UART and Support for Debug Communication Channel, Programmable ICE Access Prevention
- Periodic Interval Timer (PIT)
  - 20-bit Interval Timer plus 12-bit Interval Counter
- Watchdog Timer (WDT)
  - Key-protected, Programmable Only Once, Windowed 16-bit Counter Running at Slow Clock
- Real-Time Timer (RTT)
  - 32-bit Free-running Backup Counter Running at Slow Clock with 16-bit Prescaler
- Four 32-bit Parallel Input/Output Controllers (PIOA, PIOB, PIOC and PIOD)
  - 128 Programmable I/O Lines Multiplexed with up to Two Peripheral I/Os
  - Input Change Interrupt Capability on Each I/O Line
  - Individually Programmable Open-drain, Pull-up Resistor and Synchronous Output
- DMA Controller (DMAC)
  - Acts as one Bus Matrix Master
  - Embeds 4 Unidirectional Channels with Programmable Priority, Address Generation, Channel Buffering and Control
  - Supports Four External DMA Requests and Four Internal DMA Requests from the Metal Programmable Block (MPBlock)
- Twenty-two Peripheral DMA Controller Channels (PDC)
- One 2.0A and 2.0B Compliant CAN Controller
  - 16 Fully-programmable Message Object Mailboxes, 16-bit Time Stamp Counter
- Two Multimedia Card Interfaces (MCI)
  - SDCard/SDIO and MultiMedia™ Card 3.31 Compliant
  - Supports SDHC Devices
  - 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^2S 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

- **Three 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**
- **Two Master/Slave Serial Peripheral Interface (SPI)**
  - **8- to 16-bit Programmable Data Length, Four External Peripheral Chip Selects**
  - **Synchronous Communications at Up to 90 Mbits/sec**
- **One Three-channel 16-bit Timer/Counters (TC)**
  - **Three External Clock Inputs, Two Multi-purpose I/O Pins per Channel**
  - **Double PWM Generation, Capture/Waveform Mode, Up/Down Capability**
- **One Four-channel 20-bit PWM Controller (PWMC)**
- **One Two-wire Interface (TWI)**
  - **Master and Slave Mode Support, All Two-wire Atmel EEPROMs Supported**
- **One 8-channel, 10-bit Analog-to-Digital Converter (ADC)**
  - **Eight Channels Multiplexed with Digital I/Os**
- **IEEE® 1149.1 JTAG Boundary Scan on All Digital Pins**
- **Required Power Supplies:**
  - **1.08V to 1.32V for VDDCORE and VDDBU, VDDUPLL and VDDUTMIC**
  - **3.0V to 3.6V for VDDOSC, VDDPLL and VDDIOP0 (Peripheral I/Os) and VDDANA (ADC)**
  - **Programmable 1.65V to 1.95V or 3.0V to 3.6V for VDDIOP1 (Peripheral I/Os), VDDIOM (Memory I/Os) and VDDMPIOA/VDDMPIOB (MP Block I/Os)**
- **Available in 400-ball LFBGA RoHS-compliant Package**
- **Can also be Delivered in a 324-ball TFBGA RoHS-compliant Package According to User Needs**

1. Description

The AT91CAP9S500A/AT91CAP9S250A family is based on the integration of an ARM926EJ-S processor with fast ROM and SRAM memories, and a wide range of peripherals. By providing up to 500K gates of metal programmable logic, AT91CAP9S500A/AT91CAP9S250A is the ideal platform for creating custom designs.

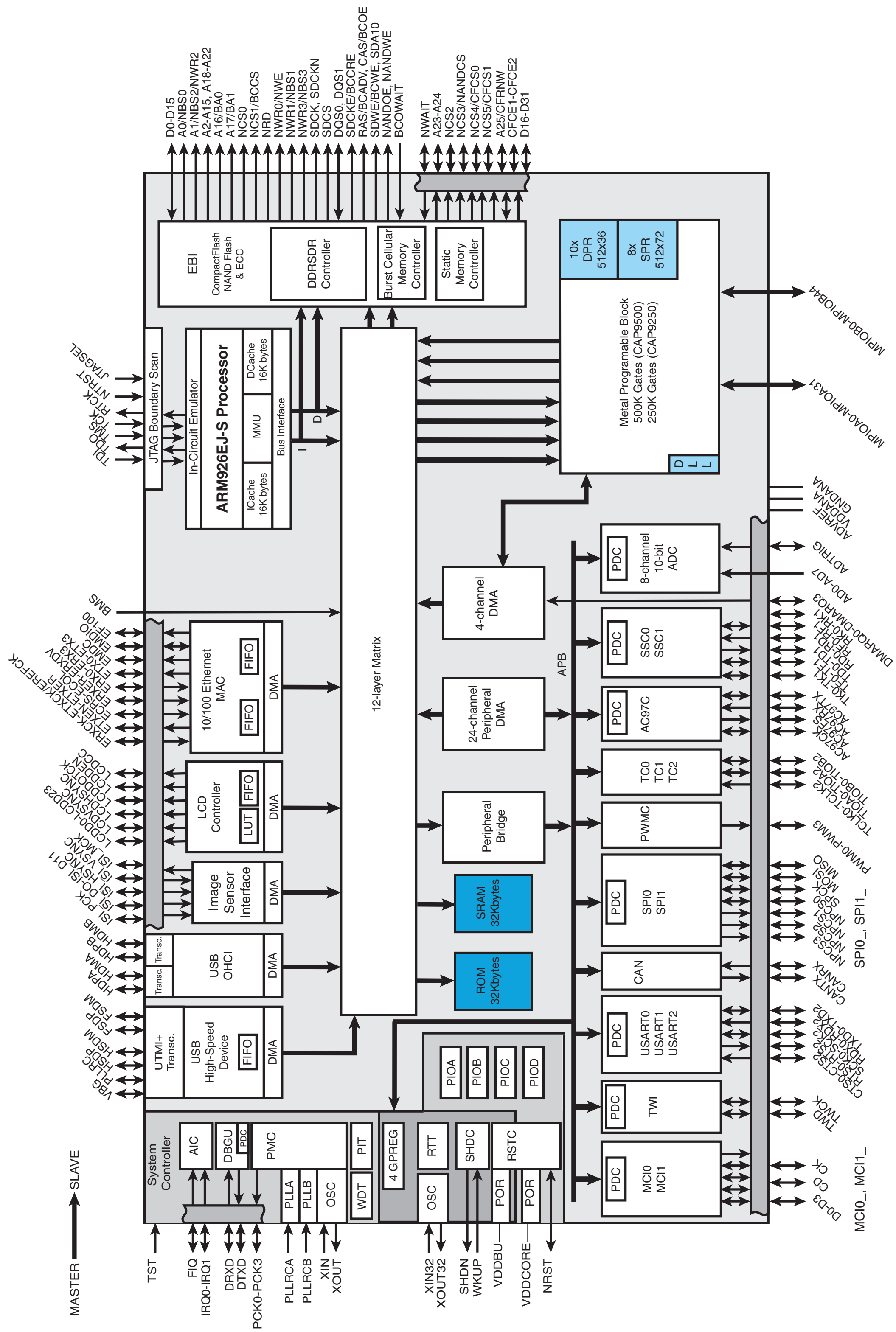
The AT91CAP9S500A/AT91CAP9S250A embeds a USB High-speed Device, a 2-port USB OHCI Host, an LCD Controller, a 4-channel DMA Controller, and one Image Sensor Interface. It also integrates several standard peripherals, such as USART, SPI, TWI, Timer Counters, PWM generators, Multimedia Card interface, and one CAN Controller.

The AT91CAP9S500A/AT91CAP9S250A is architectured on a 12-layer matrix, allowing a maximum internal bandwidth of twelve 32-bit buses. It also features one external memory bus (EBI) capable of interfacing with a wide range of memory devices.

The AT91CAP9S500A/AT91CAP9S250A is packaged in a 400-ball LFBGA RoHS-compliant package. It can also be delivered in a 324-ball TFBGA RoHS-compliant package according to the customer's requirements.

2. AT91CAP9S500A/AT91CAP9S250A Block Diagram

Figure 2-1. AT91CAP9S500A/AT91CAP9S250A Block Diagram



Note: 1. For information on signal multiplexing refer to Table 22-3, "EBI Pins and External Device Connections".

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
Power Supplies				
VDDIOM	EBI I/O Lines Power Supply	Power		1.65V to 3.6V
VDDIOP0	Peripherals I/O Lines Power Supply	Power		3.0V to 3.6V
VDDIOP1	Peripherals I/O Lines Power Supply	Power		1.65V to 3.6V
VDDIOMPA	MP Block I/O A Lines Power Supply	Power		1.65V to 3.6V
VDDIOMPB	MP Block I/O B Lines Power Supply	Power		1.65V to 3.6V
VDDBU	Backup I/O Lines Power Supply	Power		1.08V to 1.32V
VDDPLL	PLL 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
VDDUPLL	USB UTMI+ PLL Power Supply	Power		1.08V to 1.32V
VDDANA	ADC Analog Power Supply	Power		3.0V to 3.6V
VDDCORE	Core Chip Power Supply	Power		1.08V to 1.32V
GND	Ground	Ground		
GNDPLL	PLL Ground	Ground		
GNDUTMII	USB UTMI+ Interface Ground	Ground		
GNDUTMIC	USB UTMI+ Core Ground	Ground		
GNDUPLL	USB UTMI+ PLL Ground	Ground		
GNDANA	ADC Analog Ground	Ground		
GNDBU	Backup Ground	Ground		
GNDTHERMAL	Thermal Ground Ball	Ground		Thermally coupled with package substrate
Clocks, Oscillators and PLLs				
XIN	Main Oscillator Input	Input		
XOUT	Main Oscillator Output	Output		
XIN32	Slow Clock Oscillator Input	Input		
XOUT32	Slow Clock Oscillator Output	Output		
PLLRCA	PLL A Filter	Input		
PLLRCB	PLL B Filter	Input		
PCK0 - PCK3	Programmable Clock Output	Output		
Shutdown, Wakeup Logic				
SHDN	Shutdown Control	Output		Do not tie over VDDBU
WKUP	Wake-Up Input	Input		Accept between 0V and VDDBU

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
ICE and JTAG				
NTRST	Test Reset Signal	Input	Low	No pull-up resistor
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
JTAGSEL	JTAG Selection	Input		Pull-down resistor
RTCK	Return Test Clock	Output		
Reset/Test				
NRST	Microcontroller Reset	I/O	Low	Pull-up resistor
TST	Test Mode Select	Input		Pull-down resistor
BMS	Boot Mode Select	Input		Pull-up resistor
Debug Unit - DBGU				
DRXD	Debug Receive Data	Input		
DTXD	Debug Transmit Data	Output		
Advanced Interrupt Controller - AIC				
IRQ0 - IRQ1	External Interrupt Inputs	Input		
FIQ	Fast Interrupt Input	Input		
PIO Controller - PIOA - PIOB - PIOC - PIOD				
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 - PD31	Parallel IO Controller D	I/O		Pulled-up input at reset
Direct Memory Access Controller - DMA				
DMARQ0-DMARQ3	DMA Requests	Input		
External Bus Interface - EBI				
D0 - D31	Data Bus	I/O		Pulled-up input at reset
A0 - A25	Address Bus	Output		0 at reset
NWAIT	External Wait Signal	Input	Low	
Static Memory Controller - SMC				
NCS0 - NCS5	Chip Select Lines	Output	Low	
NWR0 - NWR3	Write Signal	Output	Low	
NRD	Read Signal	Output	Low	
NWE	Write Enable	Output	Low	
NBS0 - NBS3	Byte Mask Signal	Output	Low	

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
CompactFlash Support				
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		
CFCS0 - CFCS1	CompactFlash Chip Select Lines	Output	Low	
NAND Flash Support				
NANDCS	NAND Flash Chip Select	Output	Low	
NANDOE	NAND Flash Output Enable	Output	Low	
NANDWE	NAND Flash Write Enable	Output	Low	
DDR/SDRAM Controller				
SDCK	DDR/SDRAM Clock	Output		
SDCKN	DDR Inverted Clock	Output		
DQS0 - DQS1	DDR Data Qualifier Strobes	I/O		
DQM0 - DQM1	DDR/SDRAM Data Masks	Output		
DQM2 - DQM3	DDR/SDRAM Data Masks	Output		
SDCKE	DDR/SDRAM Clock Enable	Output	High	
SDCS	DDR/SDRAM Controller Chip Select	Output	Low	
BA0 - BA1	DDR/SDRAM Bank Select	Output		
SDWE	DDR/SDRAM Write Enable	Output	Low	
RAS - CAS	DDR/SDRAMRow and Column Signal	Output	Low	
SDA10	DDR/SDRAM Address 10 Line	Output		
Burst CellularRAM Controller				
BCCK	Burst CellularRAM Clock	Output		
BCCRE	Burst CellularRAM Enable	Output		
BCADV	Burst CellularRAM Burst Advance Signal	Output		
BCWE	Burst CellularRAM Write Enable	Output		
BCOE	Burst CellularRAM Output Enable	Output		
BCOWAIT	Burst CellularRAM Output Wait	Input		
Multimedia Card Interface MCI				
MCIx_CK	Multimedia Card Clock	Output		
MCIx_CD	Multimedia Card Command	I/O		
MCIx_D0 - D3	Multimedia Card Data	I/O		

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
Universal Synchronous Asynchronous Receiver Transmitter USART				
SCKx	USARTx Serial Clock	I/O		
TXDx	USARTx Transmit Data	I/O		
RXDx	USARTx Receive Data	Input		
RTSx	USARTx Request To Send	Output		
CTSx	USARTx Clear To Send	Input		
Synchronous Serial Controller - SSC				
TDx	SSCx Transmit Data	Output		
RDx	SSCx Receive Data	Input		
TKx	SSCx Transmit Clock	I/O		
RKx	SSCx Receive Clock	I/O		
TFx	SSCx Transmit Frame Sync	I/O		
RFx	SSCx Receive Frame Sync	I/O		
AC97 Controller - AC97C				
AC97RX	AC97 Receive Signal	Input		
AC97TX	AC97 Transmit Signal	Output		
AC97FS	AC97 Frame Synchronization Signal	Output		
AC97CK	AC97 Clock signal	Input		
Timer/Counter - TC				
TCLKx	TC Channel x External Clock Input	Input		
TIOAx	TC Channel x I/O Line A	I/O		
TIOBx	TC Channel x I/O Line B	I/O		
Pulse Width Modulation Controller- PWMC				
PMWx	Pulse Width Modulation Output	Output		
Serial Peripheral Interface - SPI				
SPIx_MISO	Master In Slave Out	I/O		
SPIx_MOSI	Master Out Slave In	I/O		
SPIx_SPCK	SPI Serial Clock	I/O		
SPIx_NPCS0	SPI Peripheral Chip Select 0	I/O	Low	
SPIx_NPCS1 - SPIx_NPCS3	SPI Peripheral Chip Select	Output	Low	
Two-Wire Interface - TWI				
TWD	Two-wire Serial Data	I/O		
TWCK	Two-wire Serial Clock	I/O		

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
CAN Controller				
CANRX	CAN input	Input		
CANTX	CAN output	Output		
LCD Controller - LCDC				
LCDD0 - LCDD23	LCD Data Bus	Input		
LCDVSYNC	LCD Vertical Synchronization	Output		
LCDHSYNC	LCD Horizontal Synchronization	Output		
LCDDOTCK	LCD Dot Clock	Output		
LCDDEN	LCD Data Enable	Output		
LCDCC	LCD Contrast Control	Output		
Ethernet 10/100 EMAC				
ETXCK/EREFCK	Transmit Clock or Reference Clock	Input		MII only, REFCK in RMII
ERXCK	Receive Clock	Input		MII only
ETXEN	Transmit Enable	Output		
ETX0-ETX3	Transmit Data	Output		ETX0-ETX1 only in RMII
ETXER	Transmit Coding Error	Output		MII only
ERXDV	Receive Data Valid	Input		RXDV in MII, CRSDV in RMII
ERX0-ERX3	Receive Data	Input		ERX0-ERX1 only in RMII
ERXER	Receive Error	Input		
ECRS	Carrier Sense and Data Valid	Input		MII only
ECOL	Collision Detect	Input		MII only
EMDC	Management Data Clock	Output		
EMDIO	Management Data Input/Output	I/O		
EF100	Force 100Mbit/sec.	Output	High	RMII only
USB High Speed Device				
FSDM	USB Full Speed Data -	Analog		
FSDP	USB Full Speed Data +	Analog		
HSDM	USB High Speed Data -	Analog		
HSDP	USB High Speed Data +	Analog		
VBG	Bias Voltage Reference	Analog		
PLLRCU	USB PLL Test Pad	Analog		

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
OHCI USB Host Port				
HDPA	USB Host Port A Data +	Analog		
HDMA	USB Host Port A Data -	Analog		
HDPB	USB Host Port B Data +	Analog		
HDMB	USB Host Port B Data -	Analog		
ADC				
AD0-AD7	Analog Inputs	Analog		
ADVREF	ADC Voltage Reference	Analog		
ADTRIG	ADC Trigger	Input		
Image Sensor Interface - ISI				
ISI_D0-ISI_D11	Image Sensor Data	Input		
ISI_MCK	Image Sensor Reference Clock	Output		
ISI_HSYNC	Image Sensor Horizontal Synchro	Input		
ISI_VSYNC	Image Sensor Vertical Synchro	Input		
ISI_PCK	Image Sensor Data Clock	Input		
MPBLOCK - MPB				
MPIOA0-MPIOA31	MPBlock I/Os A	I/O		
MPIOB0-MPIOB44	MPBlock I/Os B	I/O		

4. Package and Pinout

The AT91CAP9S500A/AT91CAP9S250A is available in two packages:

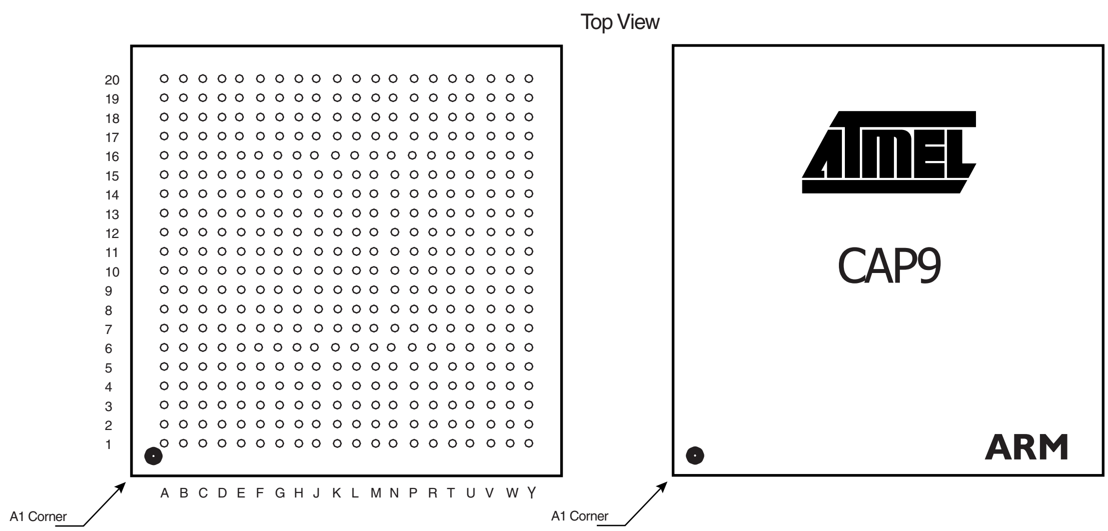
- a 400-ball RoHS-compliant LFBGA package, 17 x 17 mm, 0.8 mm ball pitch
- a 324-ball RoHS-compliant TFBGA package, 15 x 15 mm, 0.8 mm ball pitch

4.1 400-ball LFBGA Package Outline

Figure 4-1 shows the orientation of the 400-ball BGA Package.

A detailed mechanical description is given in the section "AT91CAP9S500A/AT91CAP9S250A Mechanical Characteristics" of the product datasheet.

Figure 4-1. 400-ball LFBGA Package Outline and Marking (Top View)



4.2 400-ball LFBGA Package Pinout

Table 4-1. AT91CAP9S500A/AT91CAP9S250A Pinout for 400-ball BGA Package

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
A1	PC5	F1	PA3	L1	PA22	T1	PD22
A2	PC3	F2	PA4	L2	PA25	T2	PD23
A3	PC2	F3	PA8	L3	PA29	T3	PD30
A4	PC1	F4	PA5	L4	PA31	T4	VDDCORE
A5	PC0	F5	PA6	L5	PD6	T5	SDCS
A6	BMS	F6	VDDIOM	L6	GNDIO	T6	DQS0
A7	NRST	F7	VDDIOP0	L7	GNDCORE	T7	D4
A8	GNDCORE	F8	PC24	L8	PA18	T8	D11
A9	PB18	F9	NC	L9	GNDTHERMAL	T9	D14
A10	PB17	F10	VDDCORE	L10	GNDTHERMAL	T10	SDA10
A11	PB14	F11	GNDIO	L11	GNDTHERMAL	T11	VDDCORE
A12	PB15	F12	PB23	L12	GNDTHERMAL	T12	MPIOA0
A13	GNDANA	F13	PB6	L13	GNDCORE	T13	MPIOA9
A14	PB26	F14	NC	L14	GNDIO	T14	GNDIO
A15	VDDIOP0	F15	NC	L15	VDDCORE	T15	MPIOA25
A16	GNDIO	F16	NC	L16	MPIOB28	T16	MPIOA24
A17	FSDP	F17	GNDPLL	L17	MPIOB32	T17	MPIOA29
A18	FSDM	F18	WKUP0	L18	MPIOB34	T18	MPIOB3
A19	HSDP	F19	SHDW	L19	MPIOB31	T19	MPIOB17
A20	HSDM	F20	PLLRCA	L20	MPIOB29	T20	MPIOB18
B1	PC17	G1	PA7	M1	PA26	U1	PD25
B2	PC16	G2	PA10	M2	PA30	U2	PD31
B3	PC14	G3	PA11	M3	PD11	U3	BCCLK
B4	PC11	G4	PA9	M4	PD12	U4	A0
B5	PC10	G5	PA12	M5	PD13	U5	D0
B6	PC9	G6	PD10	M6	PD15	U6	D1
B7	TDO	G7	GNDIO	M7	GNDCORE	U7	NWR1
B8	TCK	G8	GNDCORE	M8	PA28	U8	DQS1
B9	PB20	G9	VDDIOP0	M9	GNDTHERMAL	U9	A7
B10	PB19	G10	PC8	M10	GNDTHERMAL	U10	A13
B11	PB13	G11	PB25	M11	GNDTHERMAL	U11	A20
B12	ADVREF	G12	PB21	M12	GNDTHERMAL	U12	GNDIO
B13	PB16	G13	PB8	M13	NRD	U13	MPIOA4
B14	PB27	G14	PB0	M14	MPIOB26	U14	MPIOA11
B15	PB24	G15	PB2	M15	GNDIO	U15	MPIOA16
B16	HDMA	G16	NC	M16	MPIOB16	U16	VDDMPIOA
B17	VDDIOP0	G17	VDDPLL	M17	GNDCORE	U17	MPIOA23

Table 4-1. AT91CAP9S500A/AT91CAP9S250A Pinout for 400-ball BGA Package (Continued)

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
B18	GNDIO	G18	GNDCORE	M18	MPIOB27	U18	MPIOA28
B19	VDDUTMII	G19	TST	M19	MPIOB25	U19	MPIOB6
B20	GNDUTMII	G20	PLLRCB	M20	MPIOB24	U20	MPIOB9
C1	PC23	H1	PA13	N1	PD7	V1	PD26
C2	PC22	H2	PA14	N2	PD8	V2	RAS
C3	PC21	H3	PD0	N3	PD16	V3	SDCKE
C4	PC20	H4	PA15	N4	PD19	V4	D3
C5	PC18	H5	PD1	N5	PD20	V5	VDDIOM
C6	PC15	H6	VDDIOP1	N6	PD29	V6	D5
C7	PC12	H7	VDDCORE	N7	GNDIO	V7	D9
C8	PC6	H8	GNDIO	N8	VDDIOM	V8	D15
C9	NTRST	H9	GNDIO	N9	NCS1	V9	A11
C10	TDI	H10	PB10	N10	VDDCORE	V10	GNDCORE
C11	VDDANA	H11	PB4	N11	A3	V11	A22
C12	PB12	H12	VDDMPIOB	N12	A6	V12	MPIOA1
C13	PB29	H13	JTAGSEL	N13	VDDCORE	V13	MPIOA6
C14	PB9	H14	GNDCORE	N14	MPIOB11	V14	MPIOA10
C15	PB7	H15	GNDPLL	N15	MPIOB13	V15	MPIOA13
C16	HDPA	H16	NC	N16	MPIOB12	V16	MPIOA17
C17	HDPB	H17	VDDCORE	N17	MPIOB14	V17	MPIOA20
C18	VDDUPLL	H18	MPIOB44	N18	MPIOB15	V18	MPIOA27
C19	VDDUTMIC	H19	XOUT32	N19	MPIOB22	V19	MPIOB5
C20	VBG	H20	XIN32	N20	MPIOB23	V20	VDDMPIOB
D1	PC29	J1	PD3	P1	PD9	W1	SDWE
D2	PC28	J2	PD2	P2	PD14	W2	BCOWAIT
D3	PC27	J3	PD5	P3	PD18	W3	NANDWE
D4	PC26	J4	PA17	P4	PD27	W4	GNDIO
D5	PC25	J5	PA19	P5	PD28	W5	D6
D6	PC19	J6	VDDIOP0	P6	VDDIOM	W6	A2
D7	NANDOE	J7	PA16	P7	NWR3	W7	A5
D8	PC7	J8	GNDCORE	P8	D8	W8	A14
D9	GNDIO	J9	GNDTHERMAL	P9	D10	W9	A17
D10	TMS	J10	GNDTHERMAL	P10	GNDIO	W10	A19
D11	NC	J11	GNDTHERMAL	P11	A9	W11	NWR0
D12	PB31	J12	GNDTHERMAL	P12	A12	W12	MPIOA2
D13	PB22	J13	GNDIO	P13	NC	W13	MPIOA5
D14	VDDCORE	J14	GNDBU	P14	MPIOB8	W14	MPIOA8
D15	PB3	J15	GNDBU	P15	MPIOB0	W15	MPIOA12

Table 4-1. AT91CAP9S500A/AT91CAP9S250A Pinout for 400-ball BGA Package (Continued)

Pin	Signal Name
D16	PB1
D17	HDMB
D18	PLLRCU
D19	GNDUTMIC
D20	GNDUPLL
E1	PC30
E2	PA2
E3	PA1
E4	PA0
E5	PC31
E6	GNDIO
E7	VDDCORE
E8	PC13
E9	PC4
E10	RTCK
E11	VDDIOP0
E12	PB30
E13	PB28
E14	PB11
E15	PB5
E16	NC
E17	VDDPLL
E18	VDDBU
E19	XIN
E20	XOUT

Pin	Signal Name
J16	MPIOB42
J17	MPIOB39
J18	MPIOB43
J19	MPIOB41
J20	GNDIO
K1	PD4
K2	PA21
K3	PA24
K4	PA27
K5	PA23
K6	GNDIO
K7	PA20
K8	VDDCORE
K9	GNDTHERMAL
K10	GNDTHERMAL
K11	GNDTHERMAL
K12	GNDTHERMAL
K13	GNDCORE
K14	MPIOB33
K15	MPIOB30
K16	MPIOB35
K17	MPIOB38
K18	MPIOB40
K19	MPIOB37
K20	MPIOB36

Pin	Signal Name
P16	MPIOB1
P17	MPIOB7
P18	MPIOB10
P19	MPIOB21
P20	VDDMPIOB
R1	PD21
R2	PD17
R3	PD24
R4	CAS
R5	VDDCORE
R6	D2
R7	D7
R8	VDDIOM
R9	D13
R10	D12
R11	VDDIOM
R12	A16
R13	VDDIOM
R14	NC
R15	NC
R16	NC
R17	MPIOB2
R18	MPIOB4
R19	MPIOB19
R20	MPIOB20

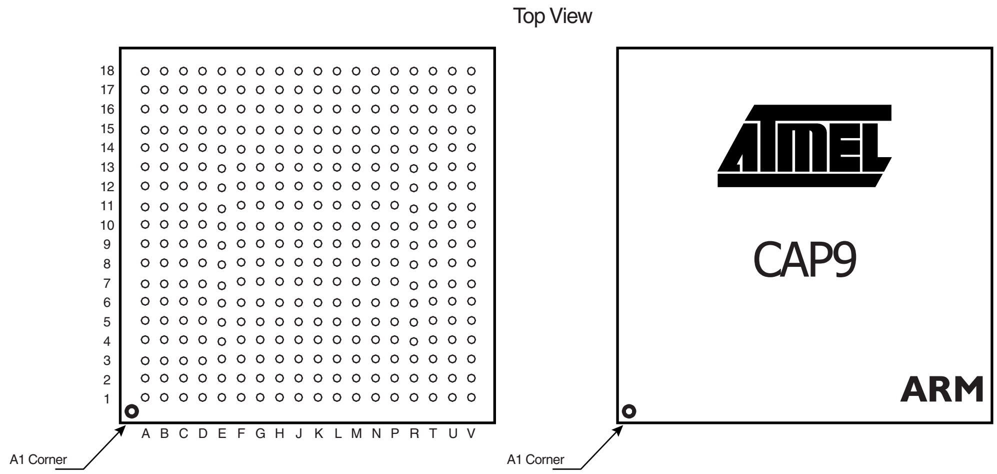
Pin	Signal Name
W16	MPIOA15
W17	MPIOA21
W18	MPIOA22
W19	GNDIO
W20	VDDCORE
Y1	SDCK
Y2	SDCKN
Y3	A1
Y4	GNDCORE
Y5	A4
Y6	A8
Y7	A10
Y8	A15
Y9	A18
Y10	A21
Y11	NCS0
Y12	MPIOA3
Y13	MPIOA7
Y14	VDDMPIOA
Y15	MPIOA14
Y16	MPIOA18
Y17	MPIOA19
Y18	MPIOA26
Y19	MPIOA30
Y20	MPIOA31

4.3 324-ball TFBGA Package Outline

Figure 4-2 shows the orientation of the 324-ball TFBGA green package.

A detailed mechanical description is given in the section "AT91CAP9S500A/AT91CAP9S250A Mechanical Characteristics" of the product datasheet.

Figure 4-2. 324-ball TFBGA Package Outline and Marking (Top View)



4.4 324-ball TFBGA Package Pinout

The pin assignment for the 324-ball TFBGA package is customizable and dependent upon the needs of the user.

Important: It is possible to partially or totally remove the connections to dedicated Metal Programmable I/0s: MPIOAO-MPIOA31 and MPIOB0-MPIOB44. Likewise, PA16-PA31, PB21-PB31, PDC0-PC27, PD0-PD10 can be partially or totally disconnected. However, it is incumbent upon the user to ensure that the associated functionality removed is not needed for the intended application. Refer to Section 10.3.1 on page 42, Section 10.3.2 on page 43, Section 10.3.3 on page 44, Section 10.3.4 on page 45 for information on PIO multiplexing and to verify functionality before disconnecting signals.

5. Power Considerations

5.1 Power Supplies

The AT91CAP9S500A/AT91CAP9S250A has several types of power supply pins:

- VDDCORE pins: Power the core, including the processor, the embedded memories and the peripherals; voltage range between1.08V 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).
- VDDIOP0 pins: Power the Peripherals I/O lines and the USB transceivers; voltage range between 3.0V and 3.6V, 3.3V nominal.
- VDDIOP1 pins: Power the Peripherals I/O lines involving the Image Sensor Interface; voltage ranges from 1.65V to 3.6V, 1.8V, 2.5V, 3V or 3.3V nominal.
- VDDIOMPA pins: Power the MP Block I/O A lines; voltage ranges from 1.65V to 3.6V, 1.8V, 2.5V, 3V or 3.3V nominal.
- VDDIOMPB pins: Power the dedicated MP Block I/O B lines; voltage ranges from 1.65V to 3.6V, 1.8V, 2.5V, 3V or 3.3V nominal.
- VDDBU pin: Powers the Slow Clock oscillator and a part of the System Controller; voltage range between1.08V and 1.32V, 1.2V nominal.
- VDDPLL pin: Powers the PLL cells; voltage ranges between 3.0V to 3.6V, 3.3V nominal.
- VDDUTMII pin: Powers the UTMI+ interface; voltage ranges from 3.0V to 3.6V, 3.3V nominal.
- VDDUTMIC pin: Powers the UTMI+ core; voltage ranges between 1.08V and 1.32V, 1.2V nominal.
- VDDUPLL pin: Powers the USB PLL cell; voltage ranges between 1.08V and 1.32V, 1.2V nominal.
- VDDANA pin: Powers the ADC cell; voltage ranges between 3.0V and 3.6V, 3.3V nominal.

The power supplies VDDIOM, VDDIOP0 and VDDIOP1 are identified in the pinout table and the multiplexing tables. These supplies enable the user to power the device differently for interfacing with memories and for interfacing with peripherals.

Ground pins GNDIO are common to VDDIOM, VDDIOP0, VDDIOP1, VDDIOMPA and VDDIOMPB pin power supplies. Separated ground pins are provided for VDDCORE, VDDBU, VDDPLL, VDDUTMII, VDDUTMIC, VDDUPLL and VDDANA. These ground pins are, respectively, GNDBU, GNDOSC, GNDPLL, GNDUTMII, GNDUTMIC, GNDUPLL and GNDANA.

Special GNDTHERMAL ground balls are thermally coupled with package substrate.

5.2 Power Consumption

The AT91CAP9S500A/AT91CAP9S250A consumes about 190 µA of static current on VDDCORE at 25°C.

On VDDBU, the current does not exceed 4 µA @25°C.

For dynamic power consumption and more details, refer to the Power Consumption section and tables in the Electrical Characteristics section of the product datasheet.

5.3 Programmable I/O Lines Power Supplies

The power supply pins VDDIOM, VDDMPIOA and VDDMPIOB accept two voltage ranges. This allows the device to reach its maximum speed either out of 1.8V or 3.3V external memories.

The target maximum speed is 100 MHz on the pin DDR/SDR and MPIOA or MPIOB pins loaded with 30 pF for power supply at 1.8V and 50 pF for power supply at 3.3V. The other signals (control, address and data signals) do not go over 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. Obviously, the device cannot reach its maximum speed if the voltage supplied to the pins is 1.8V only. The user must make sure to program the EBI voltage range before getting the device out of its Slow Clock Mode.

6. I/O Line Considerations

6.1 JTAG Port Pins

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

TDO and RTCK are outputs, driven at up to VDDIOP0, and have no pull-up resistors.

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

The NTRST signal is described in Section 6.3 "Reset Pins" on page 18.

All the JTAG signals are supplied with VDDIOP0.

6.2 Test Pin

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

This pin is supplied with VDDBU.

6.3 Reset Pins

NRST is an open-drain output integrating a non-programmable pull-up resistor. It can be driven with voltage at up to VDDIOP0.

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

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

The NRST pin integrates a permanent pull-up resistor of 90 kΩ minimum to VDDIOP0.

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 of 90 kΩ minimum. Programming of this pull-up resistor is performed independently for each I/O line through the PIO Controllers.

After reset, all the I/O lines default as inputs with pull-up resistors enabled, except those multiplexed with the External Bus Interface signals that must be enabled as Peripheral at reset. This is 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)
- 16-Kbyte Data Cache, 16-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 Bus Matrix

- 12-layer Matrix, handling requests from 12 masters
- Programmable Arbitration strategy
  - Fixed-priority Arbitration

– Round-Robin Arbitration, either with no default master, last accessed default master or fixed default master

- Burst Management
  – Breaking with Slot Cycle Limit Support
  – Undefined Burst Length Support
- One Address Decoder provided per Master
  – Three different slaves may be assigned to each decoded memory area: one for internal boot, one for external boot, one after remap
- Boot Mode Select
  – Non-volatile Boot Memory can be internal or external
  – Selection is made by BMS pin sampled at reset
- Remap Command
  – Allows Remapping of an Internal SRAM in Place of the Boot Non-Volatile Memory
  – Allows Handling of Dynamic Exception Vectors

7.3 Matrix Masters

The Bus Matrix of the AT91CAP9S500A/AT91CAP9S250A manages twelve Masters and thus each master can perform an access concurrently with the others, assuming that the slave it accesses is available.

Each Master has its own decoder, which is defined specifically for each master. In order to simplify the addressing, all the masters have the same decoding.

Table 7-1. List of Bus Matrix Masters

Master 0	ARM926™ Instruction
Master 1	ARM926 Data
Master 2	Peripheral DMA Controller
Master 3	LCD Controller
Master 4	USB High Speed Device Controller
Master 5	Image Sensor Interface
Master 6	DMA Controller
Master 7	Ethernet MAC
Master 8	OHCI USB Host Controller
Master 9	MP Block Master 0
Master 10	MP Block Master 1
Master 11	MP Block Master 2

7.4 Matrix Slaves

The Bus Matrix of the AT91CAP9S500A/AT91CAP9S250A manages ten Slaves. Each Slave has its own arbiter, thus permitting a different arbitration per Slave to be programmed.

The LCD Controller, the USB Host and the USB High Speed Device have a user interface mapped as a Slave of the Matrix. They share the same layer, as programming them does not require a high bandwidth.

Table 7-2. List of Bus Matrix Slaves

Slave 0	Internal SRAM 32 Kbytes
Slave 1	MP Block Slave 0 (MP Block Internal Memories)
Slave 2	Internal ROM
	LCD Controller User Interface
	USB High Speed Device Interface
	OHCI USB Host Interface
Slave 3	MP Block Slave 1 (MP Block Internal Memories)
Slave 4	External Bus Interface
Slave 5	DDR Controller Port 2
Slave 6	DDR Controller Port 3
Slave 7	MP Block Slave 2 (MP Block External Chip Selects)
Slave 8	MP Block Slave 3 (MP Block Internal Peripherals)
Slave 9	Internal Peripherals for AT91CAP9

7.5 Master-to-Slave Access

All the Masters can normally access all the Slaves. However, some paths do not make sense, such as allowing access from the Ethernet MAC to the Internal Peripherals. Thus, these paths are forbidden or simply not wired, and shown as "-" in Table 7-3, "AT91CAP9S500A/AT91CAP9S250A Masters to Slaves Access," on page 22.

Table 7-3. AT91CAP9S500A/AT91CAP9S250A Masters to Slaves Access

Master		0	1	2	3	4	5	6	7	8	9	10	11
Slave		ARM926 Instruction	ARM926 Data	Peripheral DMA Ctrl	LCDCtrl	USB High Speed Device Ctrl	Image Sensor Interface	DMA Ctrl	Ethernet MAC	OHCI USB Host Ctrl	MP Block Master 0	MP Block Master 1	MP Block Master 2
0	Internal SRAM 32 Kbytes	X	X	X	X	X	X	X	X	X	X	X	X
1	MP Block Slave 0	X	X	X	X	X	X	X	X	X	X	X	X
2	Internal ROM	X	X	X	X	X	X	X	X	X	X	X	X
	LCD Controller User Interface	X	X	-	-	-	-	-	-	-	X	X	X
	USB High Speed Device Interface	X	X	-	-	-	-	X	-	-	X	X	X
	OHCI USB Host Interface	X	X	-	-	-	-	-	-	-	X	X	X
3	MPBlock Slave 1	X	X	X	X	X	X	X	X	X	X	X	X
4	External Bus Interface	X	X	X	X	X	X	X	X	X	X	X	X
-													
-	DDR Port 0	X	-	-	-	-	-	-	-	-	-	-	-
5	DDR Port 1	-	X	-	-	-	-	-	-	-	-	-	-
6	DDR Port 2			X[1]	X[1]	X[1]	X[1]	X[1]	X[1]	X[1]	X[1]	X[1]	X[1]
	DDR Port 3			X[1]	X[1]	X[1]	X[1]	X[1]	X[1]	X[1]	X[1]	X[1]	X[1]
7	MPBlock Slave 2	X	X	X	X	X	X	X	X	X	X	X	X
8	MPBlock Slave 3	X	X	X	X	X	X	X	X	X	X	X	X
9	Internal Peripherals	X	X	X	-	-	-	X	-	-	X	X	X

Note: 1. DDR Port 2 or Port 3 is selectable for each master through the Matrix Remap Control Register.

7.6 Peripheral DMA Controller

- Acting as one Matrix Master
- Allows data transfers from/to peripheral to/from any memory space without any intervention of the processor.
- Next Pointer Support, forbids strong real-time constraints on buffer management.
- Twenty-two Channels
  - Two for each USART
  - Two for the Debug Unit
  - One for the TWI
  - One for the ADC Controller
  - Two for the AC97 Controller
  - Two for each Serial Synchronous Controller
  - Two for each Serial Peripheral Interface
  - One for the each Multimedia Card Interface

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

- DBGU Transmit Channel
- USART2 Transmit Channel
- USART1 Transmit Channel
- USART0 Transmit Channel
- AC97 Transmit Channel
- SPI1 Transmit Channel
- SPI0 Transmit Channel
- SSC1 Transmit Channel
- SSC0 Transmit Channel
- DBGU Receive Channel
- TWI Transmit/Receive Channel
- ADC Receive Channel
- USART2 Receive Channel
- USART1 Receive Channel
- USART0 Receive Channel
- AC97 Receive Channel
- SPI1 Receive Channel
- SPI0 Receive Channel
- SSC1 Receive Channel
- SSC0 Receive Channel
- MCI1 Transmit/Receive Channel
- MCI0 Transmit/Receive Channel

7.7 DMA Controller

- Acting as one Matrix Master
- Embeds 4 unidirectional channels with programmable priority
- Address Generation
  - Source / destination address programming
  - Address increment, decrement or no change
  - DMA chaining support for multiple non-contiguous data blocks through use of linked lists
  - Scatter support for placing fields into a system memory area from a contiguous transfer. Writing a stream of data into non-contiguous fields in system memory
  - Gather support for extracting fields from a system memory area into a contiguous transfer
  - User enabled auto-reloading of source, destination and control registers from initially programmed values at the end of a block transfer
  - Auto-loading of source, destination and control registers from system memory at end of block transfer in block chaining mode
  - Unaligned system address to data transfer width supported in hardware
- Channel Buffering
  - 8-word FIFO
  - Automatic packing/unpacking of data to fit FIFO width
- Channel Control
  - Programmable multiple transaction size for each channel
  - Support for cleanly disabling a channel without data loss
  - Suspend DMA operation
  - Programmable DMA lock transfer support
- Transfer Initiation
  - Support four External DMA Requests and four Internal DMA request from the MP Block
  - Support for Software handshaking interface. Memory mapped registers can be used to control the flow of a DMA transfer in place of a hardware handshaking interface
- Interrupt
  - Programmable Interrupt generation on DMA Transfer completion Block Transfer completion, Single/Multiple transaction completion or Error condition

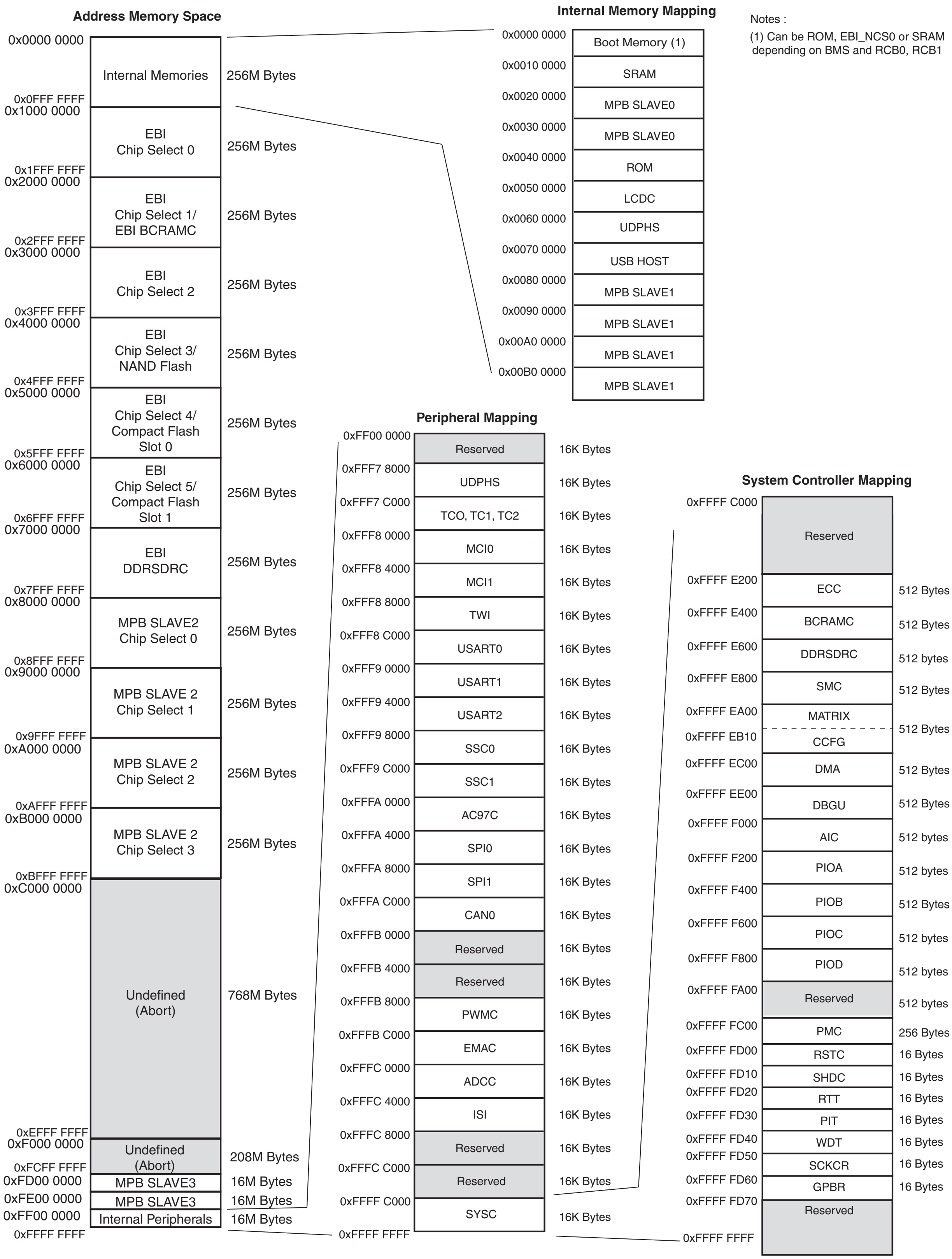
7.8 Debug and Test Features

- 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. AT91CAP9S500A/AT91CAP9S250A Memory Mapping



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

Decoding breaks up the 4G bytes of address space into 16 banks of 256M bytes. The banks 1 to 7 are directed to the EBI that associates these banks to the external chip selects NCS0 to NCS5 and SDCS. 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 banks 8 to 11 are directed to MP Block (Slave 2) and may be used to address external memories. The bank 15 is split into three parts, one reserved for the peripherals that provides access to the Advanced Peripheral Bus (APB), the two others are directed to MP Block (Slave 3) and may provide access to the MP Block APB or to other AHB peripherals.

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

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

Regarding Master 0 and Master 1 (ARM926 Instruction and Data), three different Slaves are assigned to the memory space decoded at address 0x0: one for internal boot, one for external boot and one after remap. Refer to Table 8-1, “Internal Memory Mapping,” on page 28 for details.

8.1 Embedded Memories

- 32 Kbyte ROM
  - Two Cycle Access at full matrix speed
- 32 Kbyte Fast SRAM
  - Single Cycle Access at full matrix speed
- 20 Kbyte MP Block Fast Dual Port RAM (ten 512x36 DPR instances)
  - Used as Dual Port RAM completely managed by MP Block
- 32 Kbyte MP Block Fast Single Port RAM (eight 512x72 SPR instances)
  - Used as Single Port RAM completely managed by MP Block

8.1.1 Internal Memory Mapping

Table 8-1 summarizes the Internal Memory Mapping, depending on the Remap Command Bit (RBC) status and the BMS state at reset.

REMAP allows the user to layout the internal SRAM bank to 0x0 to ease development. This is done by software once the system boots. 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 way of hardware at reset.

Table 8-1. Internal Memory Mapping

Address	ARM926 I			ARM926 D			Other Masters
0x0000 0000	RCB0 = 0		RCB0 = 1	RCB1 = 0		RCB1 = 1	
	BMS = 0	BMS = 1		BMS = 0	BMS = 1		
	NCS0	ROM	SRAM	NCS0	ROM	SRAM	Abort

8.1.1.1 Internal 32 Kbyte Fast SRAM

The AT91CAP9S500A/AT91CAP9S250A integrates a 32 Kbyte SRAM, mapped at address 0x0010 0000, which is accessible from the AHB bus. This SRAM is single cycle accessible at full matrix speed.

8.1.1.2 Internal ROM

The AT91CAP9S500A/AT91CAP9S250A 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.

The AT91CAP9S500A/AT91CAP9S250A Bus Matrix manages a boot memory that depends on the level on the BMS pin at reset. The internal memory area mapped between address 0x0 and 0x000F FFFF is reserved to this effect.

Note: Memory blocks not affected by these parameters can always be seen at their specified base addresses. See the complete memory map presented in Figure 8-1 on page 26.

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 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 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
  - NAND Flash
  - SDCard on MCI0
  - SPI DataFlash®/Serial Flash connected on NPCS0 and NPCS1 of the SPI0
  - EEPROM on TWI
- 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:

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

Refer to Figure 8-1 on page 26.

8.2.1 External Bus Interface

The AT91CAP9S500A/AT91CAP9S250A features one External Bus Interface to offer high bandwidth to the system and to prevent any bottleneck while accessing the external memories.

- Optimized for Application Memory Space support
- Integrates four External Memory Controllers:
  - Static Memory Controller
  - 4-port DDR/SDRAM Controller
  - Burst/Cellular RAM Controller
  - SLC NAND Flash ECC Controller
- Additional logic for NAND Flash and CompactFlash™
- Optional Full 32-bit External Data Bus
- Up to 26-bit Address Bus (up to 64 Mbytes linear per chip select)
- Up to 6 chip selects, configurable assignment:
  - Static Memory Controller on NCS0
  - Burst/CellularRAM Controller or Static Memory Controller on NCS1
  - Static Memory Controller on NCS2

– Static Memory Controller on NCS3, Optional NAND Flash support
  – Static Memory Controller on NCS4 - NCS5, Optional CompactFlash support
- One dedicated chip select:
  – DDR/SDRAM Controller on SDCS

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
  – Compliant with LCD Module
  – Control signals programmable setup, pulse and hold time for each Memory Bank
- Multiple Wait State Management
  – Programmable Wait State Generation
  – External Wait Request
  – Programmable Data Float Time
- Slow Clock mode supported

8.2.3 DDR/SDRAM Controller

- Supported devices:
  – Standard and Low Power SDRAM (Mobile SDRAM)
  – Mobile DDR
- Numerous configurations supported
  – 2K, 4K, 8K Row Address Memory Parts
  – SDRAM with two or four Internal Banks
  – SDRAM with 16- or 32-bit Data Path
  – Mobile DDR with four Internal Banks
  – Mobile DDR with 16-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
  – Multiport (4 Ports)
- Energy-saving capabilities
  – Self-refresh, power down and deep power down modes supported
- Error detection
  – Refresh Error Interrupt
- DDR/SDRAM Power-up Initialization by software

- SDRAM CAS Latency of 1, 2 and 3 supported
- DDR CAS latency of 3 supported
- Auto Precharge Command not used

8.2.4 Burst Cellular RAM Controller

- Supported devices:
  – Synchronous Cellular RAM version 1.0, 1.5 and 2.0
- Numerous configurations supported
  – 64K, 128K, 256K, 512K Row Address Memory Parts
  – Cellular RAM with 16- or 32-bit Data Path
- Programming facilities
  – Word, half-word, byte access
  – Automatic page break when Memory Boundary has been reached
  – Timing parameters specified by software
  – Only Continuous read or write burst supported
- Energy-saving capabilities
  – Standby and Deep Power Down (DPD) modes supported
  – Low Power features (PASR/TCSR) supported
- Cellular RAM Power-up Initialization by hardware
- Cellular RAM CAS latency of 2 and 3 supported (Version 1.0)
- Cellular RAM CAS latency of 2, 3, 4, 5 and 6 supported (Version 1.5 and 2.0)
- Cellular RAM variable or fixed latency supported (Version 1.5 and 2.0)
- Multiplexed address/data bus supported (Version 2.0)
- Asynchronous and Page mode not supported

8.2.5 NAND Flash Error Corrected Code Controller

- Hardware Error Corrected Code (ECC) Generation
  – Detection and Correction by Software
- Supports NAND Flash and SmartMedia™ Devices with 8- or 16-bit Data Path
- Supports NAND Flash/SmartMedia with Page Sizes of 528, 1056, 2112 and 4224 Bytes Specified by Software
- Supports 1 bit correction for a page of 512,1024,2048 and 4096 Bytes with 8- or 16-bit Data Path
- Supports 1 bit correction per 512 bytes of data for a page size of 512, 2048 and 4096 Bytes with 8-bit Data Path
- Supports 1 bit correction per 256 bytes of data for a page size of 512, 2048 and 4096 Bytes with 8-bit Data Path

9. System Controller

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

The System Controller User Interface also embeds the registers that allow configuration of the Matrix and a set of registers for the chip configuration. The chip configuration registers are used to configure:

– EBI chip select assignment and voltage range for external memories

– MP Block

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

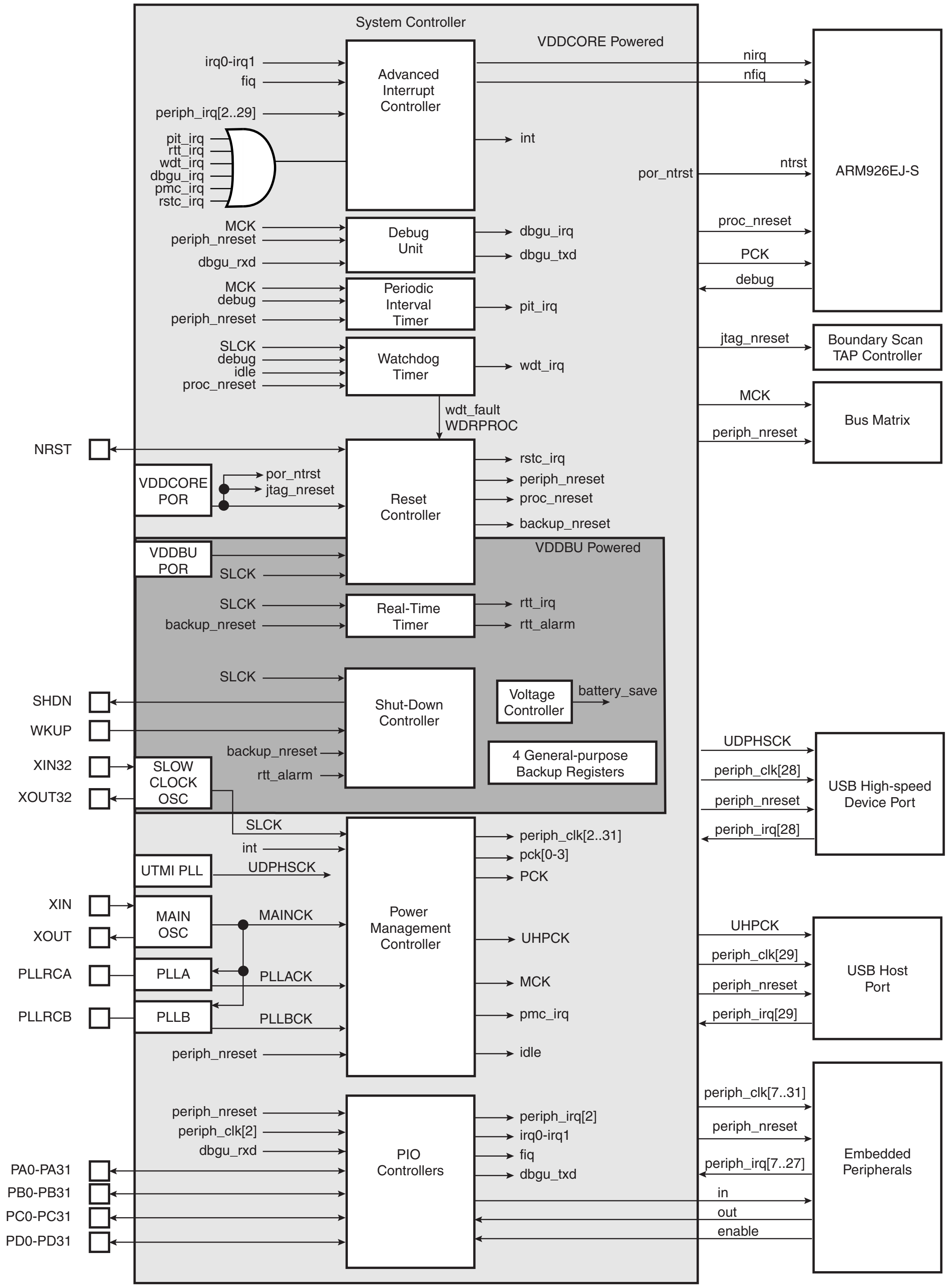
However, all the registers of System Controller are mapped on the top of the address space. This allows all the registers of the System Controller to be addressed from a single pointer by using the standard ARM instruction set, as the Load/Store instructions have an indexing mode of ± 4 Kbytes.

Figure 9-1 on page 33 shows the System Controller block diagram.

Figure 8-1 on page 26 shows the mapping of the User Interfaces of the System Controller peripherals.

9.1 System Controller Block Diagram

Figure 9-1. AT91CAP9S500A/AT91CAP9S250A System Controller Block Diagram



9.2 Reset Controller

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

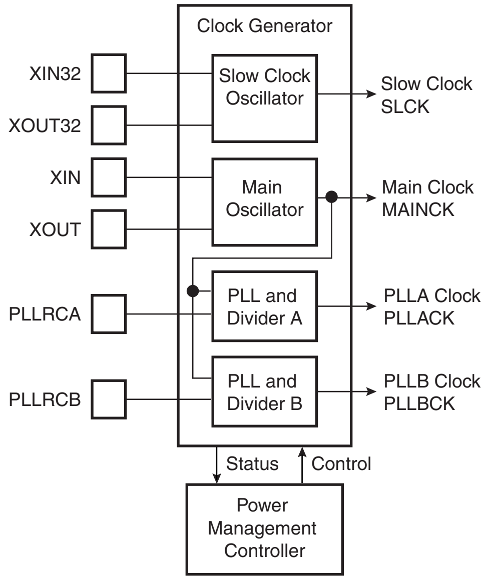
9.3 Shutdown Controller

- Shutdown and Wake-Up logic
  - Software programmable assertion of the SHDN pin
  - Deassertion Programmable on a WKUP pin level change or on alarm

9.4 Clock Generator

- Embeds a low power 32,768 Hz Slow Clock Oscillator and a low power RC oscillator
  - Provides the permanent Slow Clock SLCK to the system
- Embeds the Main Oscillator
  - Oscillator bypass feature
  - Supports 8 to 16 MHz crystals
  - 12 MHz crystal is required for USB High-Speed Device
- Embeds 2 programmable PLLs
  - Output 80 to 240 MHz clocks
  - Integrates an input divider to increase output accuracy
- Embeds 1 UTMI PLL
  - 480 MHz Fixed frequency from 12 MHz input clock
  - Integrated filter

Figure 9-2. Clock Generator Block Diagram



9.5 Slow Clock Selection

The AT91CAP9S500A/AT91CAP9S250A 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.

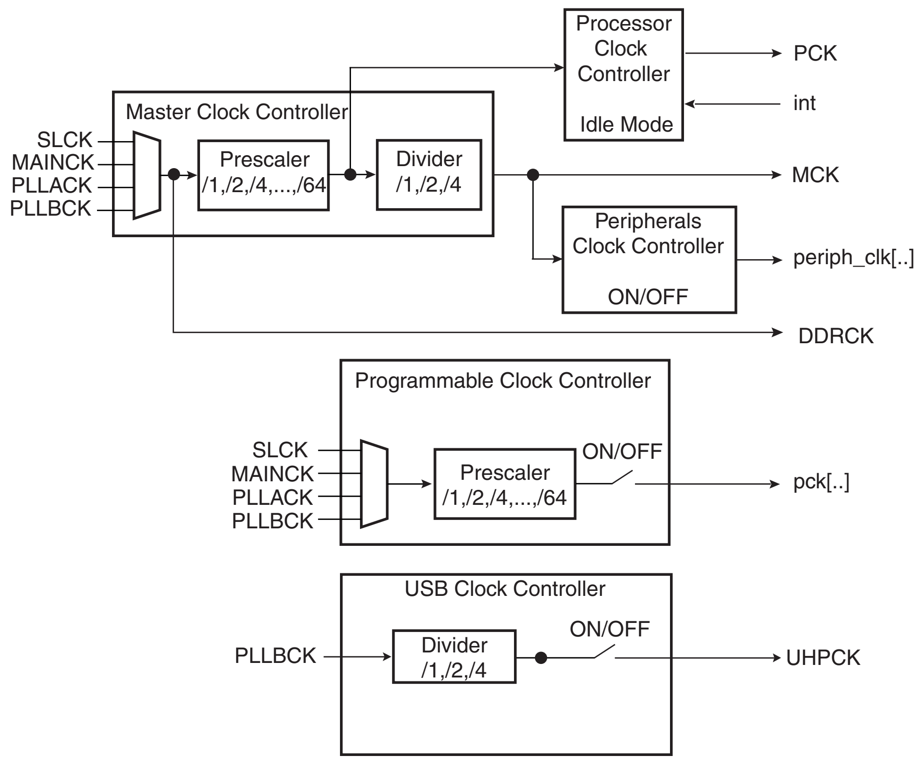
Configuration is located in the slow clock control register (SCKCR) located at address 0xFFFFFD50 in the backed up part of the system controller and so is preserved while VDDBU is present.

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

9.6 Power Management Controller

- Provides:
  - the Processor Clock PCK
  - the Master Clock MCK, in particular to the Matrix and the memory interfaces
  - the USB High-speed Device Clock UDPHSCK
  - the USB Host Clock UHPCK
  - independent peripheral clocks, typically at the frequency of MCK
  - four programmable clock outputs: PCK0 to PCK3
- Five flexible operating modes:
  - Normal Mode, processor and peripherals running at a programmable frequency
  - Idle Mode, processor stopped waiting for an interrupt
  - Slow Clock Mode, processor and peripherals running at low frequency
  - Standby Mode, mix of Idle and Backup Mode, peripheral running at low frequency, processor stopped waiting for an interrupt
  - Backup Mode, Main Power Supplies off, VDDBU powered by a battery

Figure 9-3. AT91CAP9S500A/AT91CAP9S250A Power Management Controller Block Diagram



9.7 Periodic Interval Timer

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

9.8 Watchdog Timer

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

9.9 Real-time Timer

- Two Real-time Timers, allowing backup of time with different accuracies
  - 32-bit Free-running back-up Counter
  - Integrates a 16-bit programmable prescaler running on the embedded 32,768 Hz oscillator
  - Alarm Register to generate a wake-up of the system through the Shutdown Controller

9.10 General-Purpose Backup Registers

- Four 32-bit backup general-purpose registers

9.11 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
- Four 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

9.12 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

9.13 Chip Identification

- Chip ID: 0x039A03A1 (for DevChip)
- JTAG ID: 0x15B1B03F
- ARM926 TAP ID: 0x0792603F

9.14 PIO Controllers

- 4 PIO Controllers, PIOA to PIOD, controlling a total of 128 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 32 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 User Interface

The peripherals are mapped in the upper 256 Mbytes of the address space between the addresses 0xFFFA 0000 and 0xFFFC FFFF. Each user peripheral is allocated 16 Kbytes of address space.

A complete memory map is presented in Figure 8-1 on page 26.

10.2 Identifiers

The AT91CAP9S500A/AT91CAP9S250A embeds a wide range of peripherals. Table 10-1 defines the Peripheral Identifiers of the AT91CAP9S500A/AT91CAP9S250A. 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. AT91CAP9S500A/AT91CAP9S250A Peripheral Identifiers

Peripheral ID	Peripheral Mnemonic	Peripheral Name	External Interrupt
0	AIC	Advanced Interrupt Controller	FIQ
1	SYSC	System Controller Interrupt	
2	PIOA-D	Parallel I/O Controller A to D	
3	MPB0	MP Block Peripheral 0	
4	MPB1	MP Block Peripheral 1	
5	MPB2	MP Block Peripheral 2	
6	MPB3	MP Block Peripheral 3	
7	MPB4	MP Block Peripheral 4	
8	US0	USART 0	
9	US1	USART 1	
10	US2	USART 2	
11	MCI0	Multimedia Card Interface 0	
12	MCI1	Multimedia Card Interface 1	
13	CAN	CAN Controller	
14	TWI	Two-Wire Interface	
15	SPI0	Serial Peripheral Interface 0	
16	SPI1	Serial Peripheral Interface 1	
17	SSC0	Synchronous Serial Controller 0	
18	SSC1	Synchronous Serial Controller 1	
19	AC97	AC97 Controller	
20	TC0, TC1, TC2	Timer/Counter 0, 1 and 2	
21	PWMC	Pulse Width Modulation Controller	
22	EMAC	Ethernet MAC	
23	Reserved	Reserved	

Table 10-1. AT91CAP9S500A/AT91CAP9S250A Peripheral Identifiers (Continued)

Peripheral ID	Peripheral Mnemonic	Peripheral Name	External Interrupt
24	ADCC	ADC Controller	
25	ISI	Image Sensor Interface	
26	LCDC	LCD Controller	
27	DMA	DMA Controller	
28	UDPHS	USB High Speed Device Port	
29	UHP	USB Host Port	
30	AIC	Advanced Interrupt Controller	IRQ0
31	AIC	Advanced Interrupt Controller	IRQ1

10.2.1 Peripheral Interrupts and Clock Control

10.2.1.1 System Interrupt

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

- the DDR/SDRAM Controller
- the BCRAM Controller
- the Debug Unit
- the Periodic Interval Timer
- the Real-Time Timer
- the Watchdog Timer
- the Reset Controller
- the Power Management Controller
- the MP Block

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

10.2.1.2 External Interrupts

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

10.2.1.3 Timer Counter Interrupts

The three Timer Counter channels interrupt signals are OR-wired together to provide the interrupt source 19 of the Advanced Interrupt Controller. This forces the programmer to read all Timer Counter status registers before branching the right Interrupt Service Routine.

The Timer Counter channels clocks cannot be deactivated independently. Switching off the clock of the Peripheral 19 disables the clock of the 3 channels.

10.2.2 DMA Controller Request Signals

The requests to the DMA Controller may come from eight different sources:

- four external requests
- four internal requests from the MPBlock

Table 10-2. DMA Controller Request Source and Signal Names

Internal DMA Request from MPBlock				External DMA Request			
Channel 7	Channel 6	Channel 5	Channel 4	Channel 3	Channel 2	Channel 1	Channel 0
MP_DMARQ3	MP_DMARQ2	MP_DMARQ1	MP_DMARQ0	DMARQ3	DMARQ2	DMARQ1	DMARQ0

Each request source is selected through the DMAC Channel x Configuration Register.

It is also necessary to choose the hardware handshaking interface from the SRC_H2SEL and DST_H2SEL fields.

(For more details, see the DMA Controller (DMAC) section and DMAC User Interface in the product datasheet.)

10.3 Peripheral Signal Multiplexing on I/O Lines

The AT91CAP9S500A/AT91CAP9S250A features 4 PIO controllers, PIOA, PIOB, PIOC and PIOD, that multiplex 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 may be duplicated within 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 of pins controlling memories, in particular the address lines, which require the pin to be driven as soon as the reset is released. Note that the pull-up resistor is also enabled in this case.

10.3.1 PIO Controller A Multiplexing

Table 10-3. Multiplexing on PIO Controller A

PIO Controller A					Application Usage		
I/O Line	Peripheral A	Peripheral B	Comments	Reset State	Power Supply	Function	324-BGA pkg Options[1]
PA0	MCI0_D0	SPI0_MISO		I/O	VDDIOP0		
PA1	MCI0_CD	SPI0_MOSI		I/O	VDDIOP0		
PA2	MCI0_CK	SPI0_SPCK		I/O	VDDIOP0		
PA3	MCI0_D1	SPI0_NPCS1		I/O	VDDIOP0		
PA4	MCI0_D2	SPI0_NPCS2		I/O	VDDIOP0		
PA5	MCI0_D3	SPI0_NPCS0		I/O	VDDIOP0		
PA6	AC97FS			I/O	VDDIOP0		
PA7	AC97CK			I/O	VDDIOP0		
PA8	AC97TX			I/O	VDDIOP0		
PA9	AC97RX			I/O	VDDIOP0		
PA10	IRQ0	PWM1		I/O	VDDIOP0		
PA11	DMARQ0	PWM3		I/O	VDDIOP0		
PA12	CANTX	PCK0		I/O	VDDIOP0		
PA13	CANRX			I/O	VDDIOP0		
PA14	TCLK2	IRQ1		I/O	VDDIOP0		
PA15	DMARQ3	PCK2		I/O	VDDIOP0		
PA16	MCI1_CK	ISI_D0		I/O	VDDIOP1		can be removed
PA17	MCI1_CD	ISI_D1		I/O	VDDIOP1		can be removed
PA18	MCI1_D0	ISI_D2		I/O	VDDIOP1		can be removed
PA19	MCI1_D1	ISI_D3		I/O	VDDIOP1		can be removed
PA20	MCI1_D2	ISI_D4		I/O	VDDIOP1		can be removed
PA21	MCI1_D3	ISI_D5		I/O	VDDIOP1		can be removed
PA22	TXD0	ISI_D6		I/O	VDDIOP1		can be removed
PA23	RXD0	ISI_D7		I/O	VDDIOP1		can be removed
PA24	RTS0	ISI_PCK		I/O	VDDIOP1		can be removed
PA25	CTS0	ISI_HSYNC		I/O	VDDIOP1		can be removed
PA26	SCK0	ISI_VSYNC		I/O	VDDIOP1		can be removed
PA27	PCK1	ISI_MCK		I/O	VDDIOP1		can be removed
PA28	SPI0_NPCS3	ISI_D8		I/O	VDDIOP1		can be removed
PA29	TIOA0	ISI_D9		I/O	VDDIOP1		can be removed
PA30	TIOB0	ISI_D10		I/O	VDDIOP1		can be removed
PA31	DMARQ1	ISI_D11		I/O	VDDIOP1		can be removed

Note: 1. The user must ensure that removing the designated pins does not have an adverse effect on the intended application.

10.3.2 PIO Controller B Multiplexing

Table 10-4. Multiplexing on PIO Controller B

PIO Controller B					Application Usage		
I/O Line	**Peripheral A**	**Peripheral B**	**Comments**	**Reset State**	**Power Supply**	**Function**	**324-BGA pkg Options**[(1)]
PB0	TF0			I/O	VDDIOP0		
PB1	TK0			I/O	VDDIOP0		
PB2	TD0			I/O	VDDIOP0		
PB3	RD0			I/O	VDDIOP0		
PB4	RK0	TWD		I/O	VDDIOP0		
PB5	RF0	TWCK		I/O	VDDIOP0		
PB6	TF1	TIOA1		I/O	VDDIOP0		
PB7	TK1	TIOB1		I/O	VDDIOP0		
PB8	TD1	PWM2		I/O	VDDIOP0		
PB9	RD1	LCDCC		I/O	VDDIOP0		
PB10	RK1	PCK1		I/O	VDDIOP0		
PB11	RF1			I/O	VDDIOP0		
PB12	SPI1_MISO			I/O	VDDIOP0		
PB13	SPI1_MOSI		AD0	I/O	VDDIOP0		
PB14	SPI1_SPCK		AD1	I/O	VDDIOP0		
PB15	SPI1_NPCS0		AD2	I/O	VDDIOP0		
PB16	SPI1_NPCS1		AD3	I/O	VDDIOP0		
PB17	SPI1_NPCS2		AD4	I/O	VDDIOP0		
PB18	SPI1_NPCS3		AD5	I/O	VDDIOP0		
PB19	PWM0		AD6	I/O	VDDIOP0		
PB20	PWM1		AD7	I/O	VDDIOP0		
PB21	ETXCK/EREFCK	TIOA2		I/O	VDDIOP0		can be removed
PB22	ERXDV	TIOB2		I/O	VDDIOP0		can be removed
PB23	ETX0	PCK3		I/O	VDDIOP0		can be removed
PB24	ETX1			I/O	VDDIOP0		can be removed
PB25	ERX0			I/O	VDDIOP0		can be removed
PB26	ERX1			I/O	VDDIOP0		can be removed
PB27	ERXER			I/O	VDDIOP0		can be removed
PB28	ETXEN	TCLK0		I/O	VDDIOP0		can be removed
PB29	EMDC	PWM3		I/O	VDDIOP0		can be removed
PB30	EMDIO			I/O	VDDIOP0		can be removed
PB31	ADTRIG	EF100		I/O	VDDIOP0		can be removed

Note: 1. The user must ensure that removing the designated pins does not have an adverse effect on the intended application.

10.3.3 PIO Controller C Multiplexing

Table 10-5. Multiplexing on PIO Controller C

PIO Controller C					Application Usage		
I/O Line	**Peripheral A**	**Peripheral B**	**Comments**	**Reset State**	**Power Supply**	**Function**	**324-BGA pkg Options**[1]
PC0	LCDVSYNC			I/O	VDDIOP0		can be removed
PC1	LCDHSYNC			I/O	VDDIOP0		can be removed
PC2	LCDDOTCK			I/O	VDDIOP0		can be removed
PC3	LCDDEN	PWM1		I/O	VDDIOP0		can be removed
PC4	LCDD0	LCDD3		I/O	VDDIOP0		can be removed
PC5	LCDD1	LCDD4		I/O	VDDIOP0		can be removed
PC6	LCDD2	LCDD5		I/O	VDDIOP0		can be removed
PC7	LCDD3	LCDD6		I/O	VDDIOP0		can be removed
PC8	LCDD4	LCDD7		I/O	VDDIOP0		can be removed
PC9	LCDD5	LCDD10		I/O	VDDIOP0		can be removed
PC10	LCDD6	LCDD11		I/O	VDDIOP0		can be removed
PC11	LCDD7	LCDD12		I/O	VDDIOP0		can be removed
PC12	LCDD8	LCDD13		I/O	VDDIOP0		can be removed
PC13	LCDD9	LCDD14		I/O	VDDIOP0		can be removed
PC14	LCDD10	LCDD15		I/O	VDDIOP0		can be removed
PC15	LCDD11	LCDD19		I/O	VDDIOP0		can be removed
PC16	LCDD12	LCDD20		I/O	VDDIOP0		can be removed
PC17	LCDD13	LCDD21		I/O	VDDIOP0		can be removed
PC18	LCDD14	LCDD22		I/O	VDDIOP0		can be removed
PC19	LCDD15	LCDD23		I/O	VDDIOP0		can be removed
PC20	LCDD16	ETX2		I/O	VDDIOP0		can be removed
PC21	LCDD17	ETX3		I/O	VDDIOP0		can be removed
PC22	LCDD18	ERX2		I/O	VDDIOP0		can be removed
PC23	LCDD19	ERX3		I/O	VDDIOP0		can be removed
PC24	LCDD20	ETXER		I/O	VDDIOP0		can be removed
PC25	LCDD21	ECRS		I/O	VDDIOP0		can be removed
PC26	LCDD22	ECOL		I/O	VDDIOP0		can be removed
PC27	LCDD23	ERXCK		I/O	VDDIOP0		can be removed
PC28	PWM0	TCLK1		I/O	VDDIOP0		
PC29	PCK0	PWM2		I/O	VDDIOP0		
PC30	DRXD			I/O	VDDIOP0		
PC31	DTXD			I/O	VDDIOP0		

Note: 1. The user must ensure that removing the designated pins does not have an adverse effect on the intended application.

10.3.4 PIO Controller D Multiplexing

Table 10-6. Multiplexing on PIO Controller D

PIO Controller D					Application Usage		
I/O Line	Peripheral A	Peripheral B	Comments	Reset State	Power Supply	Function	324-BGA pkg Options[(1)]
PD0	TXD1	SPI0_NPCS2		I/O	VDDIOP0		can be removed
PD1	RXD1	SPI0_NPCS3		I/O	VDDIOP0		can be removed
PD2	TXD2	SPI1_NPCS2		I/O	VDDIOP0		can be removed
PD3	RXD2	SPI1_NPCS3		I/O	VDDIOP0		can be removed
PD4	FIQ			I/O	VDDIOP0		can be removed
PD5	DMARQ2	RTS2		I/O	VDDIOP0		can be removed
PD6	NWAIT	CTS2		I/O	VDDIOM		can be removed
PD7	NCS4/CFCS0	RTS1		I/O	VDDIOM		can be removed
PD8	NCS5/CFCS1	CTS1		I/O	VDDIOM		can be removed
PD9	CFCE1	SCK2		I/O	VDDIOM		can be removed
PD10	CFCE2	SCK1		I/O	VDDIOM		can be removed
PD11	NCS2			I/O	VDDIOM		
PD12	A23			A23	VDDIOM		
PD13	A24			A24	VDDIOM		
PD14	A25/CFRNW			A25	VDDIOM		
PD15	NCS3/NANDCS			I/O	VDDIOM		
PD16	D16			I/O	VDDIOM		
PD17	D17			I/O	VDDIOM		
PD18	D18			I/O	VDDIOM		
PD19	D19			I/O	VDDIOM		
PD20	D20			I/O	VDDIOM		
PD21	D21			I/O	VDDIOM		
PD22	D22			I/O	VDDIOM		
PD23	D23			I/O	VDDIOM		
PD24	D24			I/O	VDDIOM		
PD25	D25			I/O	VDDIOM		
PD26	D26			I/O	VDDIOM		
PD27	D27			I/O	VDDIOM		
PD28	D28			I/O	VDDIOM		
PD29	D29			I/O	VDDIOM		
PD30	D30			I/O	VDDIOM		
PD31	D31			I/O	VDDIOM		

Note: 1. The user must ensure that removing the designated pins does not have an adverse effect on the intended application.

10.4 Embedded Peripherals

10.4.1 Serial Peripheral Interface

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

10.4.2 Two-wire Interface

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

10.4.3 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

10.4.4 Synchronous Serial Controller

- Provides serial synchronous communication links used in audio and telecom applications (with CODECs in Master or Slave Modes, I^2S, 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

10.4.5 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)

10.4.6 Timer Counter

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

10.4.7 Pulse Width Modulation Controller

- 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 Buffering
  - Programmable selection of the output waveform polarity
  - Programmable center or left aligned output waveform

10.4.8 Multimedia Card Interface

- 2 double-channel Multimedia Card Interface, allowing concurrent transfers with 2 cards
- 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
- Each 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

10.4.9 CAN Controller

- Fully compliant with 16-mailbox CAN 2.0A and 2.0B CAN Controllers
- Bit rates up to 1Mbit/s.
- Object-oriented mailboxes, each with the following properties:
  - CAN Specification 2.0 Part A or 2.0 Part B Programmable for Each Message
  - Object Configurable as receive (with overwrite or not) or transmit
  - Local Tag and Mask Filters up to 29-bit Identifier/Channel
  - 32 bits access to Data registers for each mailbox data object
  - Uses a 16-bit time stamp on receive and transmit message
  - Hardware concatenation of ID unmasked bitfields to speedup family ID processing
  - 16-bit internal timer for Time Stamping and Network synchronization
  - Programmable reception buffer length up to 16 mailbox object
  - Priority Management between transmission mailboxes
  - Autobaud and listening mode
  - Low power mode and programmable wake-up on bus activity or by the application
  - Data, Remote, Error and Overload Frame handling

10.4.10 USB Host Port

- Compliance with OHCI Rev 1.0 Specification
- Compliance with USB V2.0 Full-speed and Low-speed Specification
- Supports both Low-speed 1.5 Mbps and Full-speed 12 Mbps devices
- Root hub integrated with two downstream USB ports
- Two embedded USB transceivers
- Supports power management
- Operates as a master on the Matrix
- Internal DMA Controller, operating as a Master on Bus Matrix

10.4.11 USB High Speed Device Port

- 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 2 or 3 banks for isochronous and bulk endpoints
- Seven endpoints:
  - Endpoint 0: 64 bytes
  - Endpoint 1 & 2: 1024 bytes, 3 banks mode, HS isochronous capable
  - Endpoint 3 & 4: 1024 bytes, 2 banks mode, HS isochronous capable
  - Endpoint 5 & 6: 1024 bytes, 2 banks mode
  - Endpoint 7: 1024 bytes, 2 banks mode

10.4.12 LCD Controller

- 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 2048x2048
- 2D-DMA Controller for management of virtual Frame Buffer
  - Allows management of frame buffer larger than the screen size and moving the view over this virtual frame buffer
- Automatic resynchronization of the frame buffer pointer to prevent flickering

10.4.13 Ethernet 10/100 MAC

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

10.4.14 Image Sensor Interface

- ITU-R BT. 601/656 8-bit mode external interface support
- Support for ITU-R BT.656-4 SAV and EAV synchronization
- Vertical and horizontal resolutions up to 2048 x 2048
- Preview Path up to 640*480
- Support for packed data formatting for YCbCr 4:2:2 formats
- Preview scaler to generate smaller size image
- Programmable frame capture rate
- Internal DMA Controller, operating as a Master on Bus Matrix

10.4.15 Analog-to-digital Converter

- 8-channel ADC
- 10-bit 440K samples/sec. Successive Approximation Register ADC
- -2/+2 LSB Integral Non Linearity, -1/+1 LSB Differential Non Linearity
- Individual enable and disable of each channel
- External voltage reference for better accuracy on low voltage inputs
- Multiple trigger source – Hardware or software trigger – External trigger pin – Timer Counter 0 to 2 outputs TIOA0 to TIOA2 and TIOB0 to TIOB2 triggers
- Sleep Mode and conversion sequencer – Automatic wakeup on trigger and back to sleep mode after conversions of all enabled channels
- Four analog inputs shared with digital signals

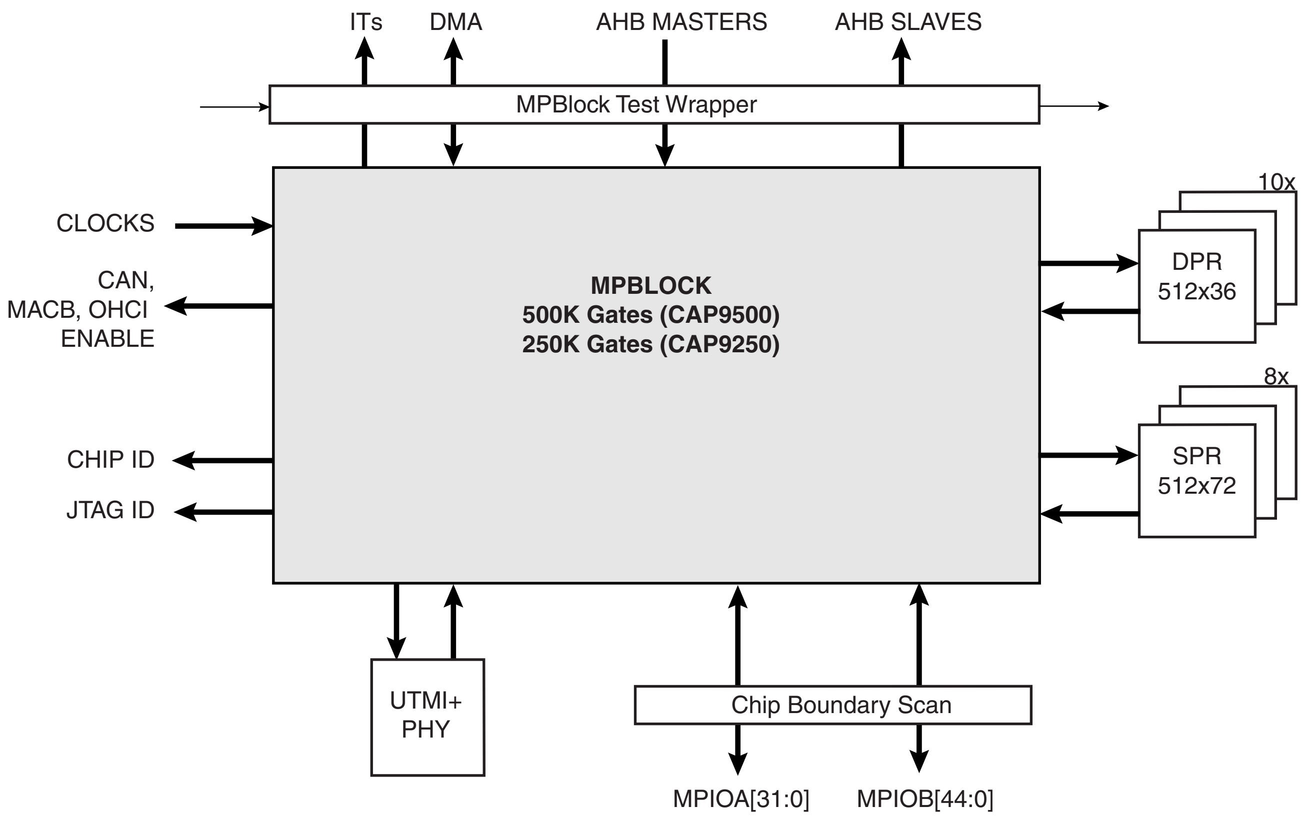
11. Metal Programmable Block

The Metal Programmable Block (MPBlock) is connected to internal resources as the AHB bus or interrupts and to external resources as dedicated I/O pads or UTMI+ core.

The MPBlock may be used to implement the Advanced High-speed Bus (AHB) or Advanced Peripheral Bus (APB) custom peripherals. The MPBlock adds approximately 500K or 250K gates of standard cell custom logic to the AT91CAP9S500A/AT91CAP9S250A base design.

Figure 11-1 shows the MPBlock and its connections to internal or external resources.

Figure 11-1. MPBlock Connectivity



11.1 Internal Connectivity

In order to connect the MPBlock custom peripheral to the AT91CAP9S500A/AT91CAP9S250A base design, the following connections are made.

11.1.1 Clocks

The MPBlock receives the following clocks:

- 32,768 Hz Slow Clock
- 8 to 16 MHz Main Oscillator Clock
- PLLA Clock
- PLLB Clock
- 48 MHz USB Clock
- 12 MHz USB Clock

- 30 or 60 MHz UTMI+ USB Clock
- MCK System Clock
- DDRCK Dual Rate System Clock
- PCK Processor Clock
- 5 Gated Peripheral Clocks (for AHB and/or APB peripherals) corresponding to Peripheral ID 3 to 7

11.1.2 AHB Master Buses

The MPBlock may implement up to three AHB masters, each having a dedicated AHB master bus connected to the Bus Matrix.

11.1.3 AHB Slave Buses

The MPBlock receives four different AHB slave buses coming from the Bus Matrix. Each bus has two or four select signals that can implement up to 12 AHB slaves.

11.1.4 Interrupts

The MPBlock is connected to 5 dedicated interrupt lines corresponding to Peripheral ID 3 to 9.

It is also connected to two other interrupt lines (through OR gate) corresponding to Peripheral ID 1 and 2

11.1.5 DMA Channels

The MPBlock is connected to 4 DMA hardware handshaking interfaces, allowing it to implement up to 4 DMA enabled peripherals.

11.1.6 Peripheral DMA Channels

The MPBlock is not connected to the Peripheral DMA Controller. In order to implement Peripheral DMA Controller (PDC) enabled APB peripherals, a PDC and an AHB-to-APB Bridge must be integrated into the MPBlock using one AHB master and one AHB slave bus.

11.1.7 MPBlock Single Port RAMs

The MPBlock is connected to eight instances of 512x72 High-Speed Single Port RAMs.

The MPBlock has control over all memory connections.

11.1.8 MPBlock Dual Port RAMs

The MPBlock is connected to ten instances of 512x36 High-Speed Dual Port RAMs.

The MPBlock has control over all memory connections.

11.1.9 Optional Peripherals Enable

The MPBlock drives the enable of the optional peripherals, and so can enable or disable any of the optional peripherals.

11.2 External Connectivity

The MPBlock is connected to the following external resources.

11.2.1 Dedicated I/O Lines

The MPBlock is directly connected to 77 (32 MPIOA and 45 MPIOB lines) dedicated I/O Pads with the following features:

- Supply/Drive control pin (needed for high-speed or low voltage interfaces)
- Pull-up control pin
- Supported logic levels include:
  - LVCMOS33 at 100 MHz maximum frequency
  - LVCMOS25 at 50 MHz maximum frequency
  - LVCMOS18 at 100 MHz maximum frequency

11.2.2 UTMI+ Transceiver

The MPBlock may be connected to the UTMI+ transceiver. As only one UTMI+ transceiver is available, the USB High-speed Device and the MPBlock do not have access to the UTMI+ at the same time. However, a dual role Master-Slave USB High-Speed may be implemented by using the USB High-speed Device and integrating a High-speed Host in the MPBlock as the switching between both is generated inside the MPBlock.

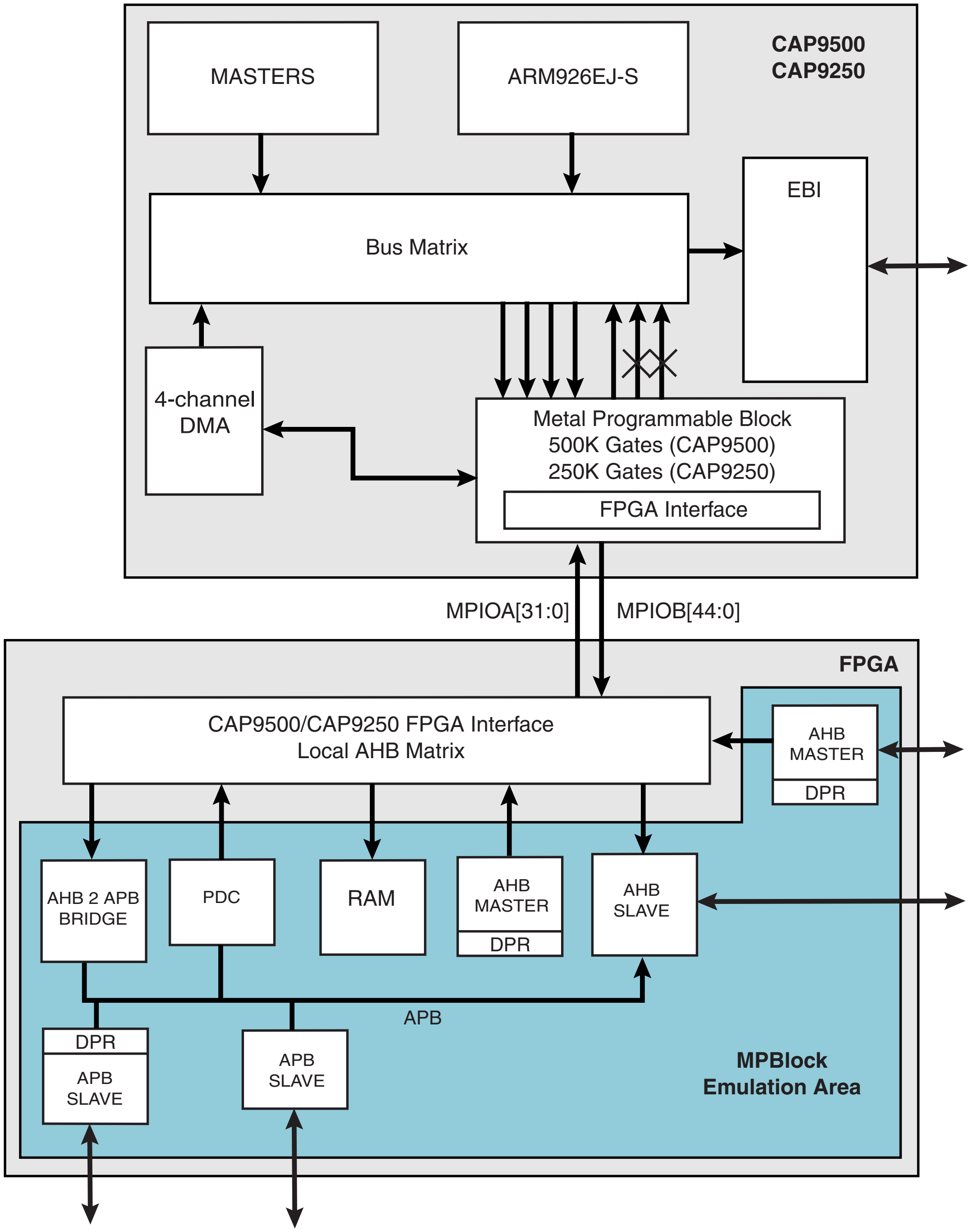
11.3 Prototyping Solution

In order to prototype the final custom design, a Prototyping Platform version of the AT91CAP9S500A/AT91CAP9S250A design has been created. The platform maps APB and AHB masters or slaves into the FPGA located outside the chip with the following features and restrictions:

- AT91CAP9S500A/AT91CAP9S250A to FPGA interface is provided to prototype AHB masters and slave into the external FPGA exactly as if it were in MPBlock.
- Prototyped AHB Masters
  - Prototyped AHB Masters have access to AT91CAP9S500A/AT91CAP9S250A slave resources.
  - Prototyped AHB Masters have access to MPBlock (FPGA) slave resources.
- Prototyped AHB Slaves
  - Prototyped AHB Slaves may be accessed from AT91CAP9S500A/AT91CAP9S250A master resources.
  - Prototyped AHB Slaves may be accessed from MPBlock (FPGA) resources.
- Prototyped APB Slaves
  - APB bus must be created locally in the FPGA by implementing AHB to APB bridge. Peripheral DMA controller may also be necessary to implement locally in the FPGA in order to prototype PDC enabled APB peripherals.

Figure 11-2 shows a typical prototyping solution.

Figure 11-2. Typical Prototyping Solution



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.

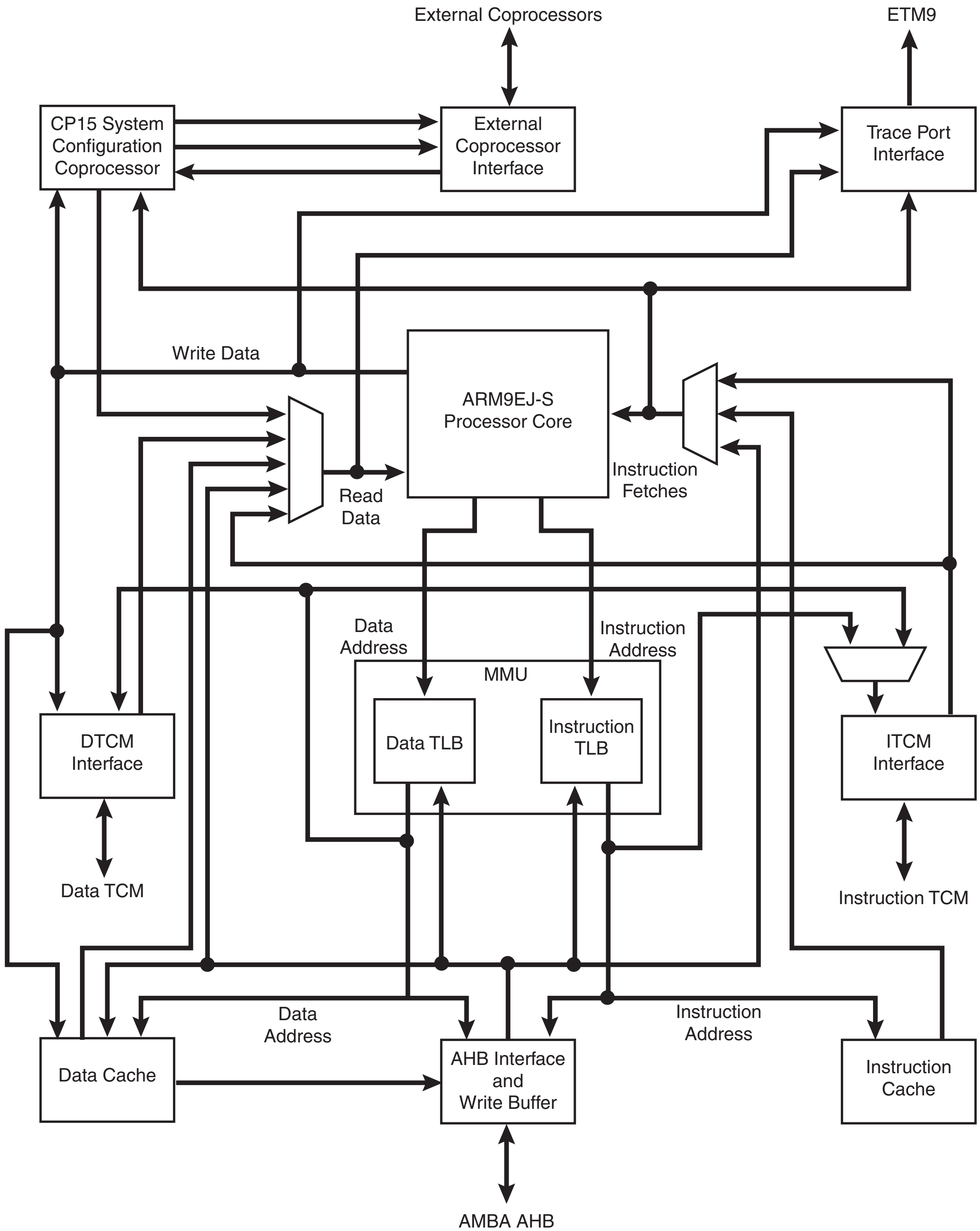
The ARM926EJ-S processor supports the ARM debug architecture and includes logic to assist in both hardware and software debug.

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

Table 12-1. ARM9TDMI Modes and Registers Layout

User and System Mode	Supervisor Mode	Abort Mode	Undefined Mode	Interrupt Mode	Fast Interrupt Mode
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 (shaded)

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

For more details, refer to ARM Software Development Kit.

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, 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

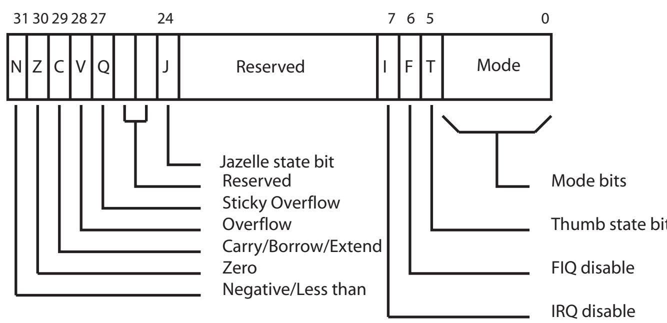


Figure 12-2 shows the status register format, where:

- 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.

Note that there is one exception in the priority scheme: 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]).

For further details, see the ARM Technical Reference Manual.

Table 12-2 gives the ARM instruction mnemonic list.

Table 12-2. ARM Instruction Mnemonic List

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

Table 12-2. ARM Instruction Mnemonic List (Continued)

Mnemonic	Operation	Mnemonic	Operation
LDRH	Load Half Word	STRH	Store Half Word
LDRB	Load Byte	STRB	Store Byte
LDRBT	Load Register Byte with Translation	STRBT	Store Register Byte with Translation
LDRT	Load Register with Translation	STRT	Store Register with Translation
LDM	Load Multiple	STM	Store Multiple
SWP	Swap Word	SWPB	Swap Byte
MCR	Move To Coprocessor	MRC	Move From Coprocessor
LDC	Load To Coprocessor	STC	Store From Coprocessor
CDP	Coprocessor Data Processing		

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 [(1)]	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

For further details, see the ARM Technical Reference Manual.

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
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	Access
0	ID Code[(1)]	Read/Unpredictable
0	Cache type[(1)]	Read/Unpredictable
0	TCM status[(1)]	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[(1)]	Read-write
5	Instruction fault status[(1)]	Read-write
6	Fault Address	Read-write
7	Cache Operations	Read-write
8	TLB operations	Unpredictable/Write
9	cache lockdown[(2)]	Read-write
9	TCM region	Read-write
10	TLB lockdown	Read-write
11	Reserved	None
12	Reserved	None
13	FCSE PID[(1)]	Read-write
13	Context ID[(1)]	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.

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.

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.

12.6 Caches and Write Buffer

The ARM926EJ-S contains a 16KB Instruction Cache (ICache), a 16KB 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).

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 line fills 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).

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.

12.6.2.3 Write-though 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.

12.6.2.4 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 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.

12.7.1 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: • data write (NCNB, NCB, WT, or WB that has missed in DCache) • data read (NCNB or NCB) • NC instruction fetch (prefetched and non-prefetched) • page table walk read
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

12.7.2 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.

12.7.3 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. Debug and Test

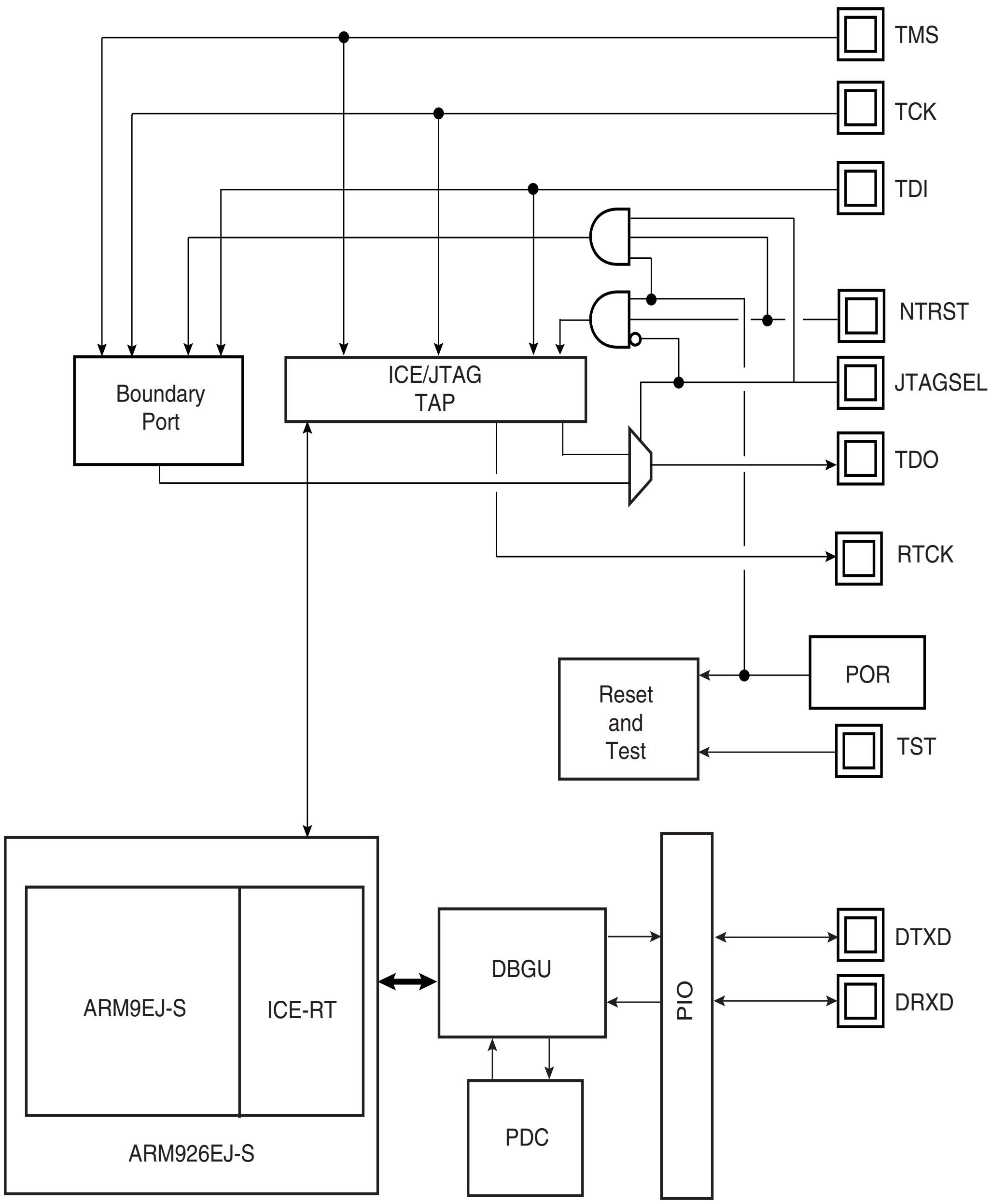
13.1 Description

The AT91CAP9 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



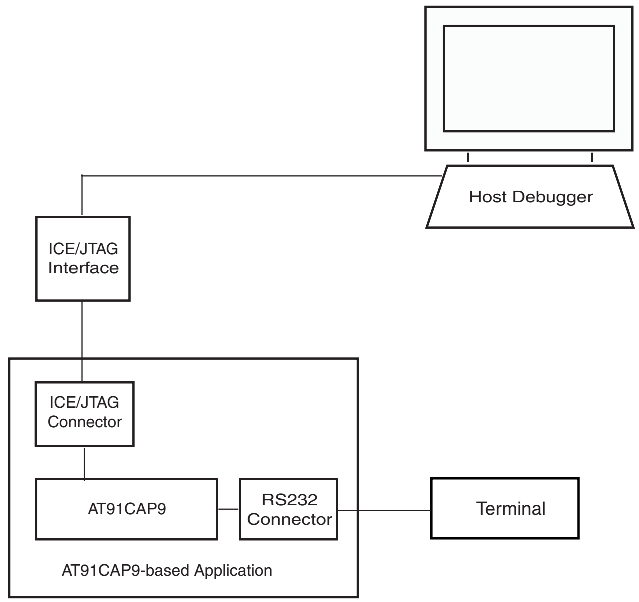
TAP: Test Access Port

13.3 Application Examples

13.3.1 Debug Environment

Figure 13-2 on page 75 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.

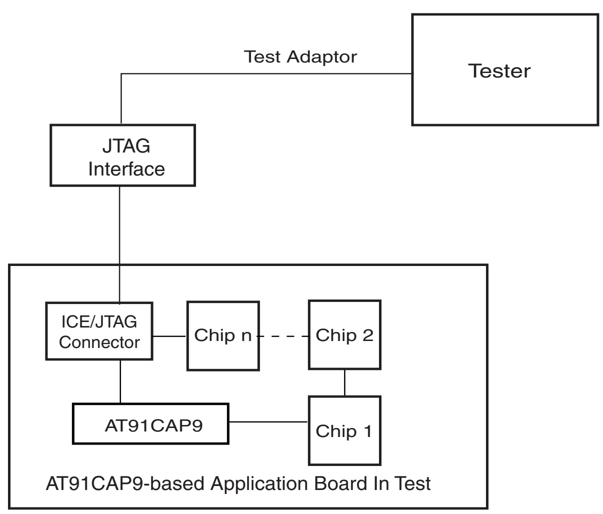
Figure 13-2. Application Debug and Trace Environment Example



13.3.2 Test Environment

Figure 13-3 on page 75 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
Reset/Test			
NRST	Microcontroller Reset	Input/Output	Low
TST	Test Mode Select	Input	High
ICE and JTAG			
TCK	Test Clock	Input	
TDI	Test Data In	Input	
TDO	Test Data Out	Output	
TMS	Test Mode Select	Input	
RTCK	Returned Test Clock	Output	
NTRST	Test Reset	Input	Low
JTAGSEL	JTAG Selection	Input	
Debug Unit			
DRXD	Debug Receive Data	Input	
DTXD	Debug Transmit Data	Output	

13.5 Functional Description

13.5.1 Test Pin

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

13.5.2 Embedded In-circuit Emulator

The 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 EmbeddedICE-RT™. The scan chains are controlled by the ICE/JTAG port.

EmbeddedICE 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 ignore 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 AT91CAP9 Debug Unit Chip ID value is 0x039A 03A0 on 32-bit width.

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

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 0x05B1_B03F.

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

Product part Number is 0x5B1B

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

Set to 0x0.

14. Boot Program

14.1 Description

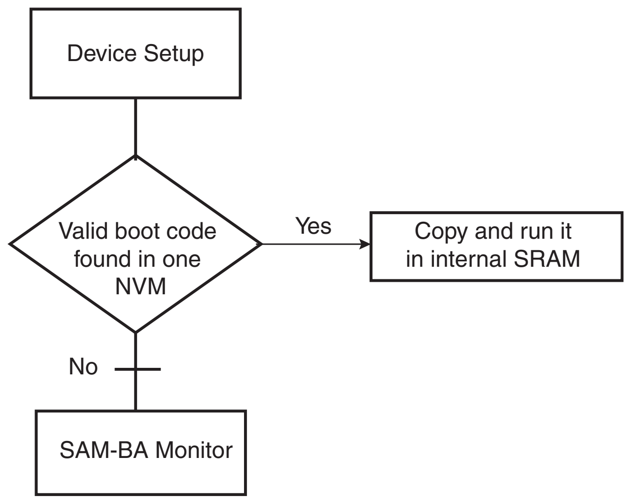
The Boot Program is contained in the embedded ROM. It is also called “Rom Code” or “First Level Bootloader”. At power on, if BMS is detected at 1, the boot memory is the embedded ROM and the Boot Program is executed. (see Section 8.1.2 “Boot Strategies”).

The Boot Program consists of several steps. First, it performs device initialization. Then it attempts to boot from external non-volatile memories (NVM). And finally, if no valid program has been found in NVM, it executes a monitor called SAM-BA® monitor.

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

14.3.1 Clock at Startup

At boot startup, the processor clock (PCK) and the master clock (MCK) is the slow clock. The slow clock can be an external 32 kHz crystal oscillator or the internal RC oscillator. By default the slow clock is the internal RC oscillator. If a battery supplies the backup power and if the external 32 kHz crystal oscillator was previously started up and selected, the slow clock at boot is the external 32 kHz crystal oscillator (see Section 29.2 “Slow Clock Crystal Oscillator”).

14.3.2 Initialization Sequence

Initialization follows the steps described below:

1. Stack setup for ARM supervisor mode.

2. **Main Oscillator Detection:** (external crystal or external clock on XIN). The Main Oscillator is disabled at startup (MOSCEN=1). First it is bypassed (OSCBYPASS set at 1). Then polling is done on the MAINRDY bit. As soon as this bit is raised, the Main Clock Frequency field is analyzed (MAINF). If the value exceeds 16, an external clock connected on XIN is detected. If not, an external crystal oscillator connected between XIN and XOUT (whose frequency is unknown at this moment) is detected.

3. **Main Oscillator Enabling**: If an external clock is connected on XIN, the Main Oscillator does not need to be started. Otherwise, If an external clock is not connected on XIN, the OSCBYPASS bit is cleared. The Main Oscillator is enabled with the maximum start-up time and a polling is done on the MOSC bit to wait for stabilization.

4. **Main Oscillator Selection**: The Master Clock source is switched from Slow Clock to the Main Oscillator without prescaler. Polling is done to wait for enabling. PCK and MCK are now the Main Oscillator clock.

5. **C variable initialization:** Non zero-initialized data are initialized in RAM (copy from ROM to RAM). Zero-initialized data are set to 0 in RAM.

6. **PLLA initialization:** PLLA is configured to allow communication on the USB link for the SAM-BA monitor. Its configuration depends on the Main Oscillator source (external clock or crystal) and on its frequency.

14.4 NVM Boot

14.4.1 NVM Bootloader Program Description

Figure 14-2. NVM Bootloader Program

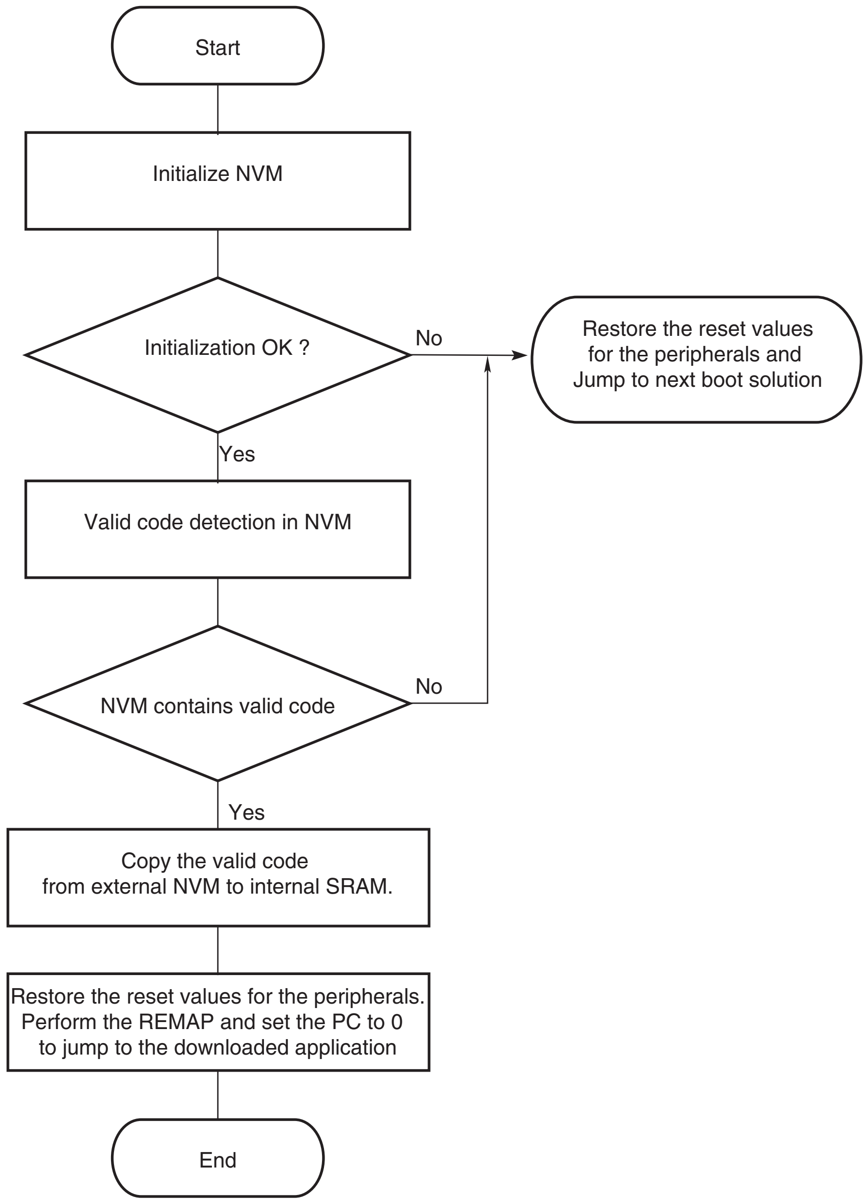
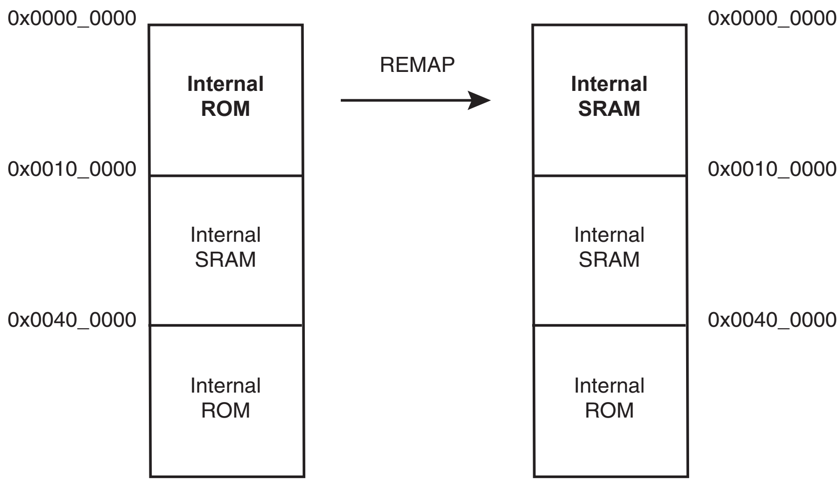


Figure 14-3. Remap Action after Download Completion



The NVM bootloader program initializes the NVM. It initializes the required PIO. It sets the right peripheral, depending on the NVM and tries to access the memory. If the initialization fails, it restores the reset values for the PIO and peripherals, then the next NVM bootloader program is executed.

If the initialization is successful, the NVM bootloader program reads the beginning of the NVM and determines if the NVM contains a valid code.

If the NVM does not contain a valid code, the NVM bootloader program restores the reset value for the peripherals and then the next NVM bootloader program is executed.

If a valid code is found, this code is loaded from NVM into internal SRAM and executed by branching at address 0x0000_0000 after remap. This code may be the application code or a second-level bootloader. All the calls to functions are PC relative and do not use absolute addresses.

14.4.2 Valid Code Detection

There are two kinds of valid code detection. Depending on the NVM bootloader, one or both of them are used.

14.4.2.1 ARM Exception Vectors Check

The NVM bootloader program reads and analyzes the first 28 bytes corresponding to the first seven ARM exception vectors. Except for the sixth vector, these bytes must implement the ARM instructions for either branch or load the PC with PC relative addressing.

Figure 14-4. LDR Opcode

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

Figure 14-5. 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 the PC with PC relative addressing instruction:

– Rn = Rd = PC = 0xF

– I==0 (12-bit immediate value)

– P==1 (pre-indexed)

– U offset added (U==1) or subtracted (U==0)

– W==1

The sixth vector, at offset 0x14, contains the size of the image to download. This vector must be replaced by the user's own vector. This information is described below.

Figure 14-6. Structure of the ARM Vector 6

31 ... 0
Size of the code to download in bytes

The value has to be less than 28 Kbytes. 28 Kbytes is the maximum size for a valid code. This size is the internal SRAM size minus the stack size used by the ROM Code at the end of the internal SRAM.

Example:

An example of valid vectors follows:

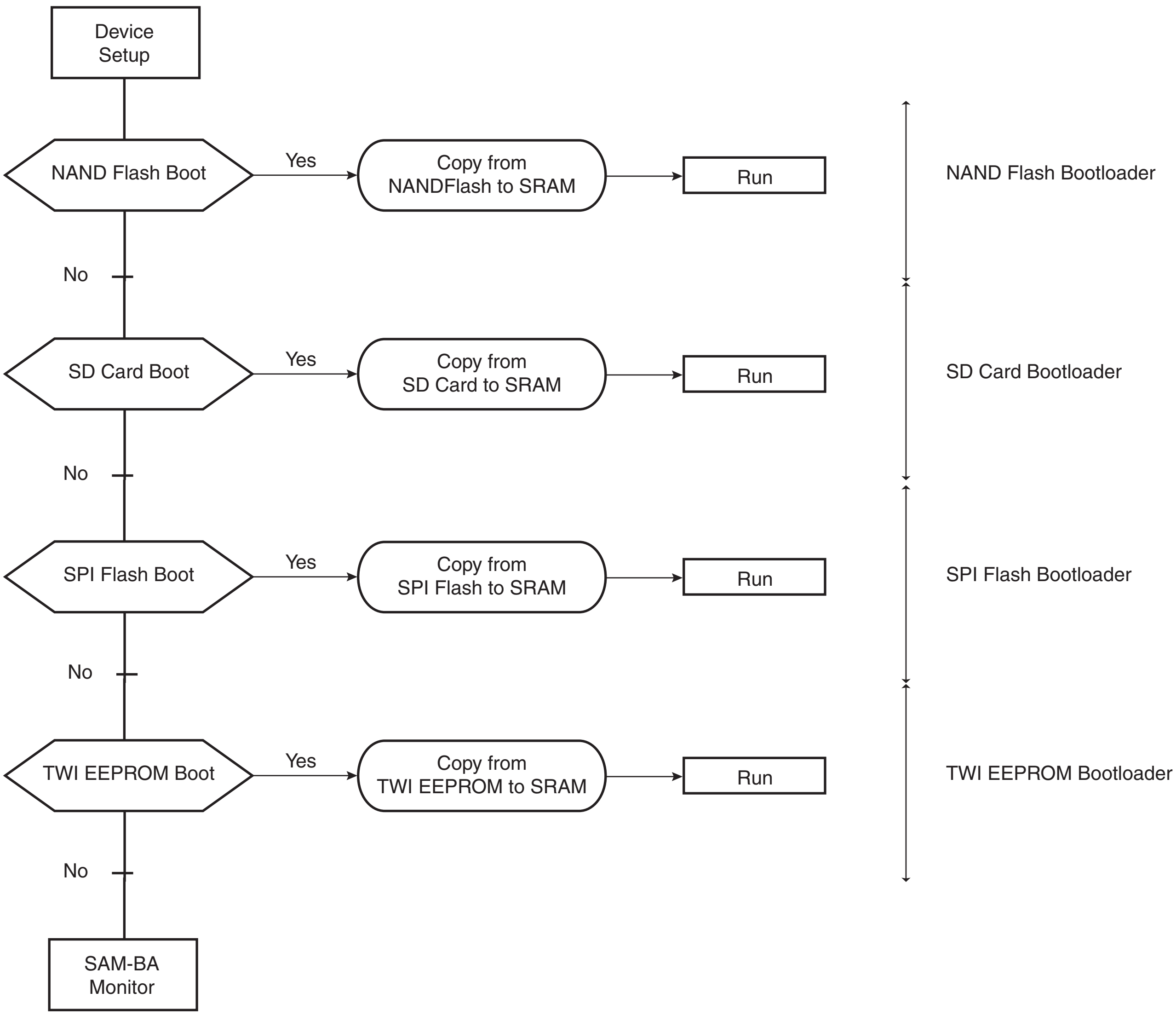
```
00    ea000006   B      0x20
04    eafffffe   B      0x04
08    ea00002f   B      _main
0c    eafffffe   B      0x0c
10    eafffffe   B      0x10
14    00001234   B      0x14      <- Code size = 4660 bytes < 28 Kbytes
18    eafffffe   B      0x18
```

14.4.2.2 boot.bin file check

The NVM bootloader program looks for a boot.bin file in the root directory of a FAT12/16/32 formatted NVM Flash.

14.4.3 NVM Bootloader Sequence

Figure 14-7. NVM Bootloader Sequence



14.4.3.1 NAND Flash Boot

The NAND Flash bootloader program uses the EBI CS0. It uses both valid code detections. First it searches a boot.bin file. Then it analyzes the ARM exception vectors.

The first block must be guaranteed by the manufacturer. There is no ECC check.

After NAND interface configuration, the Manufacturer ID is read. If it is different from 0xFF, the Device ID is read. If not, the NAND Flash boot is aborted. The Boot program contains a list of SLC small block Device ID with their characteristics (size, bus width, voltage) (see Table 14-1). If the device ID is not found in this list, the NAND Flash device is considered as a SLC large block and its characteristics are get by reading the Extended Device ID byte 3.

Supported NAND Flash Devices

The SLC small block NAND Flash devices that are supported are described in the following table.

Table 14-1. Supported SLC Small Block NAND Flash

Device ID	Size (Mbytes)	Page size (bytes)	Block size (bytes)	Bus width	*Voltage (V)*
0x6E	1	256	4096	8	5
0x64	2	256	4096	8	5
0x6B	4	512	8196	8	5
0xE8	1	256	4096	8	3.3
0xEC	1	256	4096	8	3.3
0xEA	2	256	4096	8	3.3
0xD5	4	512	8196	8	3.3
0xE3	4	512	8196	8	3.3
0xE5	4	512	8196	8	3.3
0xD6	8	512	8196	8	3.3
0x39	8	512	8196	8	1.8
0xE6	8	512	8196	8	3.3
0x49	8	512	8196	16	1.8
0x59	8	512	8196	16	3.3
0x33	16	512	16384	8	1.8
0x73	16	512	16384	8	3.3
0x43	16	512	16384	16	1.8
0x53	16	512	16384	16	3.3
0x35	32	512	16384	8	1.8
0x75	32	512	16384	8	3.3
0x45	32	512	16384	16	1.8
0x55	32	512	16384	16	3.3
0x36	64	512	16384	8	1.8
0x76	64	512	16384	8	3.3
0x46	64	512	16384	16	1.8
0x56	64	512	16384	16	3.3
0x78	128	512	16384	8	1.8
0x79	128	512	16384	8	3.3
0x72	128	512	16384	16	1.8
0x74	128	512	16384	16	3.3

The NAND boot also supports all the SLC large block NAND Flash devices.

14.4.3.2 SDCard Boot

The SDCard bootloader uses the MCI0. It uses only one valid code detection. It searches a boot.bin file.

Supported SDCard Devices

All SDCard memories compliant with SD Memory Card Specification V1.0. The SD Card boot doesn't support the SDHC cards.

14.4.3.3 SPI Flash Boot

The SPI Flash bootloader uses the SPI0, first on Chip Select 0, then on Chip Select 1. It uses only one valid code detection. It analyzes the ARM exception vectors. Two kinds of SPI Flash are supported, the DataFlash and the Serial Flash.

The SPI Flash read is done thanks to a Continuous Read command from address 0x0. This command is common for DataFlash and Serial Flash devices.

Supported DataFlash Devices

The SPI Flash Boot program supports all Atmel DataFlash devices.

Table 14-2. 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

Supported Serial Flash Devices

The SPI Flash Boot program supports all Serial Flash devices.

14.4.3.4 TWI Eeprom Boot

The TWI EEPROM Bootloader uses TWI. It uses only one valid code detection. It analyzes the ARM exception vectors.

Supported TWI EEPROM Devices

All TWI EEPROM memories using 7-bit device address 0x50.

14.4.4 Hardware and Software Constraints

The NVM 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 output pins used by the NVM drivers and the connected devices may appear.

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

Table 14-3 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.

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 14-3. PIO Driven during Boot Program Execution

NVM Bootloader	Peripheral	Pin	PIO Line
NAND	EBI CS0 SMC	NANDCS	PIOD15
	EBI CS0 SMC	NAND ALE	A21
	EBI CS0 SMC	NAND CLE	A22
	EBI CS0 SMC	Cmd/Addr/Data	D[16:0]
SD Card	MCI0	MCI0_CK	PIOA2
	MCI0	MCI0_CD	PIOA1
	MCI0	MCI0_D0	PIOA0
	MCI0	MCI0_D1	PIOA3
	MCI0	MCI0_D2	PIOA4
	MCI0	MCI0_D3	PIOA5
SPI Flash	SPI0	MOSI	PIOA1
	SPI0	MISO	PIOA0
	SPI0	SPCK	PIOA2
	SPI0	NPCS0	PIOA5
	SPI0	NPCS1	PIOA3
TWI Eeprom	TWI	TWD	PIOB4
	TWI	TWCK	PIOB5
SAM-BA monitor	DBGU	DRXD	PIOC30
	DBGU	DTXD	PIOC31

14.5 SAM-BA Monitor

If no valid code has been found in NVM during the NVM bootloader sequence, the SAM-BA monitor program is performed.

The SAM-BA monitor principle is to:

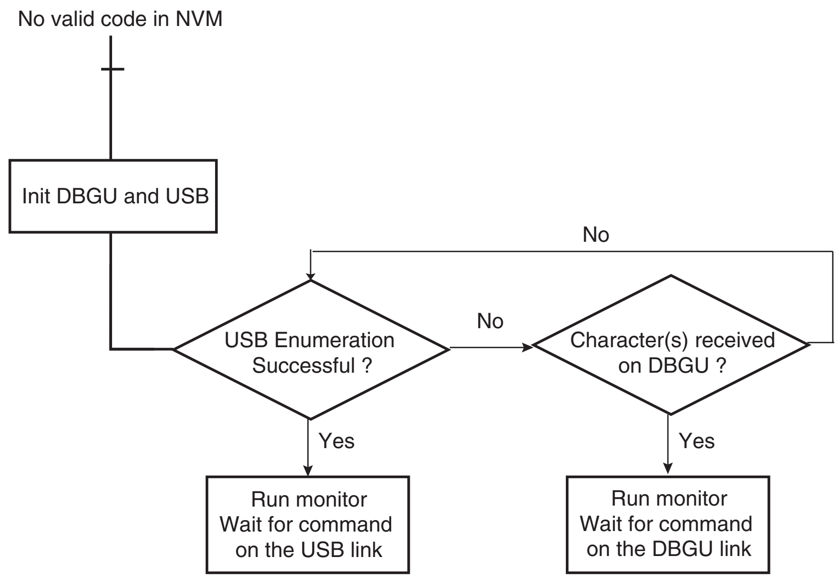
– Initialize DBGU and USB

– Check if USB 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 14-4.

Figure 14-8. SAM-BA Monitor



14.5.1 Command List

Table 14-4. Commands Available through the SAM-BA Monitor

Command	Action	Argument(s)	Example
O	write a byte	Address, Value#	**O**200001,CA#
o	read a byte	Address,#	**o**200001,#
H	write a half word	Address, Value#	**H**200002,CAFE#
h	read a half word	Address,#	**h**200002,#
W	write a word	Address, Value#	**W**200000,CAFEDECA#
w	read a word	Address,#	**w**200000,#
S	send a file	Address,#	**S**200000,#
R	receive a file	Address, NbOfBytes#	**R**200000,1234#
G	go	Address#	**G**200200#
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 followed 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 Boot Program version
  - *Output*: the boot program version, followed by: '>'

14.5.2 DBGU Serial Port

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

14.5.2.1 Supported External Crystal/External Clocks

The supported frequencies by the SAM-BA monitor to allow the DBGU communication are:

- for external crystal: 12 MHz
- for external clock: 1.4 MHz, 2 MHz, 2.8 MHz, 4 MHz, 5.5 MHz, 7.5 MHz, 10 MHz, 14 MHz, 20 MHz, 28 MHz, 40 MHz, 50 MHz.

14.5.2.2 Xmodem Protocol

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.

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.

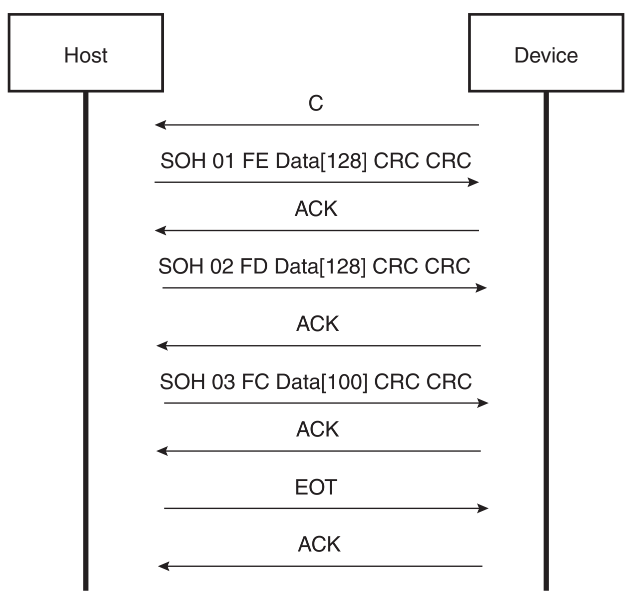
Xmodem protocol with CRC is accurate provided both sender and receiver report successful transmission. Each block of the transfer looks like:

<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 14-9 shows a transmission using this protocol.

Figure 14-9. Xmodem Transfer Example



14.5.3 USB Device Port

14.5.3.1 Supported External Crystal/External Clocks

The supported frequencies by the SAM-BA monitor to allow the USB communication are:

- for external crystal: 12 MHz
- for external clock: 5 MHz, 12 MHz, 27 MHz

14.5.3.2 USB Class

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, 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.

14.5.3.3 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 14-5. 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.
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 14-6. 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.

14.5.3.4 Communication Endpoints

There are two communication endpoints and endpoint 0 is used for the enumeration process. Endpoint 1 is a 64-byte Bulk OUT endpoint and endpoint 2 is a 64-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.

15. Reset Controller (RSTC)

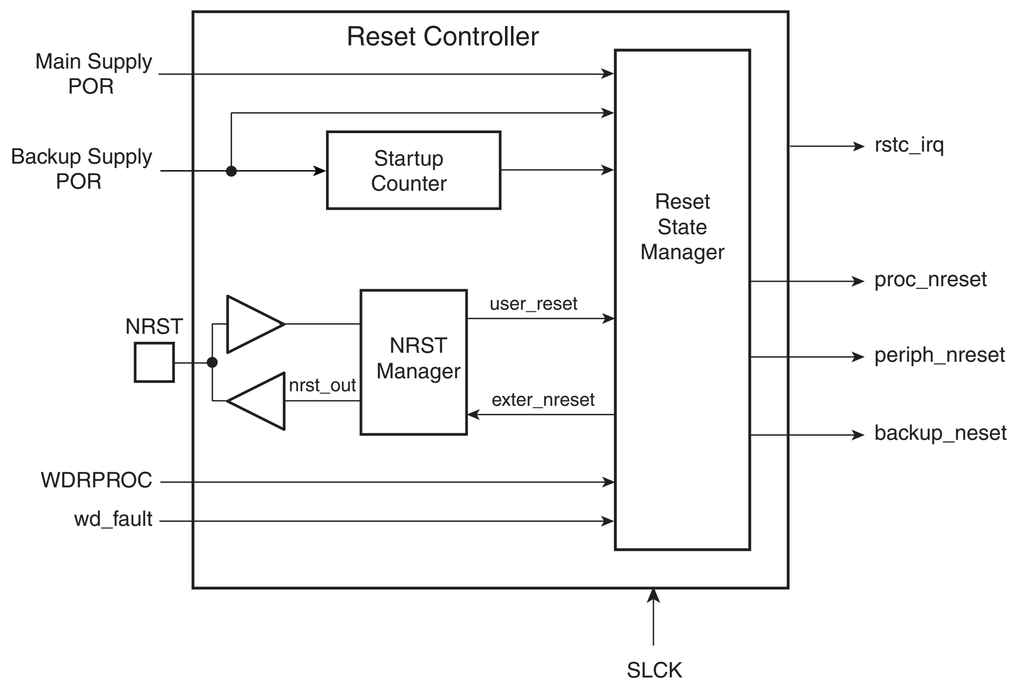
15.1 Description

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.

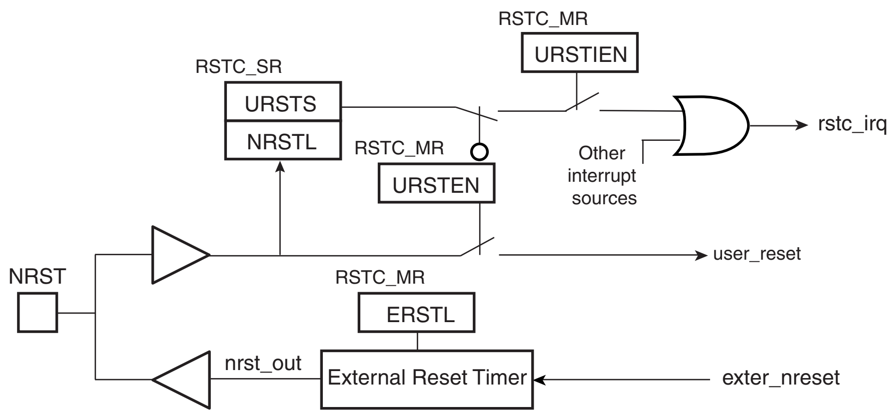
The startup counter waits for the complete crystal oscillator startup. The wait delay is given by the crystal oscillator startup time maximum value that can be found in the section Crystal Oscillator Characteristics in the Electrical Characteristics section of the product documentation.

The Reset Controller Mode Register (RSTC_MR), allowing the configuration of the Reset Controller, is powered with VDDBU, so that its configuration is saved as long as VDDBU 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 µs and 2 seconds. Note that ERSTL at 0 defines a two-cycle duration for the NRST pulse.

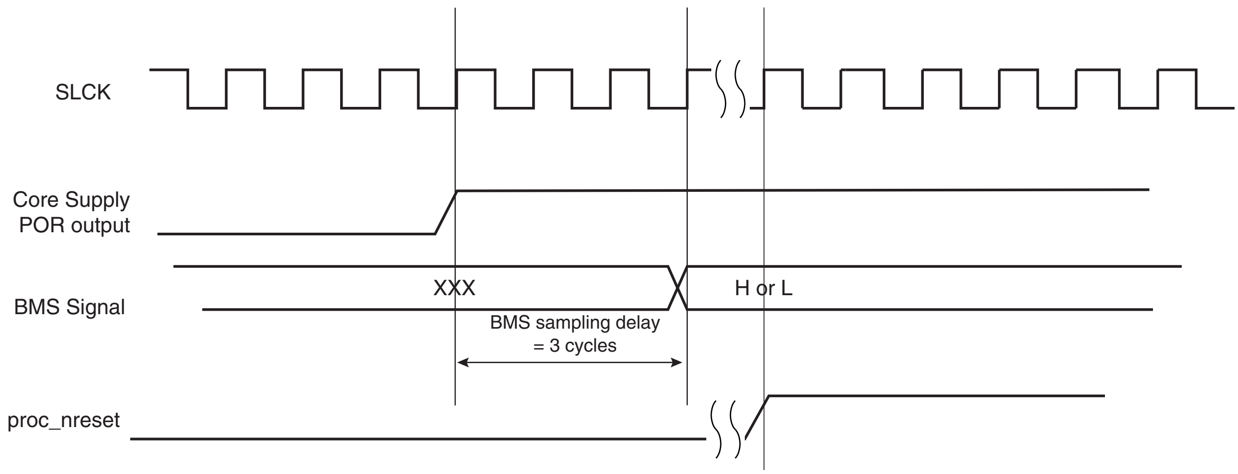
This feature allows the Reset Controller to shape the NRST pin level, and thus to guarantee that the NRST line is driven low for a time compliant with potential external devices connected on the system reset.

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

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

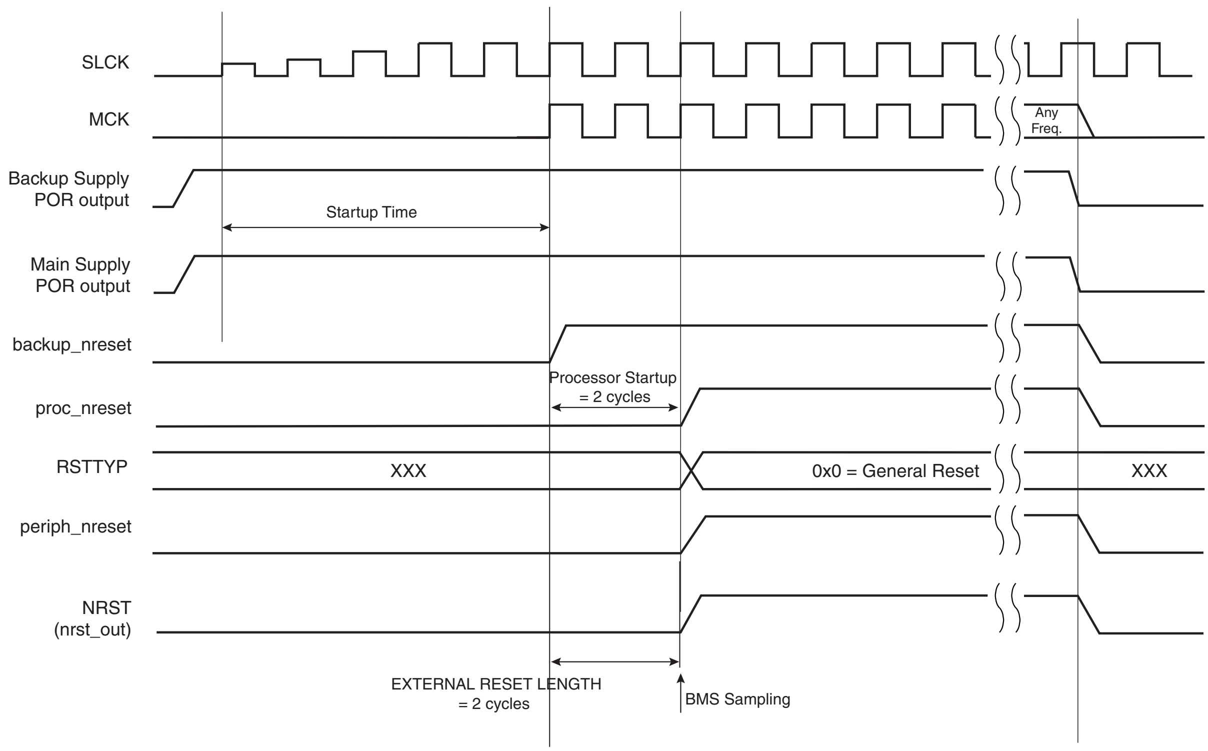
When VDDBU is detected low by the Backup Supply POR Cell, all resets signals are immediately asserted, even if the Main Supply POR Cell does not report a Main Supply shutdown.

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*

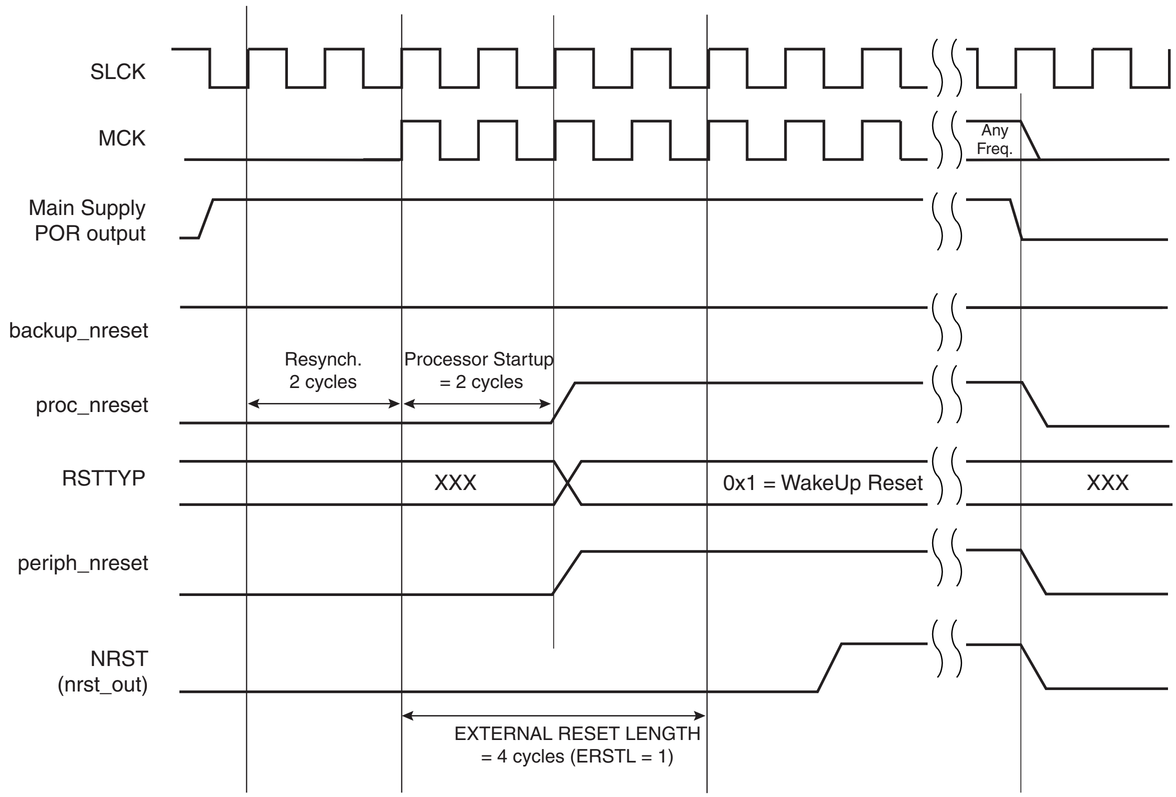
The Wake-up Reset occurs when the Main Supply is down. When the Main Supply POR output is active, all the reset signals are asserted except backup_nreset. When the Main Supply powers up, the POR output is resynchronized on Slow Clock. The processor clock is then re-enabled during 2 Slow Clock cycles,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

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

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

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

Figure 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.

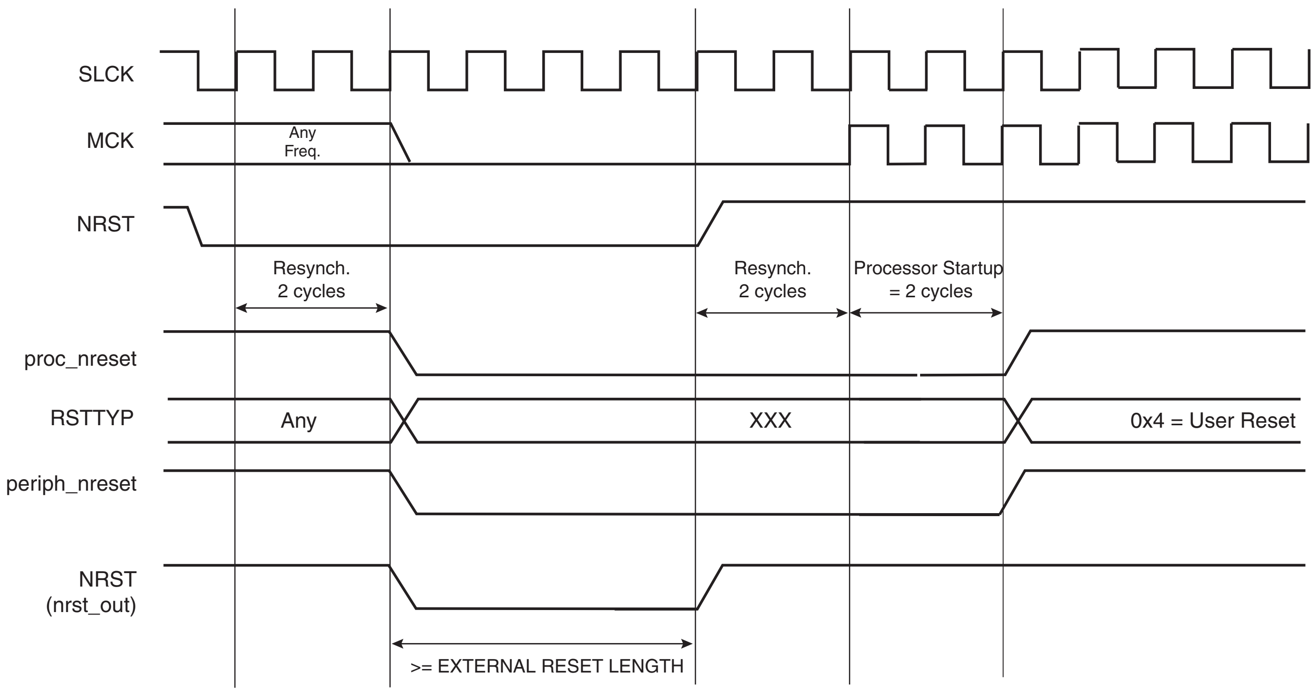
The User Reset is entered as soon as a low level is detected on NRST. The Processor Reset and the Peripheral Reset are asserted.

The User Reset is left when NRST rises, after a two-cycle resynchronization time and a 2-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.
  Except for Debug purposes, PERRST must always be used in conjunction with PROCRST (PERRST and PROCRST set both at 1 simultaneously.)
- EXTRST: Writing EXTRST at 1 asserts low the NRST pin during a time defined by the field ERSTL in the Mode Register (RSTC_MR).

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

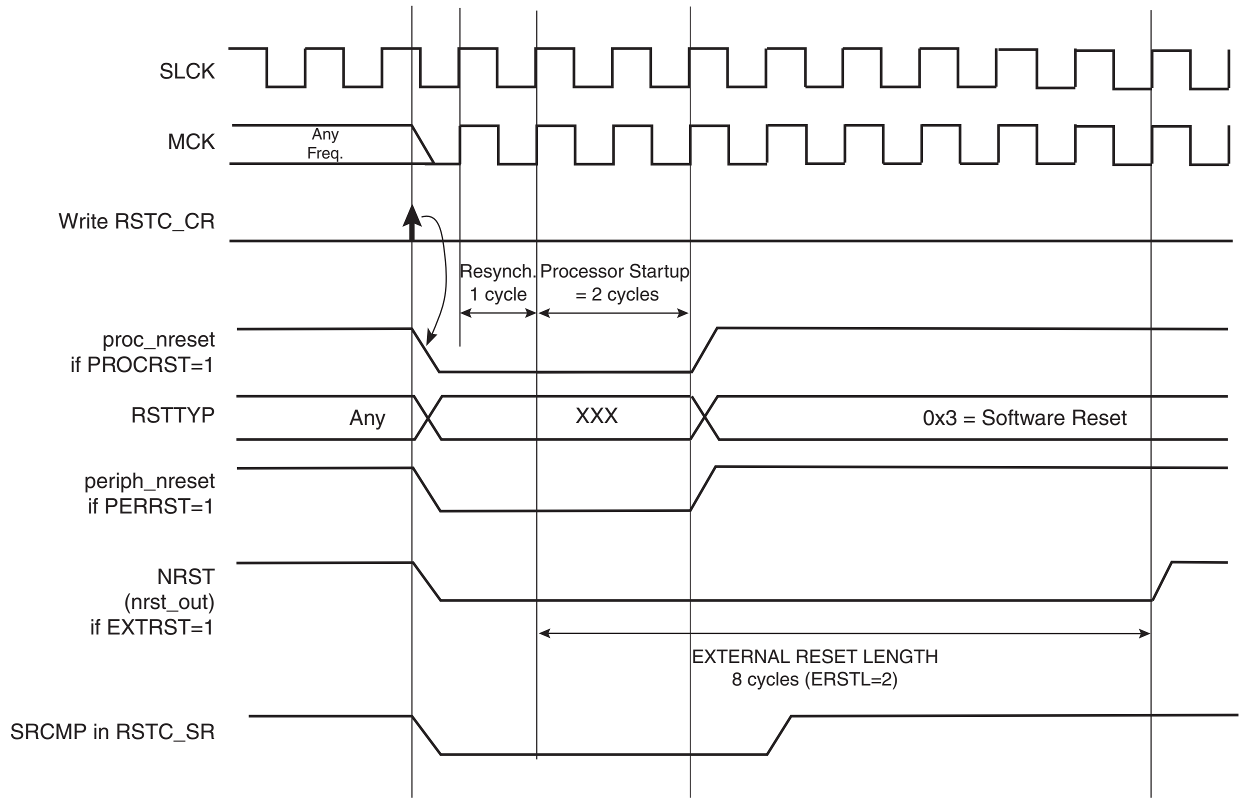
The internal reset signals are asserted as soon as the register write is performed. This is detected on the Master Clock (MCK). They are released when the software reset is left, i.e.; synchronously to SLCK.

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

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

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

Figure 15-7. Software Reset



15.3.4.5 Watchdog Reset

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

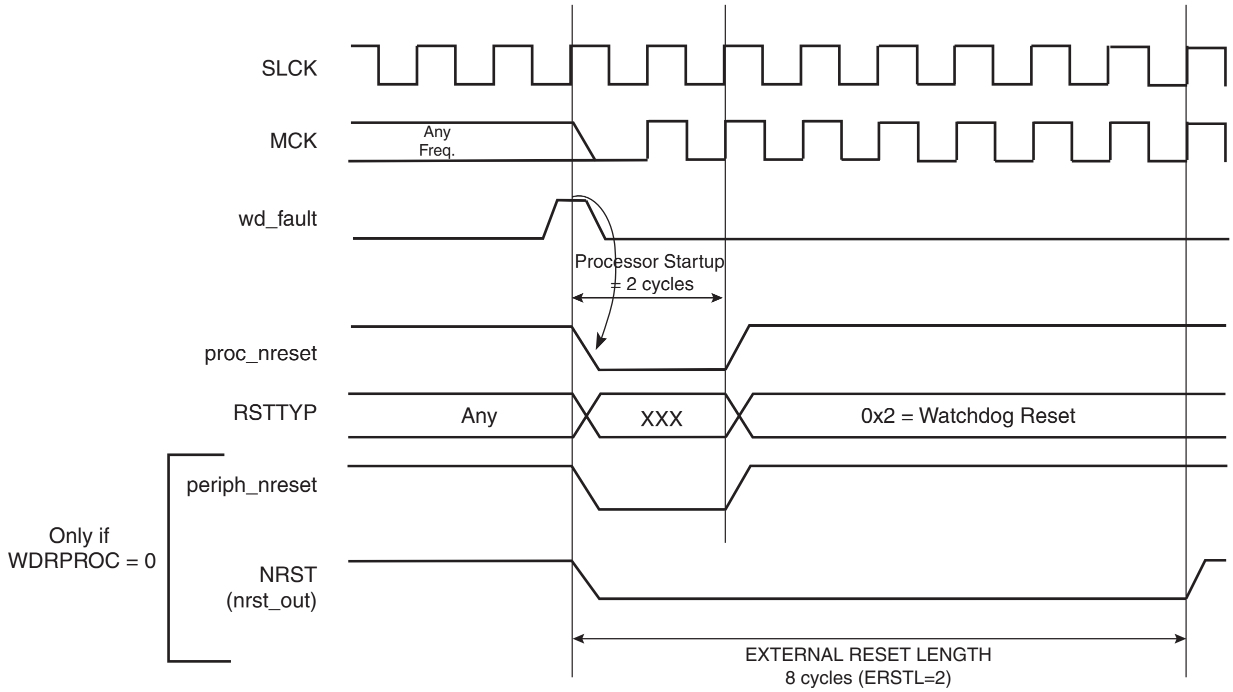
When in Watchdog Reset, assertion of the reset signals depends on the WDRPROC bit in WDT_MR:

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

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

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

Figure 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:

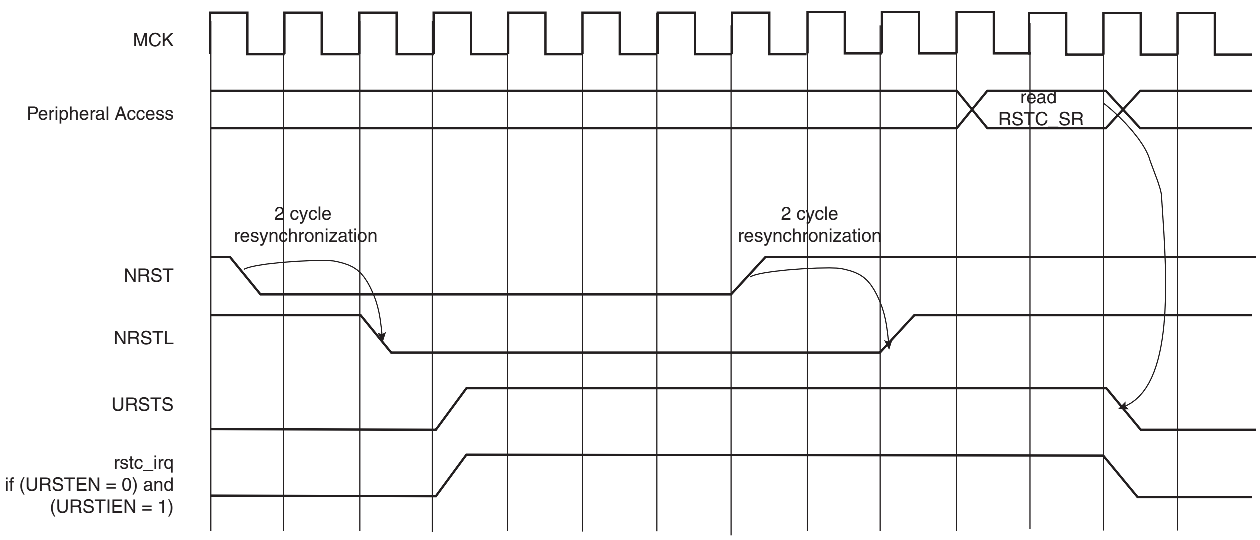
- When in User Reset:
  - A watchdog event is impossible because the Watchdog Timer is being reset by the proc_nreset signal.
  - A software reset is impossible, since the processor reset is being activated.
- When in Software Reset:
  - A watchdog event has priority over the current state.
  - The NRST 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. Register Mapping

Offset	Register	Name	Access	Reset	Back-up Reset
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

Name: RSTC_CR

Address: 0xFFFFFD00

Access: 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

Name: RSTC_SR

Address: 0xFFFFFD04

Access: 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

Name: RSTC_MR

Address: 0xFFFFFD08

Access: Read-write

31	30	29	28	27	26	25	24
KEY							

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

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

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

- **URSTEN: User Reset Enable**

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

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

- **URSTIEN: User Reset Interrupt Enable**

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

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

- **ERSTL: External Reset Length**

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

- **KEY: Password**

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

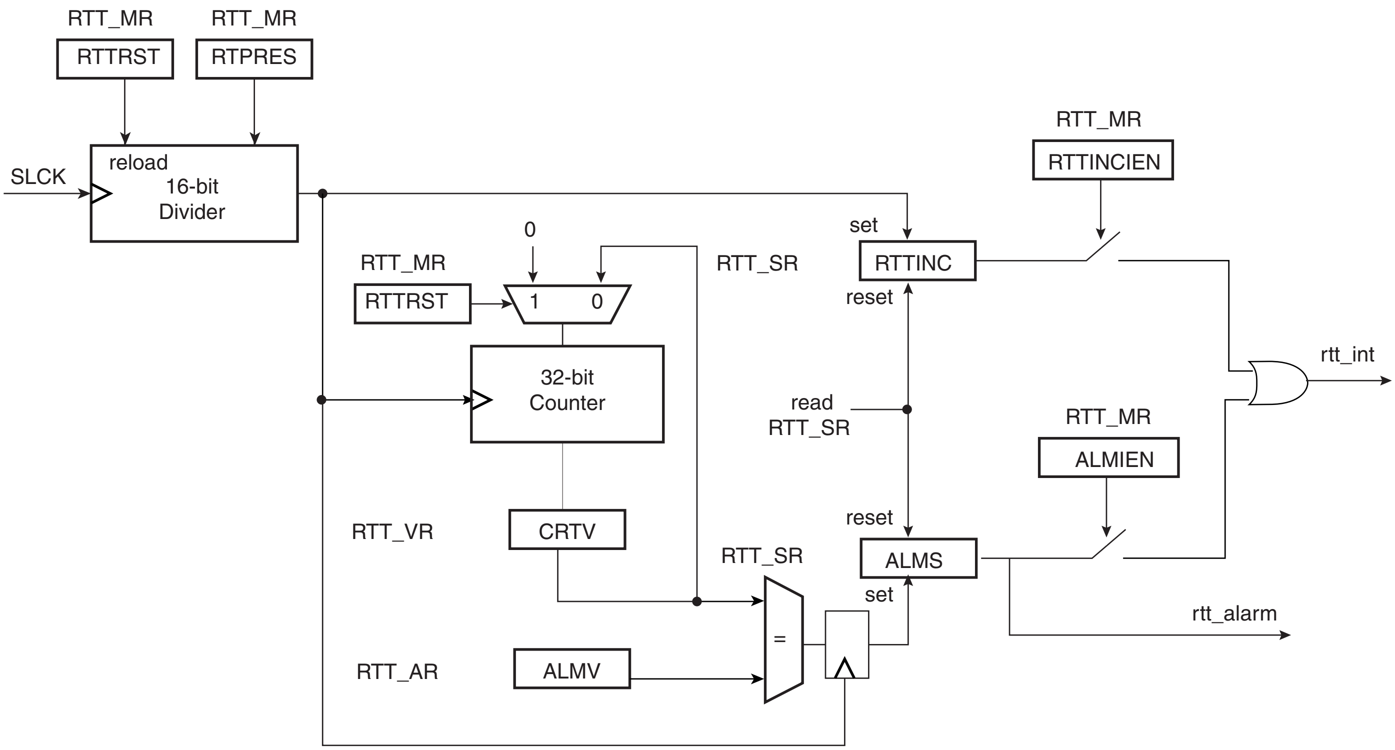
16. Real-time Timer (RTT)

16.1 Description

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

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 kHz). 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.

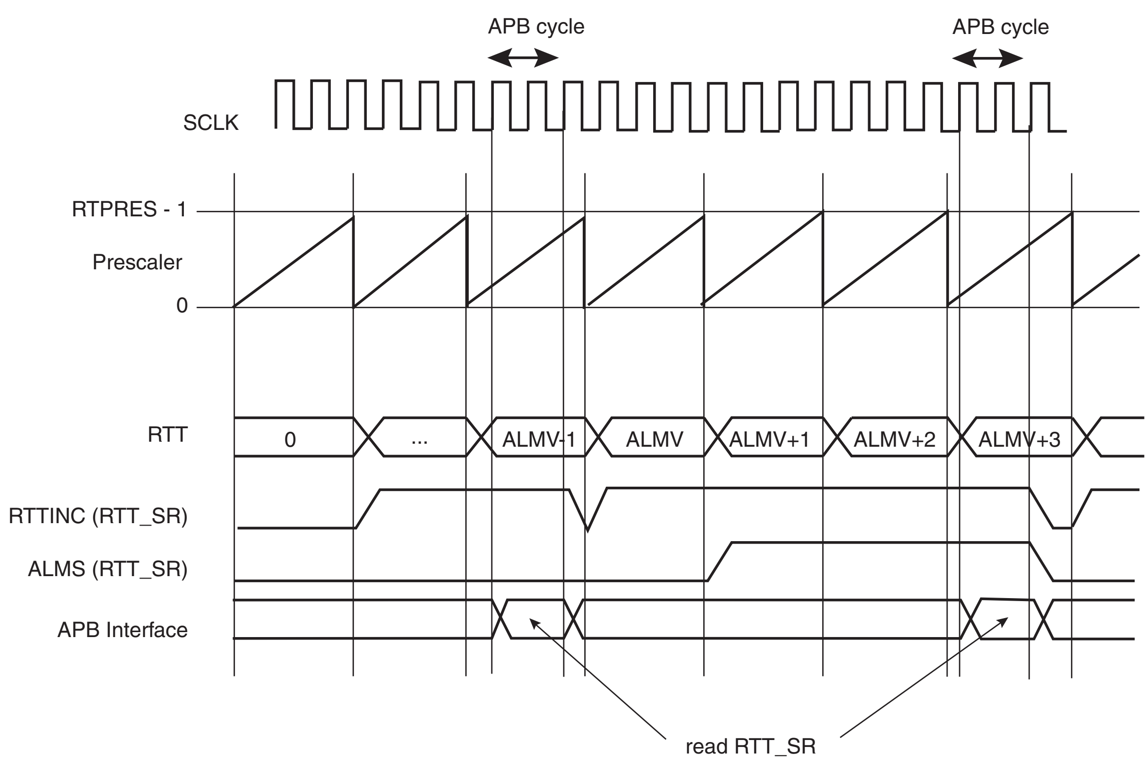
The bit RTTINC in RTT_SR is set each time the Real-time Timer counter is incremented. This bit can be used to start a periodic interrupt, the period being one second when the RTPRES is programmed with 0x8000 and Slow Clock equal to 32.768 Hz.

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

Table 16-1. Register Mapping

Offset	Register	Name	Access	Reset
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

16.4.1 Real-time Timer Mode Register

Name: RTT_MR

Address: 0xFFFFFD20

Access Type: Read/Write

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

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

15	14	13	12	11	10	9	8
RTPRES							

7	6	5	4	3	2	1	0
RTPRES							

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

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

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

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

- **ALMIEN: Alarm Interrupt Enable**

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

1 = The bit ALMS in RTT_SR asserts interrupt.

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

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

1 = The bit RTTINC in RTT_SR asserts interrupt.

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

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

16.4.2 Real-time Timer Alarm Register

Name: RTT_AR

Address: 0xFFFFFD24

Access: 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.

16.4.3 Real-time Timer Value Register

Name: RTT_VR

Address: 0xFFFFFD28

Access: 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.

16.4.4 Real-time Timer Status Register

Name: RTT_SR

Address: 0xFFFFFD2C

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	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.

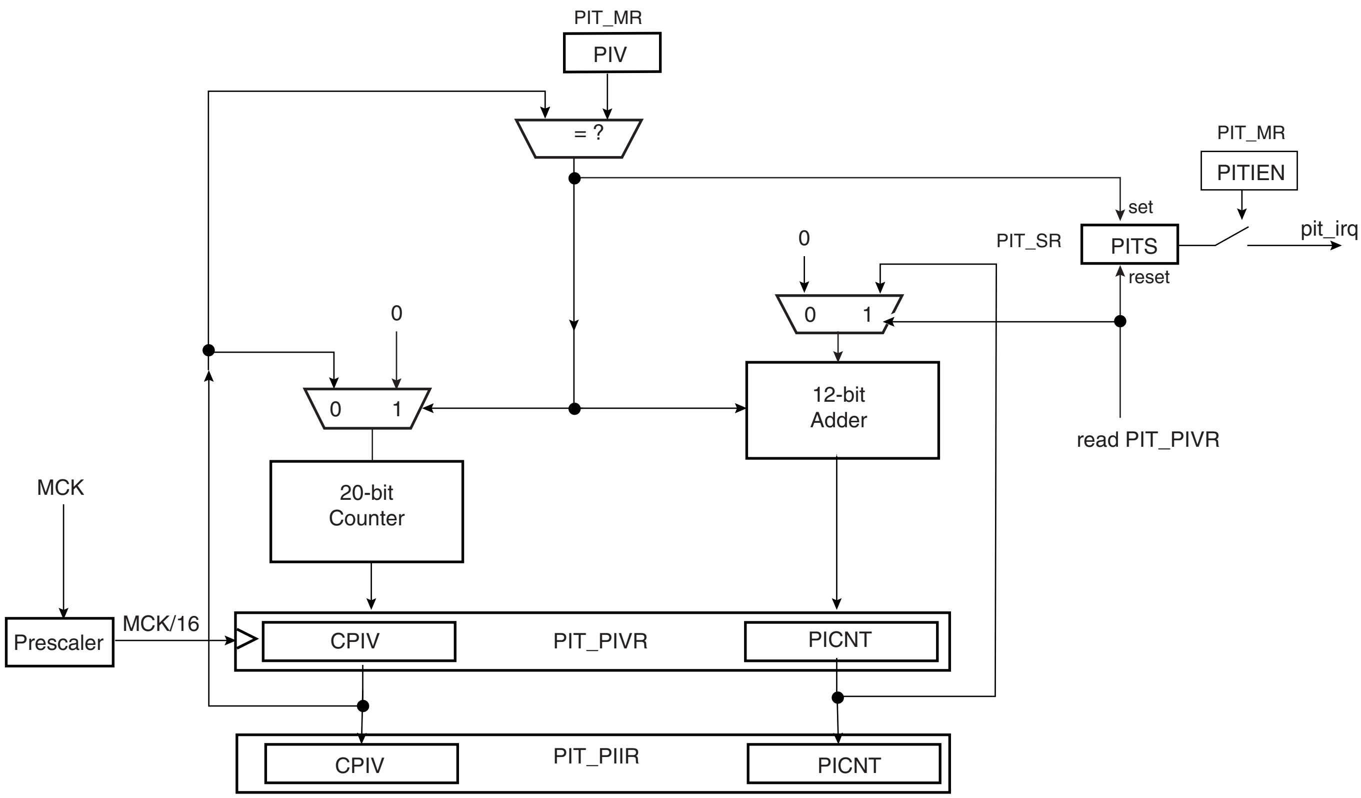
17. Periodic Interval Timer (PIT)

17.1 Description

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

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.

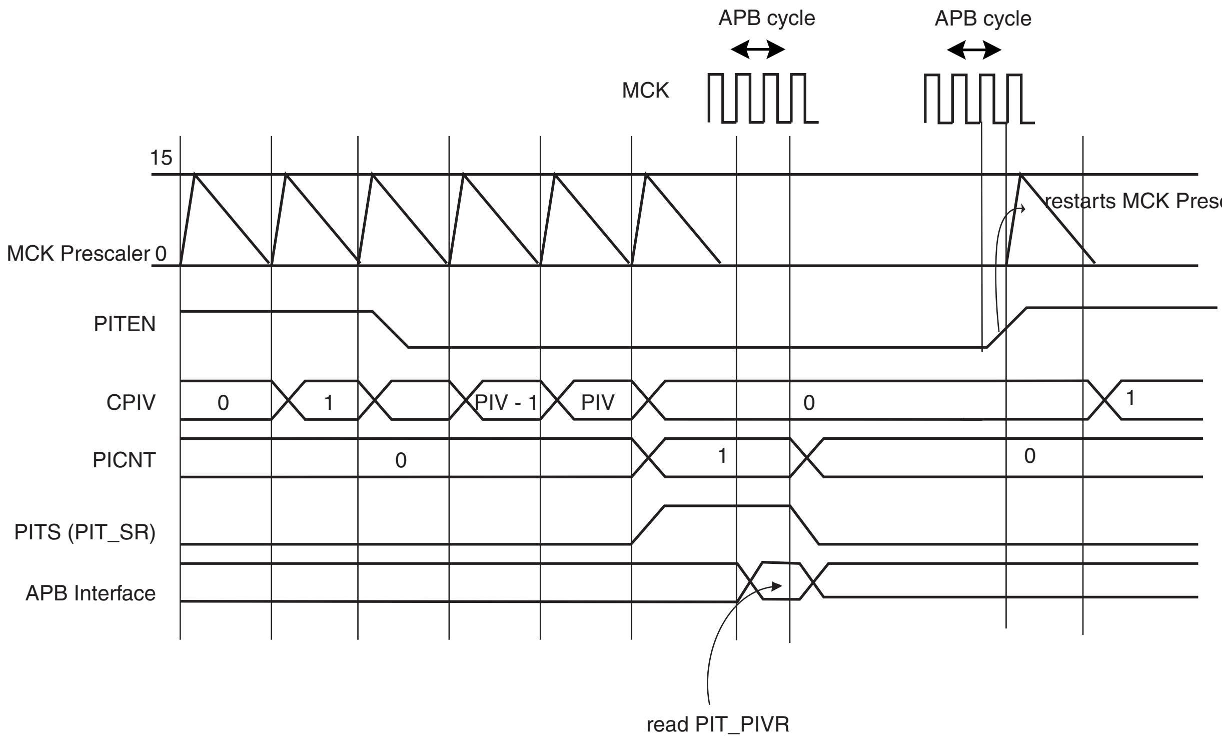
When CPIV and PICNT values are obtained by reading the Periodic Interval Value Register (PIT_PIVR), the overflow counter (PICNT) is reset and the PITS is cleared, thus acknowledging the interrupt. The value of PICNT gives the number of periodic intervals elapsed since the last read of PIT_PIVR.

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. Register Mapping

Offset	Register	Name	Access	Reset
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

Name: PIT_MR

Address: 0xFFFFFD30

Access: 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

Name: PIT_SR

Address: 0xFFFFFD34

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	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

Name: PIT_PIVR

Address: 0xFFFFFD38

Access: Read-only

31	30	29	28	27	26	25	24
PICNT							

23	22	21	20	19	18	17	16
PICNT				CPIV			

15	14	13	12	11	10	9	8
CPIV							

7	6	5	4	3	2	1	0
CPIV							

Reading this register clears PITS in PIT_SR.

- **CPIV: Current Periodic Interval Value**

Returns the current value of the periodic interval timer.

- **PICNT: Periodic Interval Counter**

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

17.4.4 Periodic Interval Timer Image Register

Name: PIT_PIIR

Address: 0xFFFFFD3C

Access: Read-only

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

- **CPIV: Current Periodic Interval Value**

Returns the current value of the periodic interval timer.

- **PICNT: Periodic Interval Counter**

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

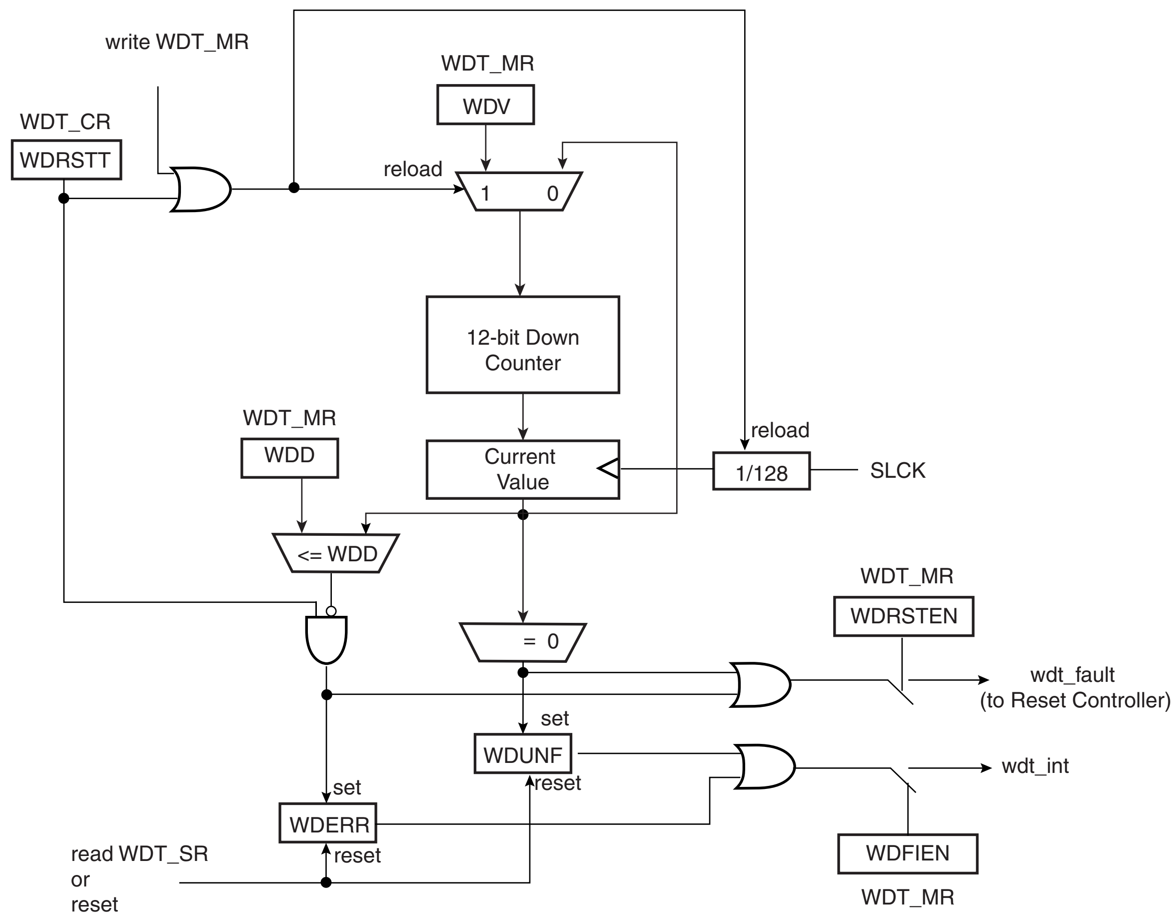
18. Watchdog Timer (WDT)

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

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

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

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

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

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

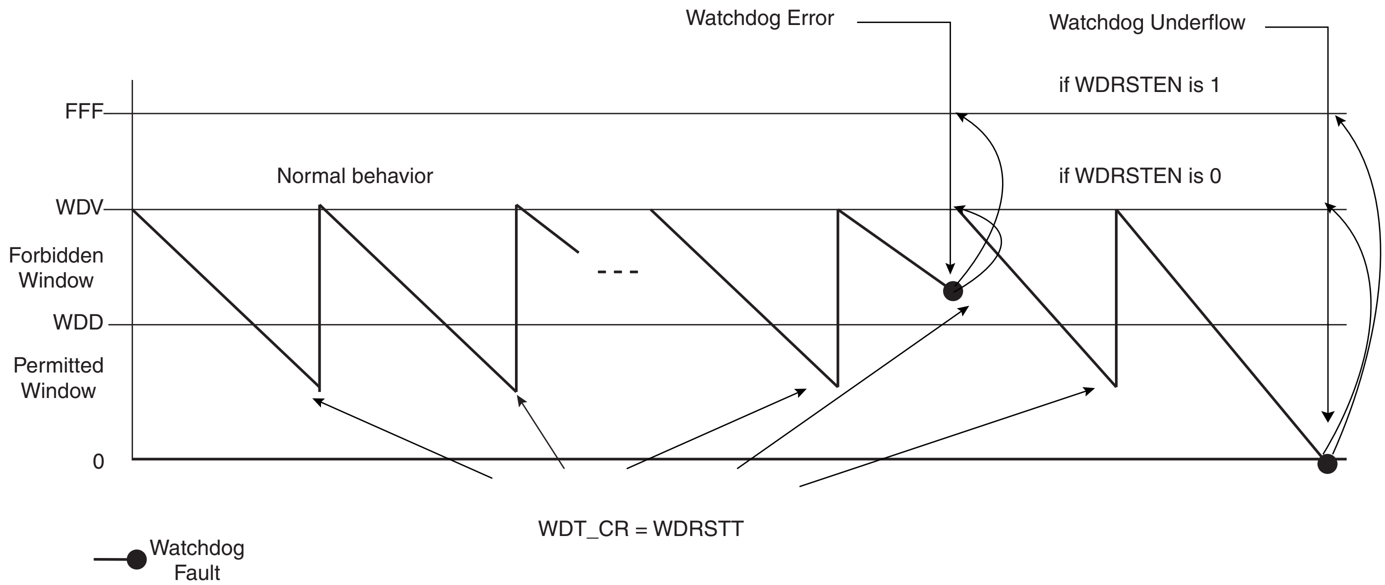
The status bits WDUNF (Watchdog Underflow) and WDERR (Watchdog Error) trigger an interrupt, provided the bit WDFIEN is set in the mode register. The signal “wdt_fault” to the reset controller causes a Watchdog reset if the WDRSTEN bit is set as already explained in the reset controller programmer Datasheet.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

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

Writing the WDT_MR reloads and restarts the down counter.

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

Figure 18-2. Watchdog Behavior



18.4 Watchdog Timer (WDT) User Interface

Table 18-1. Register Mapping

Offset	Register	Name	Access	Reset
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

Name: WDT_CR

Address: 0xFFFFFD40

Access: 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

Name: WDT_MR

Address: 0xFFFFFD44

Access: 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

Name: WDT_SR

Address: 0xFFFFFD48

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	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. General Purpose Backup Registers (GPBR)

19.1 Description

The System Controller embeds 4 general-purpose backup registers.

19.2 General Purpose Backup Registers (GPBR) User Interface

Table 19-1. Register Mapping

Offset	Register	Name	Access	Reset
0x0	General Purpose Backup Register 0	SYS_GPBR0	Read-write	–
...	...	...	...	...
0xC	General Purpose Backup Register 3	SYS_GPBR3	Read-write	–

19.2.1 General Purpose Backup Register x

Name: SYS_GPBRx

Addresses: 0xFFFFFD60 [0], 0xFFFFFD64 [1], 0xFFFFFD68 [2], 0xFFFFFD6C [3]

Access: Read-write

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

- **GPBR_VALUEx: Value of GPBR x**

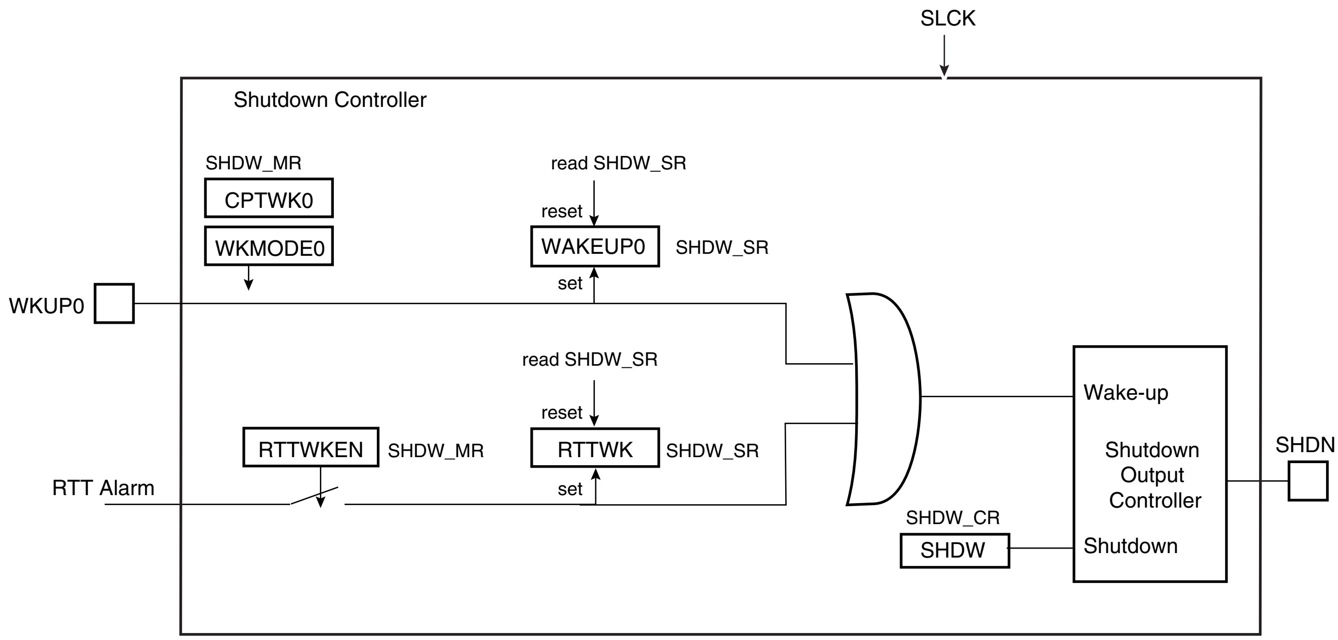
20. Shutdown Controller (SHDWC)

20.1 Description

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

20.2 Block Diagram

Figure 20-1. Shutdown Controller Block Diagram



20.3 I/O Lines Description

Table 20-1. I/O Lines Description

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

20.4 Product Dependencies

20.4.1 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.

20.5 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.

20.6 Shutdown Controller (SHDWC) User Interface

Table 20-2. 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

20.6.1 Shutdown Control Register

Name: SHDW_CR

Address: 0xFFFFFD10

Access: 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.

20.6.2 Shutdown Mode Register

Name: SHDW_MR

Address: 0xFFFFFD14

Access: Read/Write

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

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

15	14	13	12	11	10	9	8
–				–	–	–	

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

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

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.

20.6.3 Shutdown Status Register

Name: SHDW_SR

Address: 0xFFFFFD18

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WAKEUP0

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

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

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

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

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

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

21. Bus Matrix

21.1 Description

The 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 12 AHB Masters to 10 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 the 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

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

The Bus Matrix user interface provides Master Remap Control Register (MATRIX_MRCR) that 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 selects the default master type (no default, last access master, fixed default master) whereas the 4-bit FIXED_DEFMSTR field selects a fixed default master provided that DEFMSTR_TYPE is set to fixed default master. Refer to the Section 21.6 "Bus Matrix (MATRIX) User Interface".

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, arbitrating 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 Section 21.4.1.1 "Undefined Length Burst Arbitration").
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 Section 21.4.1.2 "Slot Cycle Limit Arbitration").

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 algorithms 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 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 EBI Pad Output Strength

In order to manage 1.8 or 3.3V memory devices, the Output Strength (OS) of the EBI pads are programmable.

The EBI pads are divided into 3 groups:

- DATA (D0-D15, DQM0-DQM1)
- SDCK (DQS0-DQS1, SDCK, SDCKN, BCCK)
- ADDR (A2-A22, D16-D31, DQM2-DQM3, NANDWE, NANDOE, SDA10, NWR0, NCS0, NCS1, NRD, RAS, CAS,SDWE,SDCKW,SDDRCS, PD7-PD15)

According to the mode, the EBI pad groups can be controlled together or independently. (See, Section 21.7.3 "EBI Chip Select Assignment Register", EBI_CSA.)

In the first mode (EBI_OSMODE = 0), the output strength of all the EBI pads is controlled together via bit fields: EBI_OSALLN0 and EBI_OSALLN1 (**Warning:** EBI_OSALLN1 is bit 16 and EBI_OSALLN0 is bit 17).

Table 21-1. Output Strength Configuration for EBI_OSMODE = 0

EBI_OSALLN1	EBI_OSALLN0	Description
1	1	lowest drive strength of the pads, recommended for 3.3V devices
1	0	
0	1	recommended for 1.8V devices
0	0	highest drive strength of the pads.

In the second mode (EBI_OSMODE = 1), the output strength of each group is controllable independently via bit fields: EBI_OSDATA, EBI_OSSDCK, EBI_OSADDR.

Table 21-2. Output Strength Configuration for EBI_OSMODE = 1

EBI_OSxxxx	Description	
0	0	lowest drive strength of the pads, recommended for 3.3V devices
0	1	
1	0	recommended for 1.8V devices
1	1	highest drive strength of the pads

21.6 Bus Matrix (MATRIX) User Interface

Table 21-3. Register Mapping

Offset	Register	Name	Access	Reset
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	Master Configuration Register 6	MATRIX_MCFG6	Read-write	0x00000000
0x001C	Master Configuration Register 7	MATRIX_MCFG7	Read-write	0x00000000
0x0020	Master Configuration Register 8	MATRIX_MCFG8	Read-write	0x00000000
0x0024	Master Configuration Register 9	MATRIX_MCFG9	Read-write	0x00000000
0x0028	Master Configuration Register 10	MATRIX_MCFG10	Read-write	0x00000000
0x002C	Master Configuration Register 11	MATRIX_MCFG11	Read-write	0x00000000
0x0030 - 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	Slave Configuration Register 6	MATRIX_SCFG6	Read-write	0x00000010
0x005C	Slave Configuration Register 7	MATRIX_SCFG7	Read-write	0x00000010
0x0060	Slave Configuration Register 8	MATRIX_SCFG8	Read-write	0x00000010
0x0064	Slave Configuration Register 9	MATRIX_SCFG9	Read-write	0x00000010
0x0068 - 0x007C	Reserved	–	–	–
0x0080	Priority Register A for Slave 0	MATRIX_PRAS0	Read-write	0x00000000
0x0084	Priority Register B for Slave 0	MATRIX_PRBS0	Read-write	0x00000000
0x0088	Priority Register A for Slave 1	MATRIX_PRAS1	Read-write	0x00000000
0x008C	Priority Register B for Slave 1	MATRIX_PRBS1	Read-write	0x00000000
0x0090	Priority Register A for Slave 2	MATRIX_PRAS2	Read-write	0x00000000
0x0094	Priority Register B for Slave 2	MATRIX_PRBS2	Read-write	0x00000000
0x0098	Priority Register A for Slave 3	MATRIX_PRAS3	Read-write	0x00000000
0x009C	Priority Register B for Slave 3	MATRIX_PRBS3	Read-write	0x00000000
0x00A0	Priority Register A for Slave 4	MATRIX_PRAS4	Read-write	0x00000000
0x00A4	Priority Register B for Slave 4	MATRIX_PRBS4	Read-write	0x00000000
0x00A8	Priority Register A for Slave 5	MATRIX_PRAS5	Read-write	0x00000000

Table 21-3. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x00AC	Priority Register B for Slave 5	MATRIX_PRBS5	Read-write	0x00000000
0x00B0	Priority Register A for Slave 6	MATRIX_PRAS6	Read-write	0x00000000
0x00B4	Priority Register B for Slave 6	MATRIX_PRBS6	Read-write	0x00000000
0x00B8	Priority Register A for Slave 7	MATRIX_PRAS7	Read-write	0x00000000
0x00BC	Priority Register B for Slave 7	MATRIX_PRBS7	Read-write	0x00000000
0x00C0	Priority Register A for Slave 8	MATRIX_PRAS8	Read-write	0x00000000
0x00C4	Priority Register B for Slave 8	MATRIX_PRBS8	Read-write	0x00000000
0x00C8	Priority Register A for Slave 9	MATRIX_PRAS9	Read-write	0x00000000
0x00CC	Priority Register B for Slave 9	MATRIX_PRBS9	Read-write	0x00000000
0x00D0 - 0x00FC	Reserved	–	–	–
0x0100	Master Remap Control Register	MATRIX_MRCR	Read-write	0x00000000
0x0104 - 0x010C	Reserved	–	–	–

21.6.1 Bus Matrix Master Configuration Registers

Name: MATRIX_MCFG0...MATRIX_MCFG11

Address: 0xFFFFEA00

Access: 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 rearbitration at each beat of the INCR burst.

2: Four-beat Burst

The undefined length burst is split into four-beat burst allowing rearbitration at each four-beat burst end.

3: Eight-beat Burst

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

4: Sixteen-beat Burst

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

21.6.2 Bus Matrix Slave Configuration Registers

Name: MATRIX_SCFG0...MATRIX_SCFG9

Address: 0xFFFFEA40

Access: 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 DEFMSTR_TYPE is 2. Specifying the number of a master which is not connected to the selected slave is equivalent to setting DEFMSTR_TYPE to 0.

- **ARBT: Arbitration Type**

0: Round-Robin Arbitration

1: Fixed Priority Arbitration

2: Reserved

3: Reserved

21.6.3 Bus Matrix Priority Registers A For Slaves

Name: MATRIX_PRAS0...MATRIX_PRAS9

Addresses: 0xFFFFEA80 [0], 0xFFFFEA88 [1], 0xFFFFEA90 [2], 0xFFFFEA98 [3], 0xFFFFEAA0 [4], 0xFFFFEAA8 [5], 0xFFFFEAB0 [6], 0xFFFFEAB8 [7], 0xFFFFEAC0 [8], 0xFFFFEAC8 [9]

Access: Read-write

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

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

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

7	6	5	4	3	2	1	0
–	–	M1PR		–	–	M0PR	

- **MxPR: Master x Priority**

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

21.6.4 Bus Matrix Priority Registers B For Slaves

Name: MATRIX_PRBS0...MATRIX_PRBS9

Addresses: 0xFFFFEA84 [0], 0xFFFFEA8C [1], 0xFFFFEA94 [2], 0xFFFFEA9C [3], 0xFFFFEAA4 [4], 0xFFFFEAAC [5], 0xFFFFEAB4 [6], 0xFFFFEABC [7], 0xFFFFEAC4 [8], 0xFFFFEACC [9]

Access: 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
–	–	M11PR		–	–	M10PR	

7	6	5	4	3	2	1	0
–	–	M9PR		–	–	M8PR	

- **MxPR: Master x Priority**

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

21.6.5 Bus Matrix Master Remap Control Register

Name: MATRIX_MRCR

Address: 0xFFFFEB00

Access: 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
–	–	–	–	–	–	–	RCB8

7	6	5	4	3	2	1	0
RCB7	RCB6	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.

21.7 Chip Configuration User Interface

Table 21-4. Chip Configuration User Interface

Offset	Register	Name	Access	Reset
0x0110	Reserved	–	–	–
0x0114	MPBlock Slave 0 Special Function Register	MPBS0_SFR	Read-write	0x00000000
0x0118	Reserved	–	–	–
0x011C	MPBlock Slave 1 Special Function Register	MPBS1_SFR	Read-write	0x00000000
0x0120	EBI Chip Select Assignment Register	EBI_CSA	Read-write	0x00010000
0x0124 - 0x0128	Reserved	–	–	–
0x012C	MPBlock Slave 2 Special Function Register	MPBS2_SFR	Read-write	0x00000000
0x0130	MPBlock Slave 3 Special Function Register	MPBS3_SFR	Read-write	0x00000000
0x0134	MPBlock Slave 1 Special Function Register	MPBS1_SFR	Read-write	0x00000000
0x0138 - 0x01FC	Reserved	–	–	–

21.7.1 MPBlock Slave 0 Special Function Register

Name: MPBS0_SFR

Access: Read-write

Reset: 0x0000_0000

31	30	29	28	27	26	25	24
MPBS0_SFR							

23	22	21	20	19	18	17	16
MPBS0_SFR							

15	14	13	12	11	10	9	8
MPBS0_SFR							

7	6	5	4	3	2	1	0
MPBS0_SFR							

- **MPBS0_SFR: MPBlock Slave 0 Special Function Register**

The value of the register is directly connected to MPBlock inputs and may be used to implement any MPBlock configuration register.

21.7.2 MPBlock Slave 1 Special Function Register

Name: MPBS1_SFR

Access: Read-write

Reset: 0x0000_0000

31	30	29	28	27	26	25	24
MPBS1_SFR							

23	22	21	20	19	18	17	16
MPBS1_SFR							

15	14	13	12	11	10	9	8
MPBS1_SFR							

7	6	5	4	3	2	1	0
MPBS1_SFR							

- **MPBS0_SFR: MPBlock Slave 1 Special Function Register**

The value of the register is directly connected to MPBlock inputs and may be used to implement any MPBlock configuration register.

21.7.3 EBI Chip Select Assignment Register

Name: EBI_CSA

Access: Read-write

Reset: 0x0001_0000

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

23	22	21	20	19	18	17	16
EBI_OSADDR		EBI_OSSDCK		EBI_OSDATA		EBI_OSALLAN0	EBI_OSALLAN1

15	14	13	12	11	10	9	8
–	–	–	–	–	–	EBI_DQSPDC	EBI_DBPUC

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- **EBI_DQSPDC: EBI Data Qualifier Strobe Pull-Down Configuration**

0 = EBI DQS0 and DQS1 signals are internally pulled-down to the GNDIOM power ground.

1 = EBI DQS0 and DQS1 signals are not internally pulled-down.

- **EBI_OSALLN1: All EBI Output Strength Configuration Bit 1**

Used when EBI_OSMODE = 0. See Table 21-1, "Output Strength Configuration for EBI_OSMODE = 0"

- **EBI_OSALLN1: All EBI Output Strength Configuration Bit 0**

Used when EBI_OSMODE = 0. See Table 21-1, "Output Strength Configuration for EBI_OSMODE = 0"

- **EBI_OSDATA: EBI DATA Output Strength Configuration**

Used when EBI_OSMODE = 1. See Table 21-2, "Output Strength Configuration for EBI_OSMODE = 1"

- **EBI_ OSSDCK: EBI SDCK Output Strength Configuration**

Used when EBI_OSMODE = 1. See Table 21-2, “Output Strength Configuration for EBI_OSMODE = 1”

- **EBI_ OSADDR: EBI ADDR Output Strength Configuration**

Used when EBI_OSMODE = 1. See Table 21-2, “Output Strength Configuration for EBI_OSMODE = 1”

- **EBI_OSMODE: EBI Output Strength Mode**

0 = EBI pads output strength are controlled by bits EBI_OSALLN0 and EBI_ OSALLN1

1 = EBI pads output strength are controlled by fields EBI_OSDATA, EBI_OSSDCK, EBI_OSADDR

21.7.4 MPBlock Slave 2 Special Function Register

Name: MPBS2_SFR

Access: Read-write

Reset: 0x0000_0000

31	30	29	28	27	26	25	24
MPBS2_SFR							

23	22	21	20	19	18	17	16
MPBS2_SFR							

15	14	13	12	11	10	9	8
MPBS2_SFR							

7	6	5	4	3	2	1	0
MPBS2_SFR							

- **MPBS0_SFR: MPBlock Slave 2 Special Function Register**

The value of the register is directly connected to MPBlock inputs and may be used to implement any MPBlock configuration register.

21.7.5 MPBlock Slave 3 Special Function Register

Name: MPBS3_SFR

Access: Read-write

Reset: 0x0000_0000

31	30	29	28	27	26	25	24
MPBS3_SFR							

23	22	21	20	19	18	17	16
MPBS3_SFR							

15	14	13	12	11	10	9	8
MPBS3_SFR							

7	6	5	4	3	2	1	0
MPBS3_SFR							

- **MPBS0_SFR: MPBlock Slave 3 Special Function Register**

The value of the register is directly connected to MPBlock inputs and may be used to implement any MPBlock configuration register.

21.7.6 APB Bridge Special Function Register

Name: APB_SFR

Access: Read-write

Reset: 0x0000_0000

31	30	29	28	27	26	25	24
APB_SFR							

23	22	21	20	19	18	17	16
APB_SFR							

15	14	13	12	11	10	9	8
APB_SFR							

7	6	5	4	3	2	1	0
APB_SFR							

- **APB_SFR: APB Bridge Special Function Register**

The value of the register is directly connected to MPBlock inputs and may be used to implement any MPBlock configuration register.

22. External Bus Interface (EBI)

22.1 Description

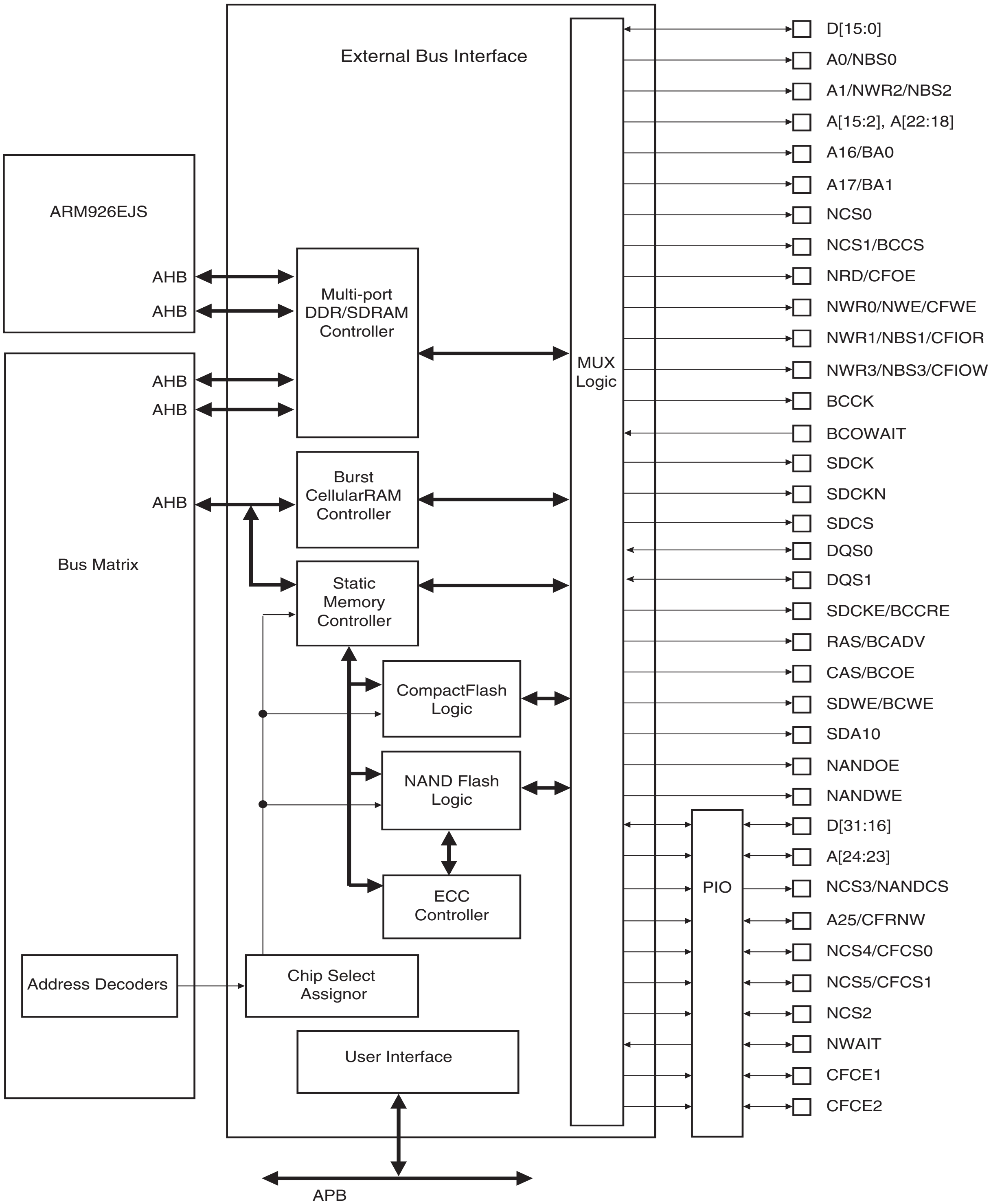
The External Bus Interface (EBI) is designed to ensure the successful data transfer between several external devices and the embedded Memory Controller of an ARM-based device. The Static Memory, DDR/SDRAM, Burst Cellular RAM and ECC Controllers are all featured external Memory Controllers on the EBI.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

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 seven external devices, each assigned to seven 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 seven chip select lines (NCS[5:0] and SDCS) and several control pins that are generally multiplexed between the different external Memory Block Diagram

22.1.1 External Bus Interface

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

Figure 22-1. Organization of the External Bus Interface



22.2 I/O Lines Description

Table 22-1. EBI I/O Lines Description

Name	Function	Type	Active Level
	EBI		
D0 - D31	Data Bus	I/O	
A0 - A25	Address Bus	Output	
NWAIT	External Wait Signal	Input	Low
	SMC		
NCS0 - NCS5	Chip Select Lines	Output	Low
NWR0 - NWR3	Write Signals	Output	Low
NRD	Read Signal	Output	Low
NWE	Write Enable	Output	Low
NBS0 - NBS3	Byte Mask Signals	Output	Low
	EBI for CompactFlash Support		
CFCE1 - CFCE2	CompactFlash Chip Enable	Output	Low
CFOE	CompactFlash Output Enable	Output	Low
CFWE	CompactFlash Write Enable	Output	Low
CFIOR	CompactFlash I/O Read Signal	Output	Low
CFIOW	CompactFlash I/O Write Signal	Output	Low
CFRNW	CompactFlash Read Not Write Signal	Output	
CFCS0 - CFCS1	CompactFlash Chip Select Lines	Output	Low
	EBI for NAND Flash Support		
NANDCS	NAND Flash Chip Select Line	Output	Low
NANDOE	NAND Flash Output Enable	Output	Low
NANDWE	NAND Flash Write Enable	Output	Low
	DDR/SDRAM Controller		
SDCK	DDR/SDRAM Clock	Output	
SDCKN	DDR Inverted Clock	Output	
DQS0	DDR Data Qualifier Strobe 0	I/O	
DQS1	DDR Data Qualifier Strobe 1	I/O	
SDCKE	DDR/SDRAM Clock Enable	Output	High
SDCS	DDR/SDRAM Chip Select Line	Output	Low
BA0 - BA1	DDR/SDRAM Bank Select	Output	
SDWE	DDR/SDRAM Write Enable	Output	Low
RAS - CAS	DDR/SDRAM Row and Column Signal	Output	Low
NBS0 - NBS3	DDR/SDRAM Byte Mask Signals	Output	Low
SDA10	DDR/SDRAM Address 10 Line	Output	
	Burst CellularRAM Controller		

Table 22-1. EBI I/O Lines Description

Name	Function	Type	Active Level
BCCK	Burst CellularRAM Clock	Output	
BCCRE	Burst CellularRAM Clock Enable	Output	High
BCCS	Burst CellularRAM Chip Select Line	Output	Low
BCWE	Burst CellularRAM Write Enable	Output	Low
BCADV	Burst CellularRAM Burst Advance Signal	Output	Low
BCOE	Burst CellularRAM Output Enable	Output	Low
NBS0 - NBS1	Burst CellularRAM Byte Mask Signals	Output	Low
BCOWAIT	Burst CellularRAM Output Wait Signal	Input	

22.3 Application Example

22.3.1 Hardware Interface

Table 22-3, "EBI Pins and External Device Connections," details the connections to be applied between the EBI pins and the external devices for each Memory Controller.

Table 22-2. EBI Pins and External Static Devices Connections

Signals	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 - D15	D8 - D15
D16 - D23	–	–	–	D16 - D23	D16 - D23	D16 - D23
D24 - D31	–	–	–	D24 - D31	D24 - D31	D24 - D31
A0/NBS0	A0	–	NLB	–	NLB(3)	BE0(5)
A1/NWR2/NBS2	A1	A0	A0	WE(2)	NLB(4)	BE2(5)
A2 - 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/BCCS	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(1)	WE	WE(2)	WE	WE
NWR1/NBS1	–	WE(1)	NUB	WE(2)	NUB(3)	BE1(5)
NWR3/NBS3	–	–	–	WE(2)	NUB(4)	BE3(5)

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

Table 22-3. EBI Pins and External Device Connections

Signals	Pins of the Interfaced Device					
	SDRAM	Mobile DDR	Burst CellularRAM	CompactFlash	CompactFlash True IDE Mode	NAND Flash
Controller	**DDR/SDRAMC**	**DDR/SDRAMC**	**BCRAMC**	**SMC**		
D0 - D7	D0 - D7	D0 - D7	D0 - D7	D0 - D7	D0 - D7	AD0-AD7
D8 - D15	D8 - D15	D8 - D15	D8 - D15	D8 - D15	D8 - D15	AD8-AD15
D16 - D31	D16 - D31	–	–	–	–	–
A0/NBS0	DQM0	DQM0	DQM0	A0	A0	–
A1/NWR2/NBS2	DQM2	–	DQM2	A1	A1	–
A2 - A10	A[0:8]	A[0:8]	A[0:8]	A[2:10]	A[2:10]	–
A11	A9	A9	A9	–	–	–
SDA10	A10	A10	–	–	–	–
A12	–	–	A10	–	–	–
A13 - A14	A[11:12]	A[11:12]	A[11:12]	–	–	–
A15	–	A13	A13	–	–	–
A16/BA0	BA0	BA0	A14	–	–	–
A17/BA1	BA1	BA1	A15	–	–	–
A18 - A20	–	–	A[16:18]	–	–	–
A21	–	–	A19	–	–	ALE[3]
A22	–	–	A20	REG	REG	CLE[3]
A23 - A24	–	–	A[21:22]	–	–	–
A25	–	–	A23	CFRNW[1]	CFRNW[1]	–
NCS0	–	–	–	–	–	–
NCS1/BCCS	–	–	CS	–	–	–
NCS2	–	–	–	–	–	–
NCS3/NANDCS	–	–	–	–	–	CE[4]
NCS4/CFCS0	–	–	–	CFCS0[1]	CFCS0[1]	–
NCS5/CFCS1	–	–	–	CFCS1[1]	CFCS1[1]	–
SDCS	CS	CS	–	–	–	–
NANDOE	–	–	–	–	–	OE
NANDWE	–	–	–	–	–	WE
NRD/CFOE	–	–	–	OE	–	–
NWR0/NWE/CFWE	–	–	–	WE	WE	–
NWR1/NBS1/CFIOR	DQM1	DQM1	DQM1	IOR	IOR	–
NWR3/NBS3/CFIOW	DQM3	–	DQM3	IOW	IOW	–
CFCE1	–	–	–	CE1	CS0	–
CFCE2	–	–	–	CE2	CS1	–
BCCK	–	–	CLK	–	–	–

Table 22-3. EBI Pins and External Device Connections (Continued)

Signals	Pins of the Interfaced Device					
	SDRAM	Mobile DDR	Burst CellularRAM	CompactFlash	CompactFlash True IDE Mode	NAND Flash
Controller	**DDR/SDRAMC**	**DDR/SDRAMC**	**BCRAMC**	**SMC**		
SDCK	CLK	CLK	–	–	–	–
SDCKN	–	CLKN	–	–	–	–
SDCKE/BCCRE	CKE	CKE	CRE	–	–	–
DQS0 - DQS1	–	DQS0 - DQS1	–	–	–	–
RAS/BCADV	RAS	RAS	ADV	–	–	–
CAS/BCOE	CAS	CAS	OE	–	–	–
SDWE/BCWE	WE	WE	WE	–	–	–
BCOWAIT	–	–	OWAIT	–	–	–
NWAIT	–	–	–	WAIT	WAIT	–
Pxx(2)	–	–	–	CD1 or CD2	CD1 or CD2	–
Pxx(2)	–	–	–	–	–	CE(4)
Pxx(2)	–	–	–	–	–	RDY

Note:
1. Not directly connected to the CompactFlash slot. Permits the control of the bidirectional buffer between the EBI data bus and the CompactFlash slot.
2. Any PIO line.
3. The CLE and ALE signals of the NAND Flash device may be driven by any address bit. For details, see "NAND Flash Support" on page 167.
4. NAND CE may be connected to any PIO line if CE don't care mode is not supported by NAND device. Otherwise NANDCS may be used.

22.4 Product Dependencies

22.4.1 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.

22.5 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 DDR/SDRAM Controller (DDR/SDRAMC)
- the Burst Cellular RAM Controller (BCRAMC)
- 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

22.5.1 Bus Multiplexing

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

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

22.5.2 Pull-up Control

The EBI_CSA registers 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 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.

22.5.3 Supply Control

The EBI I/O pads may be supplied with two different voltage (3.3V or 1.8V). The EBI_CSA registers in the Chip Configuration User Interface allows to choose between the two different voltages. At power-up, the selected supply control is 3.3V, allowing the EBI to work at low system speed even supplied at 1.8V.

22.5.4 Static Memory Controller

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

22.5.5 DDR/SDRAM Controller

For information on the DDR/SDRAM Controller, refer to the section “DDR/SDRAM”.

22.5.6 BCRAM Controller

For information on the BCRAM Controller, refer to the section “BCRAM”.

22.5.7 ECC Controller

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

22.5.8 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 section “Bus Matrix”. Access to an external CompactFlash device is then made by accessing the address space reserved for NCS4 and/or NCS5 (i.e., between 0x5000 0000 and 0x5FFF FFFF for NCS4 and between 0x6000 0000 and 0x6FFF FFFF for NCS5).

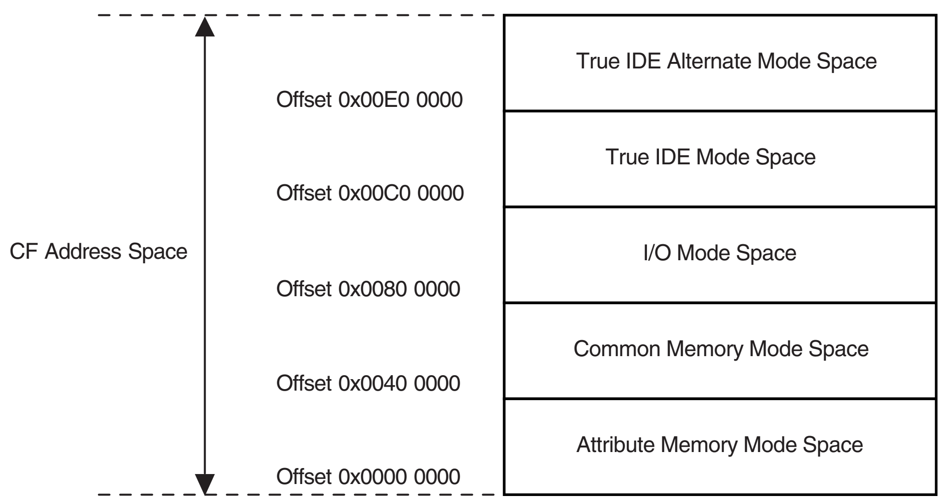
All CompactFlash modes (Attribute Memory, Common Memory, I/O and True IDE) are supported but the signals _IOIS16 (I/O and True IDE modes) and _ATA SEL (True IDE mode) are not handled.

22.5.8.1 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 22-2. A[23:21] bits of the transfer address are used to select the desired mode as described in Table 22-4 on page 164.

Figure 22-2. CompactFlash Memory Mapping



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

Table 22-4. 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

22.5.8.2 *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 22-5 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 section "Static Memory Controller".

Table 22-5. 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	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]	
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	–	–	–

22.5.8.3 Read/Write Signals

In I/O mode and True IDE mode, the CompactFlash logic drives the read and write command signals of the SMC on CFIOR and CFIOW signals, while the CFOE and CFWE signals are deactivated. Likewise, in common memory mode and attribute memory mode, the SMC signals are driven on the CFOE and CFWE signals, while the CFIOR and CFIOW are deactivated. Figure 22-3 on page 165 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” in “Static Memory Controller (SMC)”.

Figure 22-3. CompactFlash Read/Write Control Signals

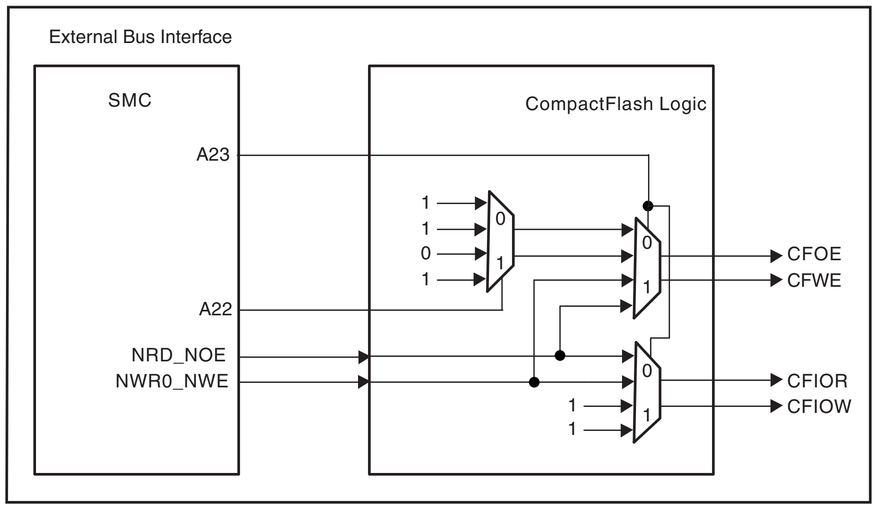


Table 22-6. CompactFlash Mode Selection

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

22.5.8.4 Multiplexing of CompactFlash Signals on EBI Pins

Table 22-7 on page 166 and Table 22-8 on page 166 illustrate the multiplexing of the CompactFlash logic signals with other EBI signals on the EBI pins. The EBI pins in Table 22-7 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 22-8 on page 166 remain shared between all memory areas when the corresponding CompactFlash interface is enabled (EBI_CS4A = 1 and/or EBI_CS5A = 1).

Table 22-7. Dedicated CompactFlash Interface Multiplexing

Pins	CompactFlash Signals		EBI Signals	
	EBI_CS4A = 1	**EBI_CS5A = 1**	**EBI_CS4A = 0**	**EBI_CS5A = 0**
NCS4/CFCS0	CFCS0		NCS4	
NCS5/CFCS1		CFCS1		NCS5

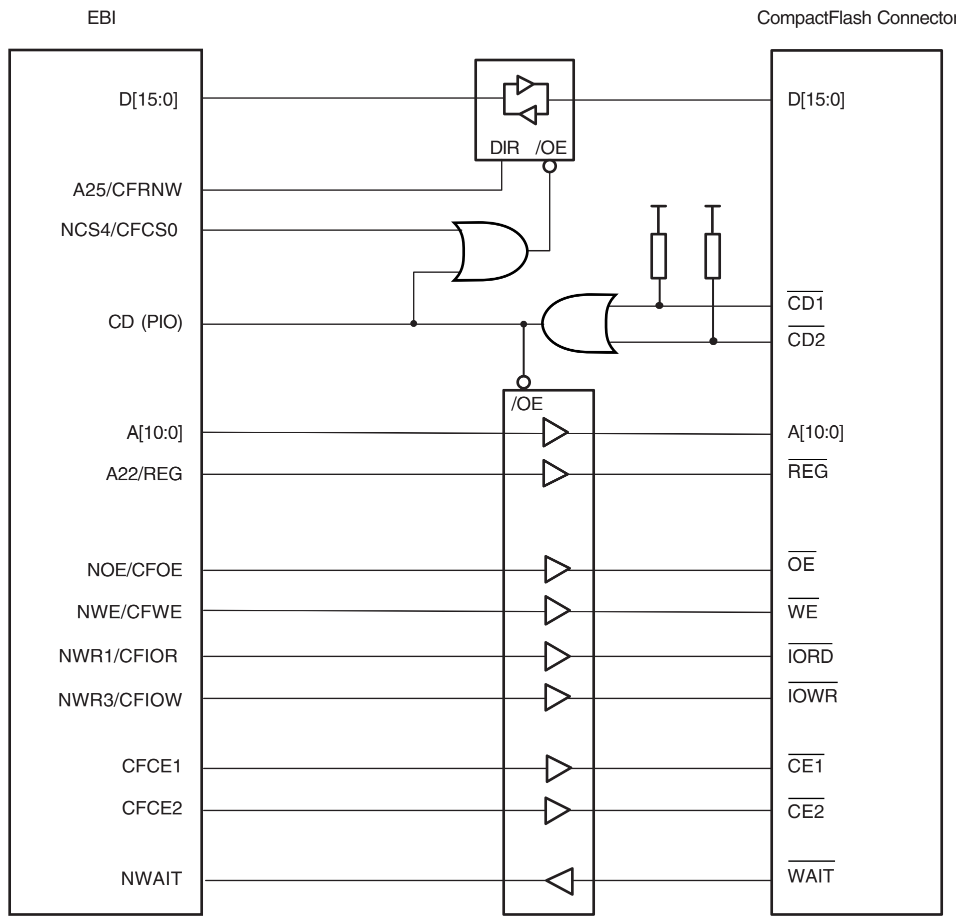
Table 22-8. Shared CompactFlash Interface Multiplexing

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

22.5.8.5 Application Example

Figure 22-4 on page 167 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 section “Static Memory Controller (SMC)”.

Figure 22-4. CompactFlash Application Example



22.5.9 NAND Flash Support

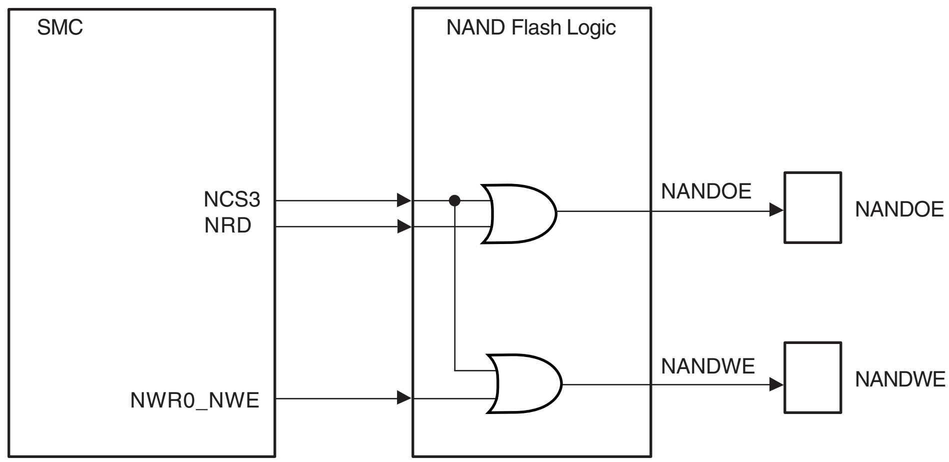
External Bus Interface integrates circuitry that interfaces to NAND Flash devices.

22.5.9.1 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 Section 21. “Bus Matrix”. 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 22-5 on page 168 for more information. For details on these waveforms, refer to the section “Static Memory Controller”.

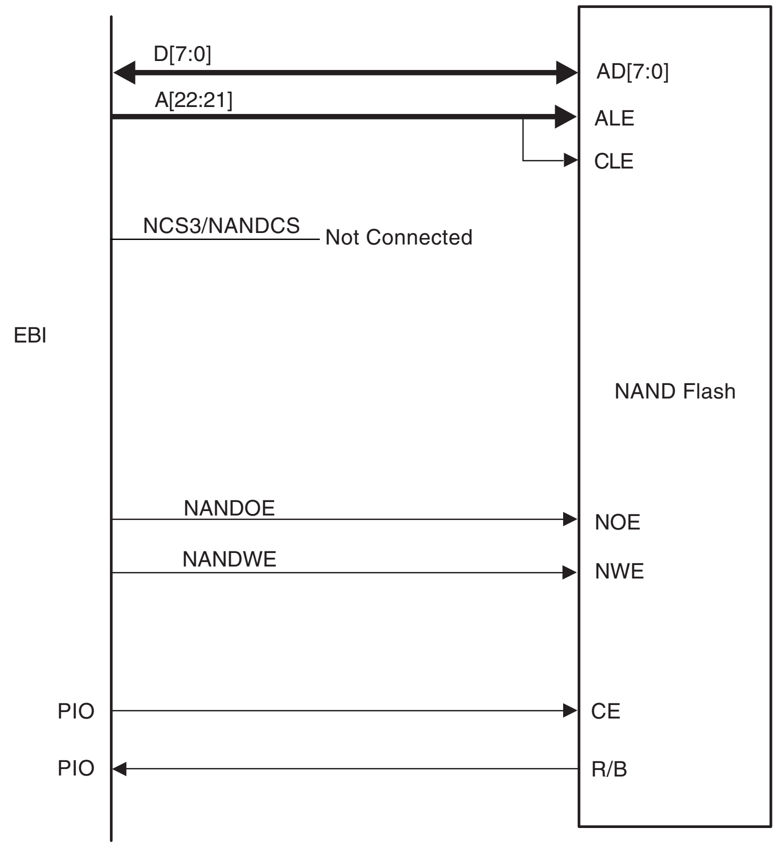
Figure 22-5. NAND Flash Signal Multiplexing on EBI Pins



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

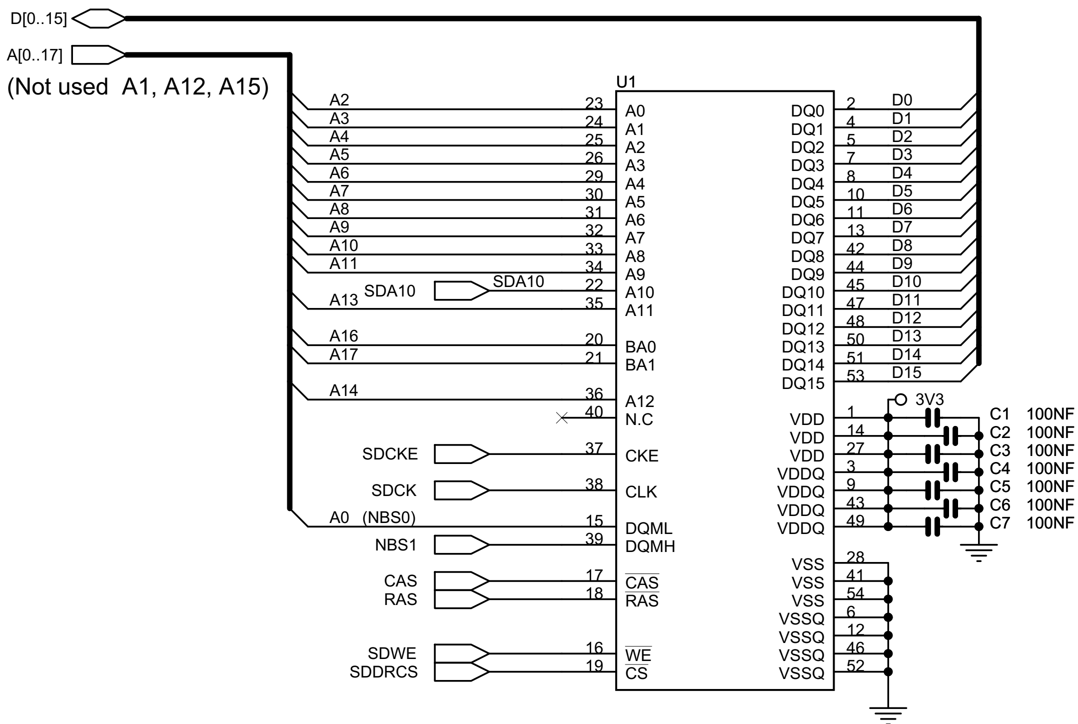
Figure 22-6. NAND Flash Application Example



22.6 Implementation Examples

22.6.1 16-bit SDRAM

Figure 22-7. Hardware Configuration



22.6.1.1 *Software Configuration*

The following configuration has to be performed:

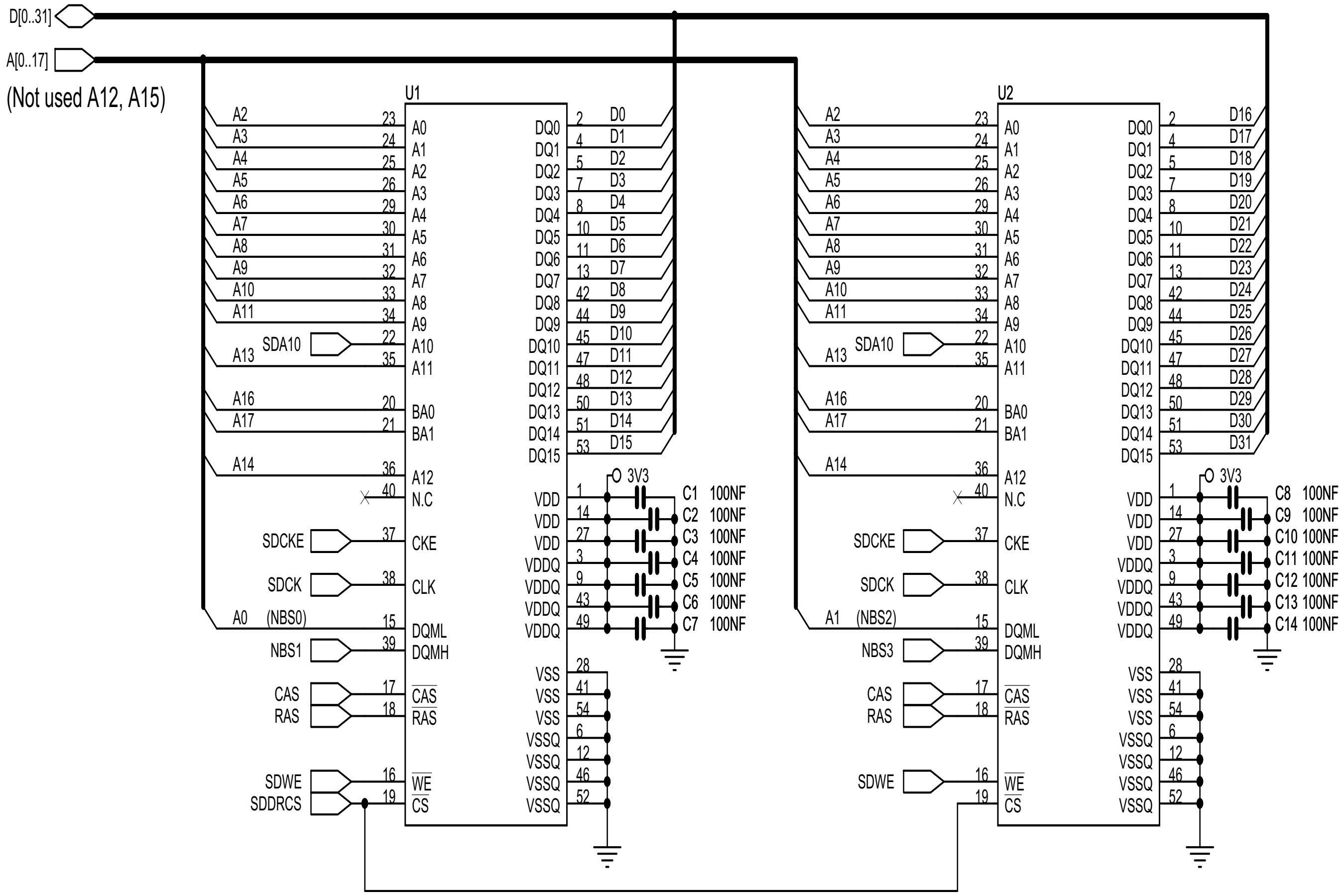
- Assign the EBI CS1 to the SDRAM controller by setting the bit EBI_CS1A in the EBI Chip Select Assignment Register located in the bus matrix memory space.
- Initialize the SDRAM Controller depending on the SDRAM device and system bus frequency.

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

The SDRAM initialization sequence is described in the “SDRAM device initialization” part of the SDRAM controller.

22.6.2 32-bit SDRAM

22.6.2.1 Hardware Configuration



22.6.2.2 Software Configuration

The following configuration has to be performed:

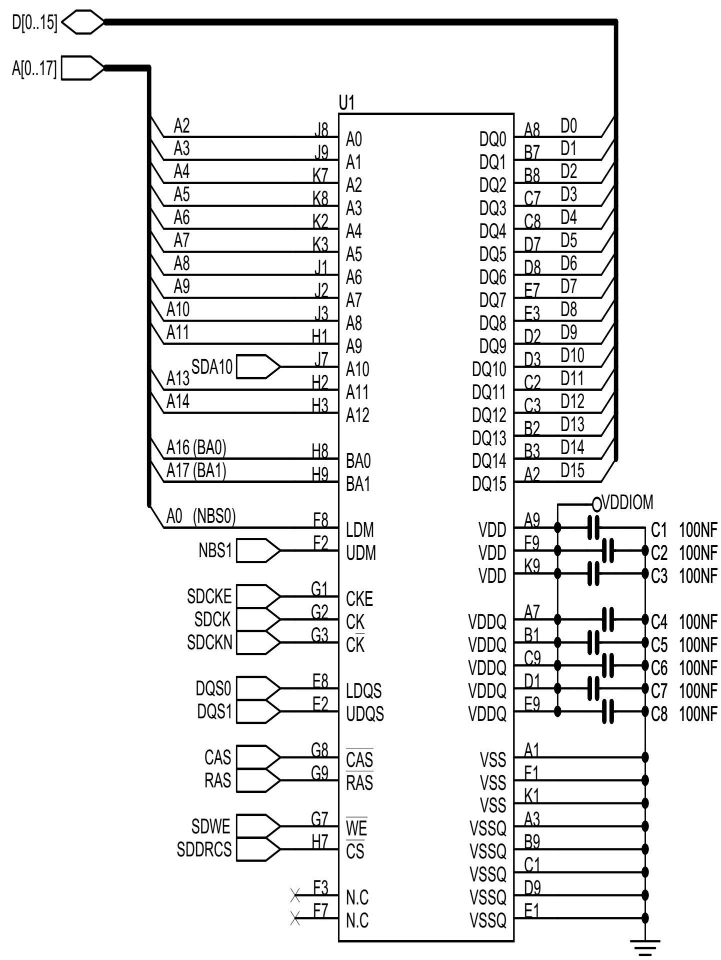
- Assign the EBI CS1 to the SDRAM controller by setting the bit EBI_CS1A in the EBI Chip Select Assignment Register located in the bus matrix memory space.
- Initialize the SDRAM Controller depending on the SDRAM device and system bus frequency.

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 "SDR-SDRAM device initialization" part of the DDRSDR-SDRAM controller.

22.6.3 16-bit Mobile DDR

Figure 22-8. Hardware Configuration



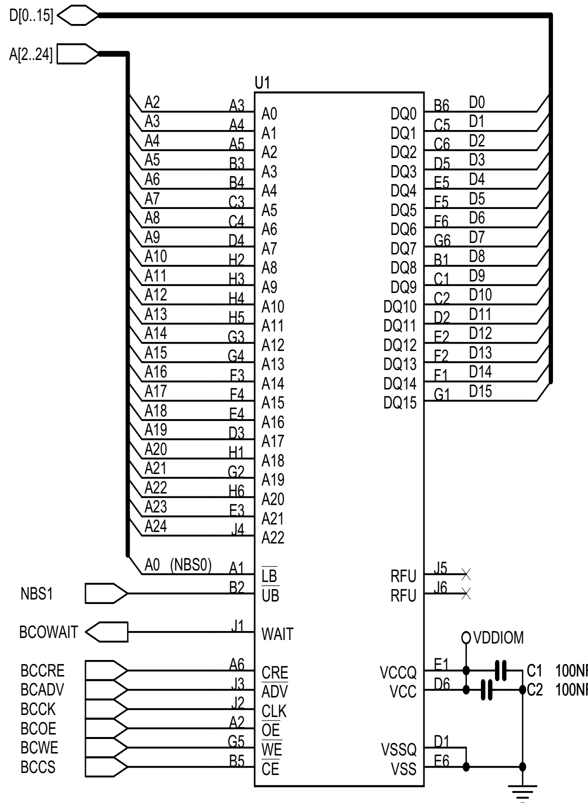
22.6.3.1 Software Configuration

The following configuration has to be performed:

- Initialize the SDRAM Controller depending on the DDR device and system bus frequency.
- The Data Bus Width is to be programmed to 16 bits.
- The SDRAM initialization sequence is described in the “DDR-SDRAM device initialization” part of the DDRSDR-SDRAM controller.

22.6.4 16-bit BCRAM

Figure 22-9. Hardware Configuration



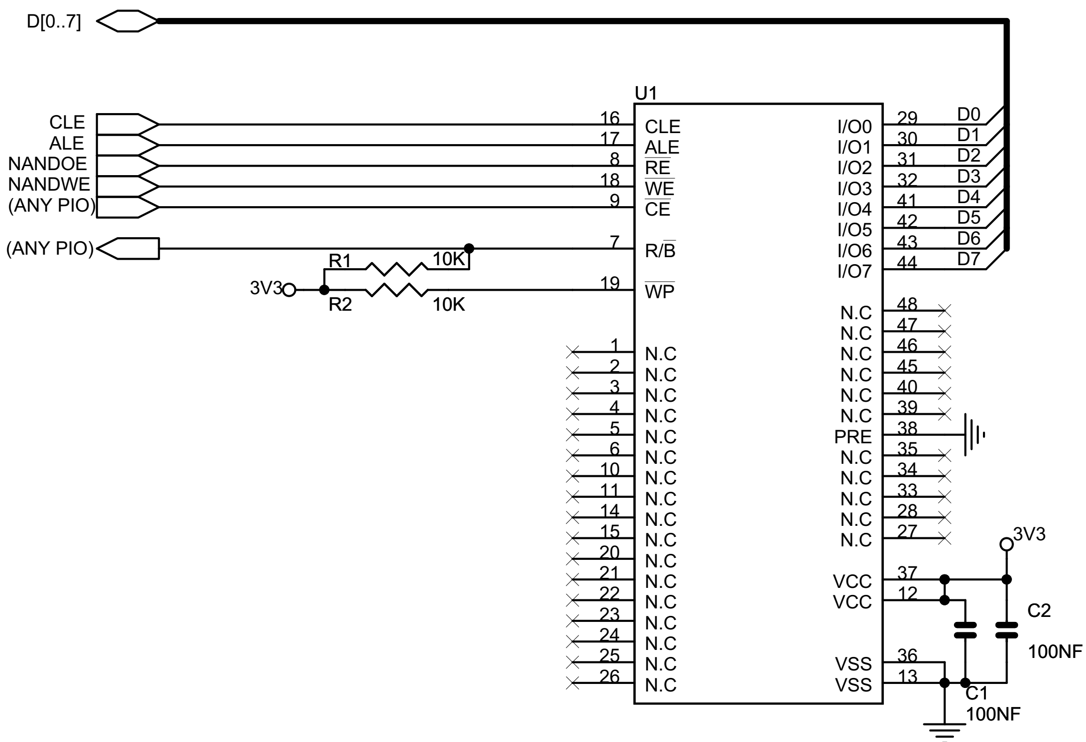
22.6.4.1 Software Configuration

The following configuration has to be performed:

- Assign the EBI CS1 to the BCRAM by setting the bit EBI_CS1A in the EBI Chip Select Assignment Register located in the bus matrix memory space
- The Data Bus Width is to be programmed to 16 bits.
- The BCRAM initialization sequence is described in the “Cellular Ram initialization” part of the BCRAM controller.

22.6.5 8-bit NAND Flash

22.6.5.1 Hardware Configuration



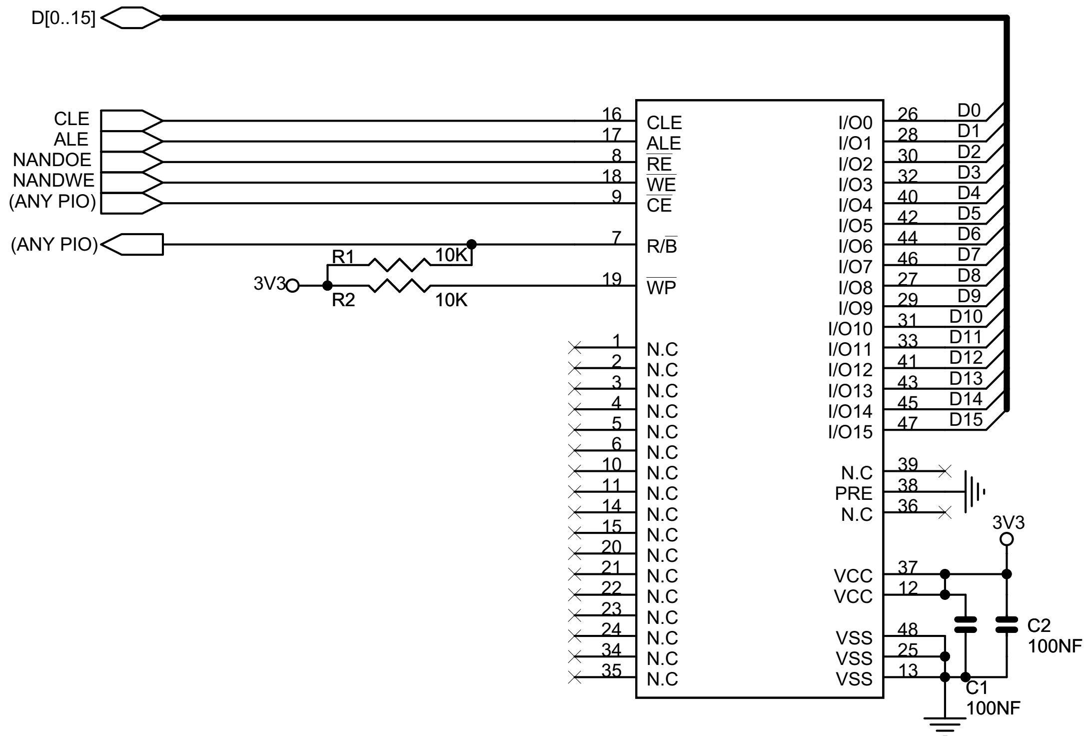
22.6.5.2 Software Configuration

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.
- 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.

22.6.6 16-bit NAND Flash

22.6.6.1 Hardware Configuration

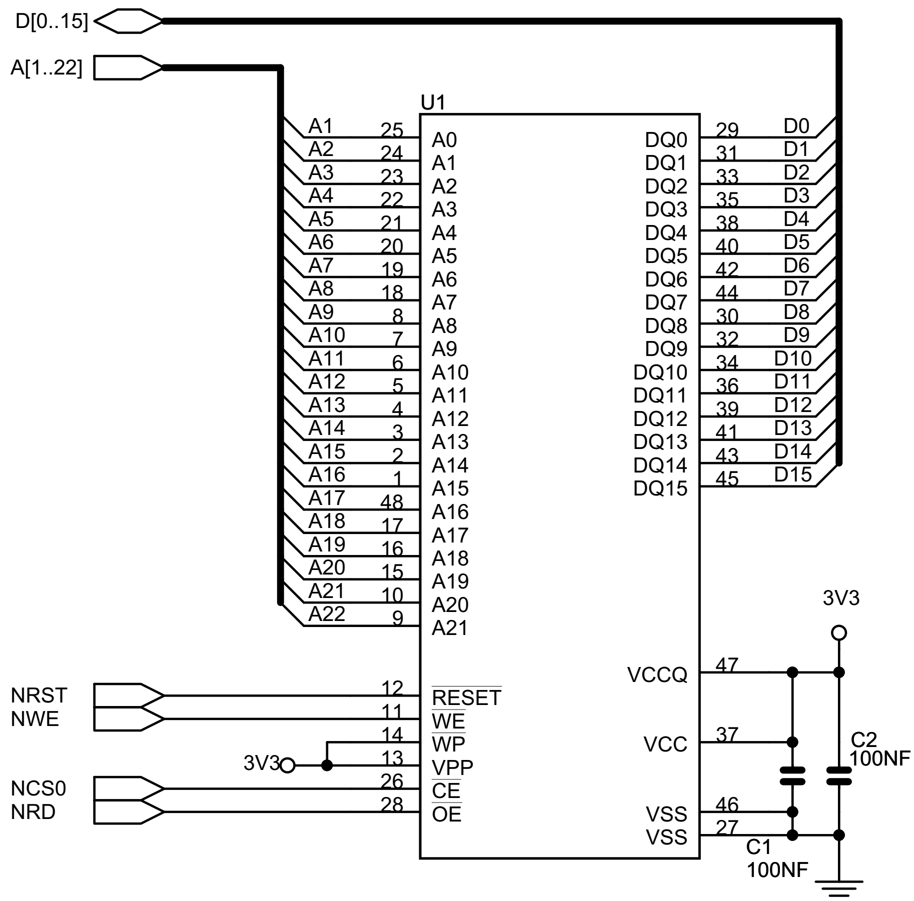


22.6.6.2 Software Configuration

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.

22.6.7 NOR Flash on NCS0

22.6.7.1 Hardware Configuration



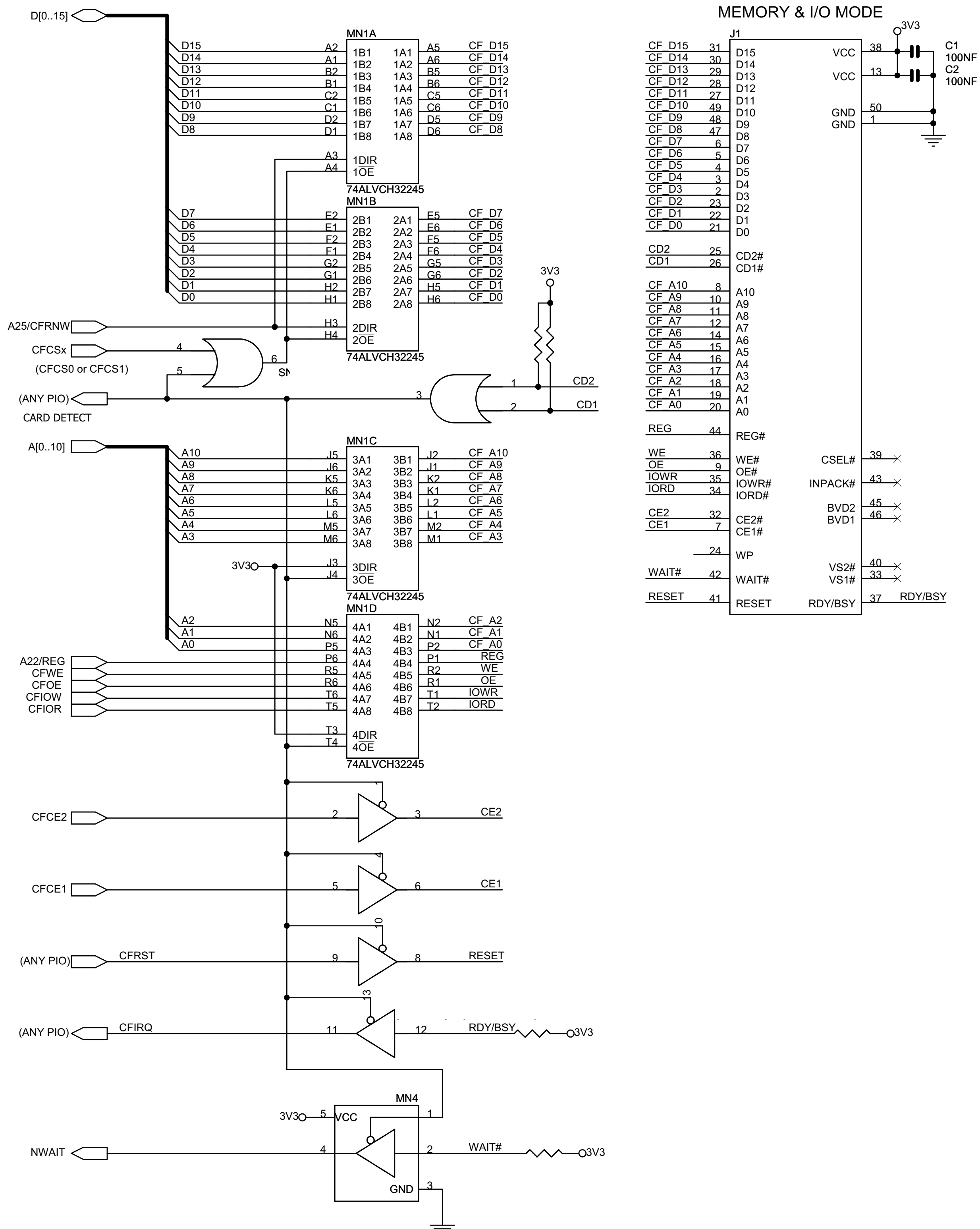
22.6.7.2 Software Configuration

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.

22.6.8 Compact Flash

22.6.8.1 Hardware Configuration



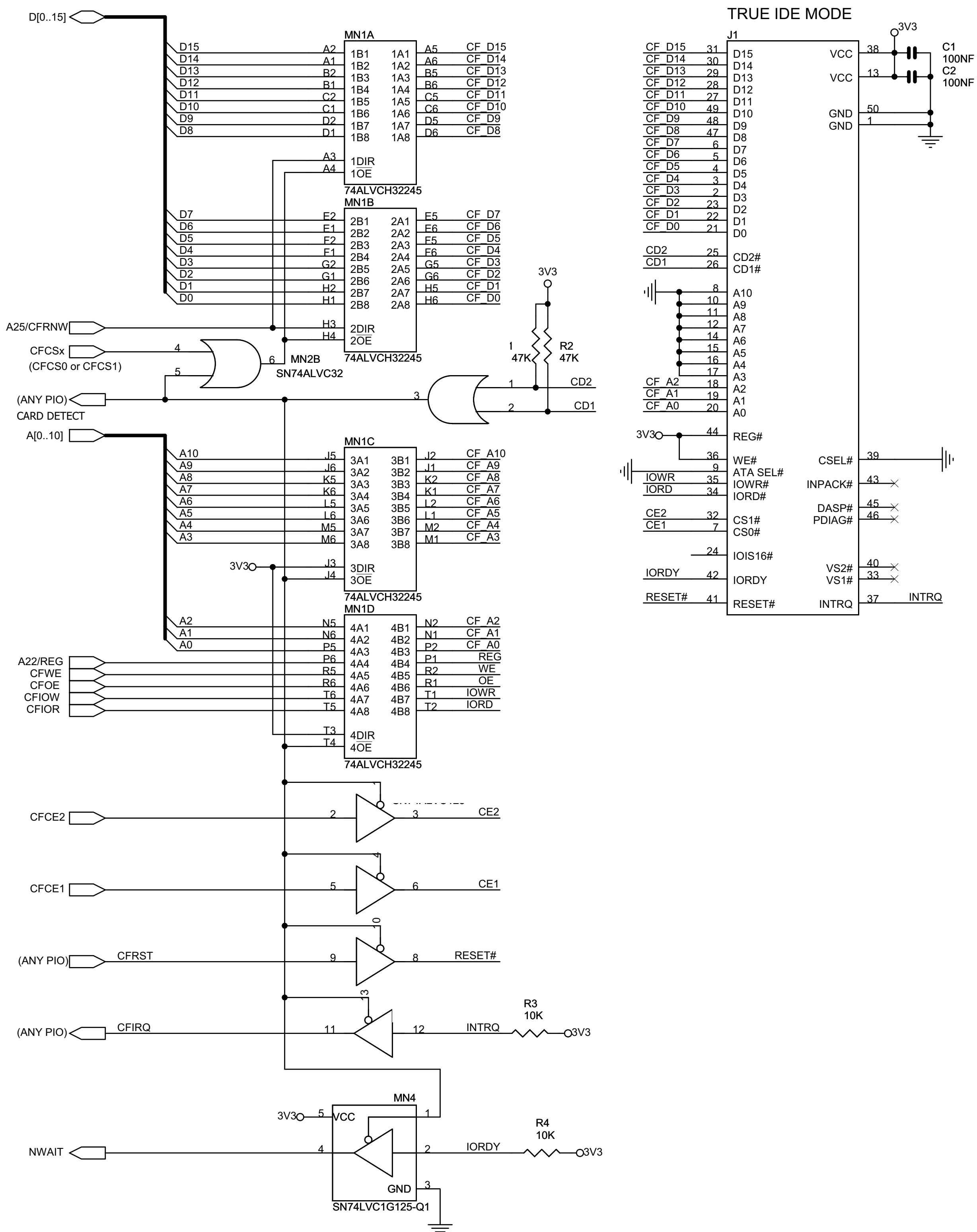
22.6.8.2 Software Configuration

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.

22.6.9 Compact Flash True IDE

22.6.9.1 Hardware Configuration



22.6.9.2 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

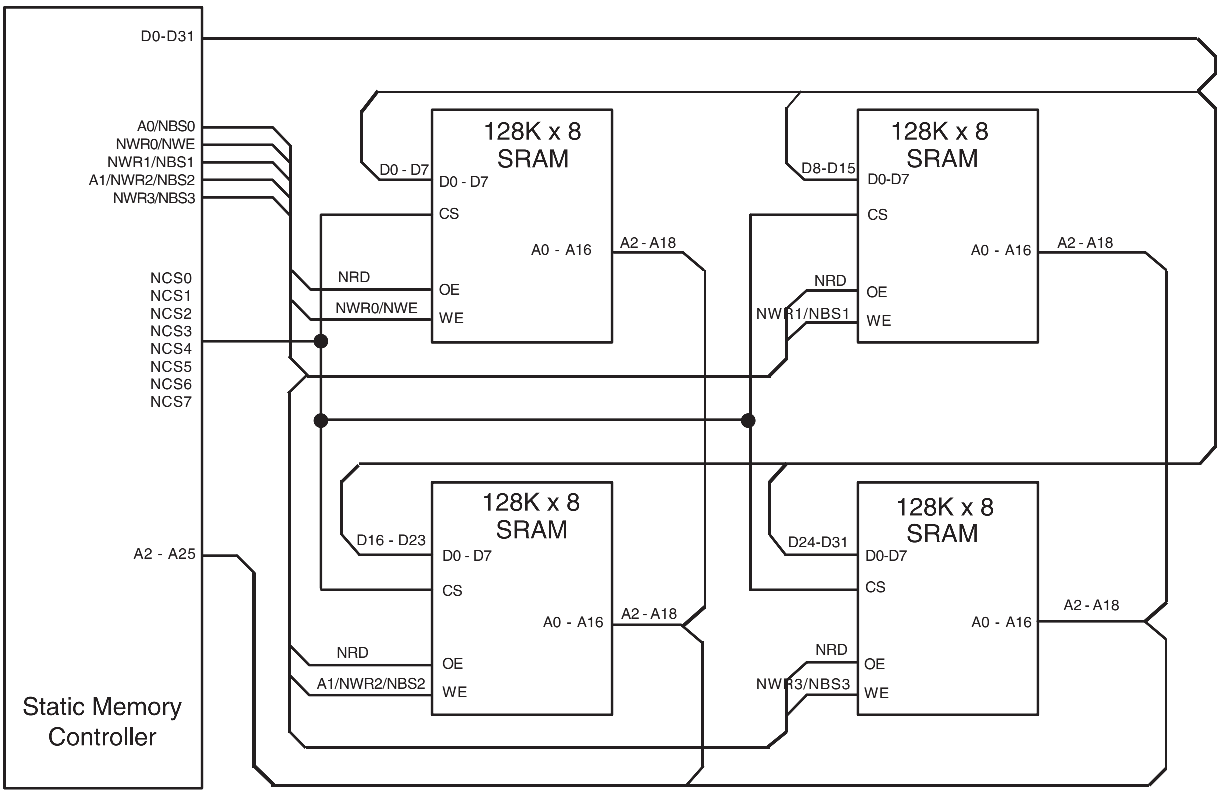
Table 23-2. Static Memory Controller (SMC) Multiplexed Signals

Multiplexed Signals			Related Function
NWR0	NWE		Byte-write or byte-select access, see “Byte Write or Byte Select Access” on page 183
A0	NBS0		8-bit or 16-/32-bit data bus, see “Data Bus Width” on page 183
NWR1	NBS1		Byte-write or byte-select access see “Byte Write or Byte Select Access” on page 183
A1	NWR2	NBS2	8-/16-bit or 32-bit data bus, see “Data Bus Width” on page 183. Byte-write or byte-select access, see “Byte Write or Byte Select Access” on page 183
NWR3	NBS3		Byte-write or byte-select access see “Byte Write or Byte Select Access” on page 183

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.

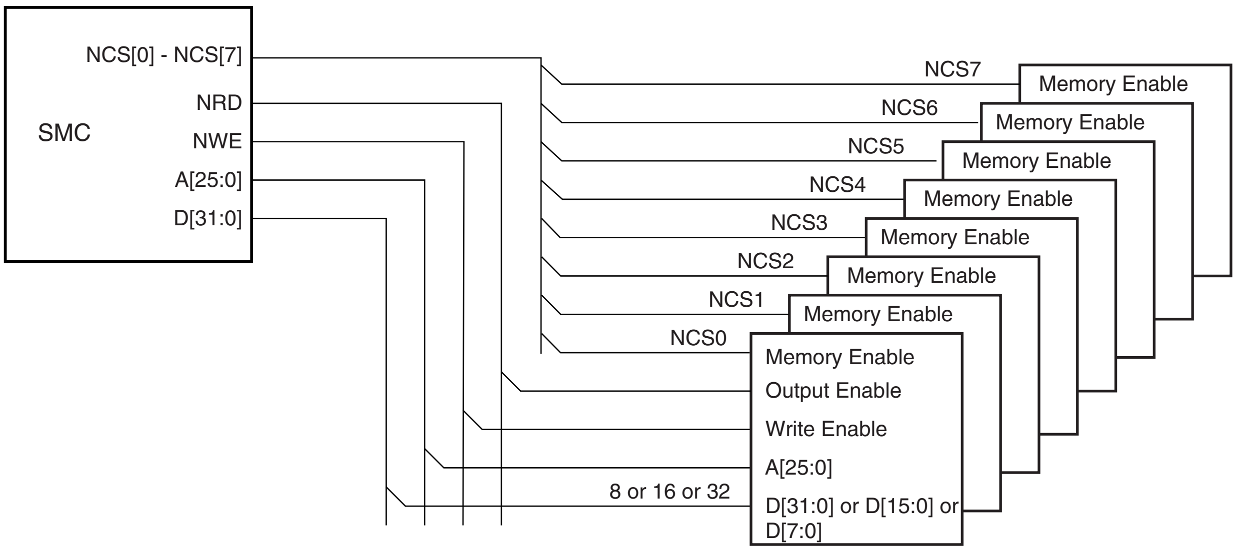
23.6 External Memory Mapping

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

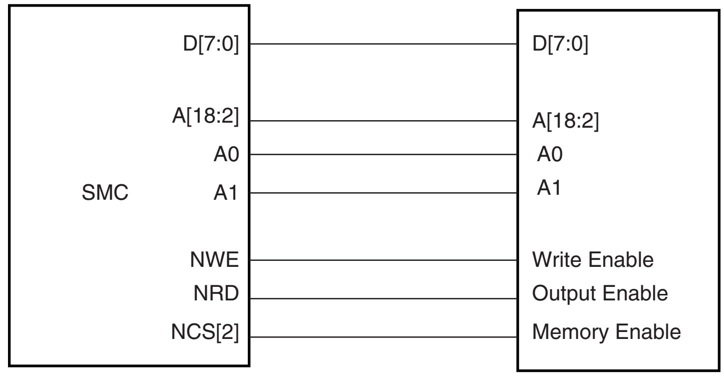


Figure 23-4. Memory Connection for a 16-bit Data Bus

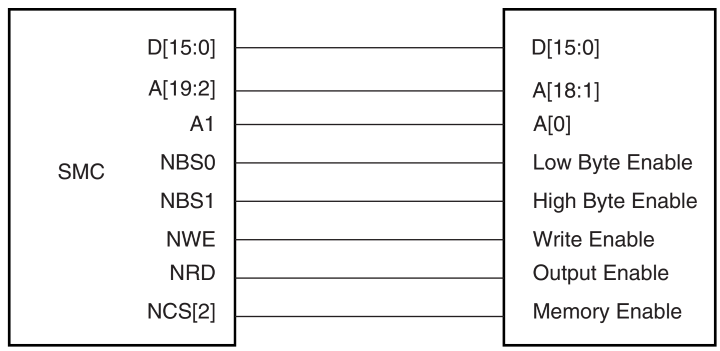
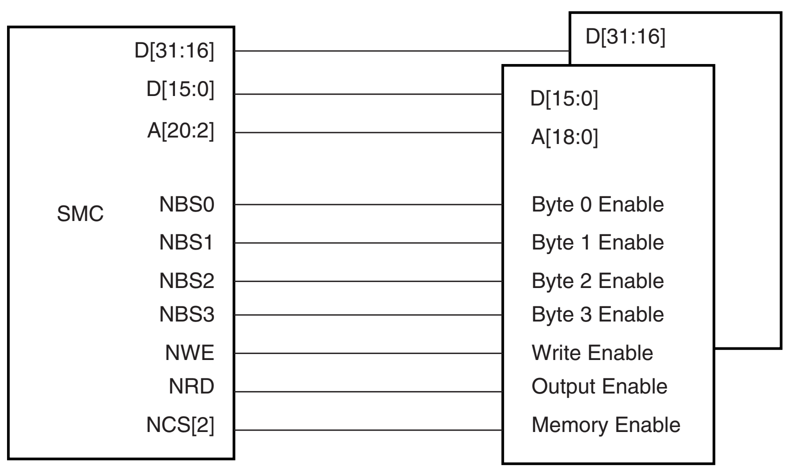


Figure 23-5. Memory Connection for a 32-bit Data Bus



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.

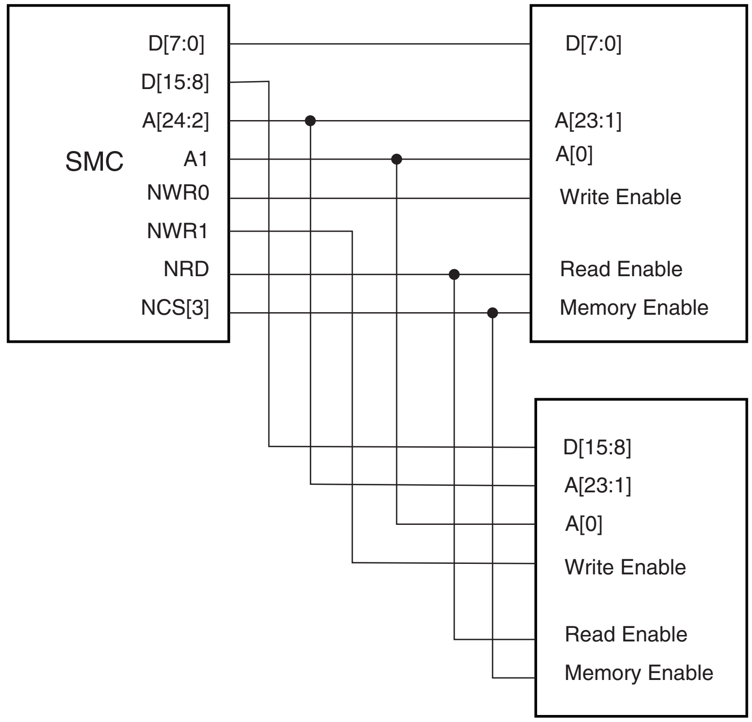
- For 16-bit devices: the SMC provides NBS0 and NBS1 selection signals for respectively byte0 (lower byte) and byte1 (upper byte) of a 16-bit bus.

  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)

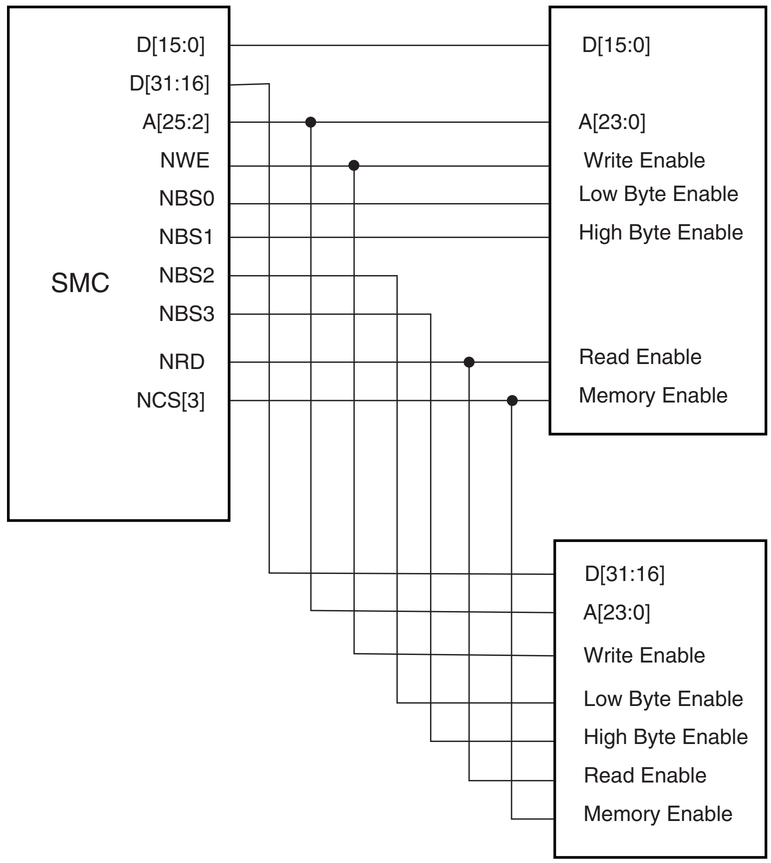


Table 23-3. SMC Multiplexed Signal Translation

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

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.

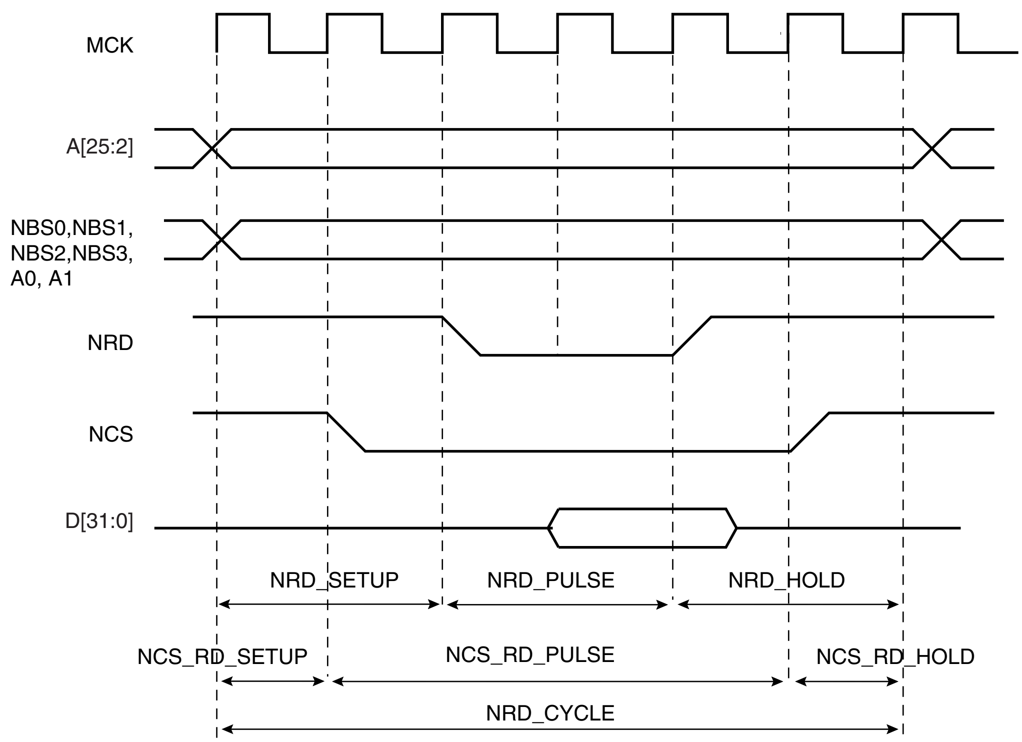
The read cycle starts with the address setting on the memory address bus, i.e.:

{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:

NRD_CYCLE = NRD_SETUP + NRD_PULSE + NRD_HOLD

= NCS_RD_SETUP + NCS_RD_PULSE + NCS_RD_HOLD

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

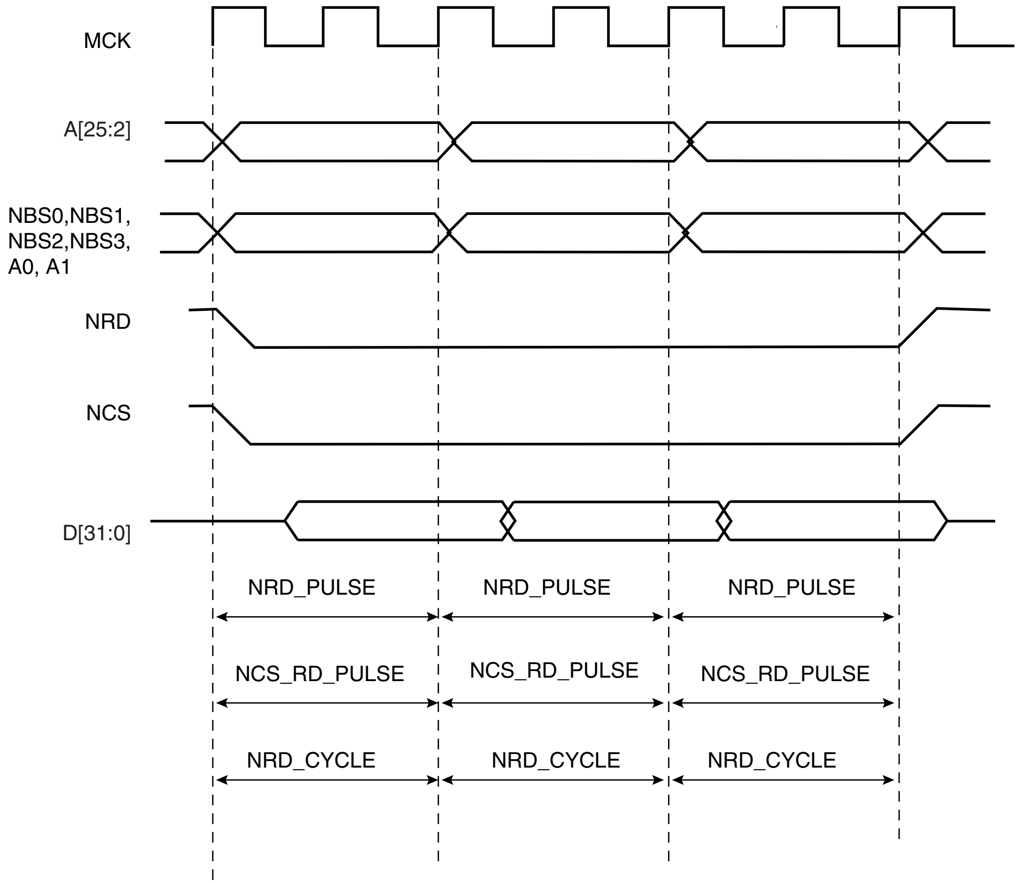
NRD_HOLD = NRD_CYCLE - NRD SETUP - NRD PULSE

NCS_RD_HOLD = NRD_CYCLE - NCS_RD_SETUP - NCS_RD_PULSE

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.

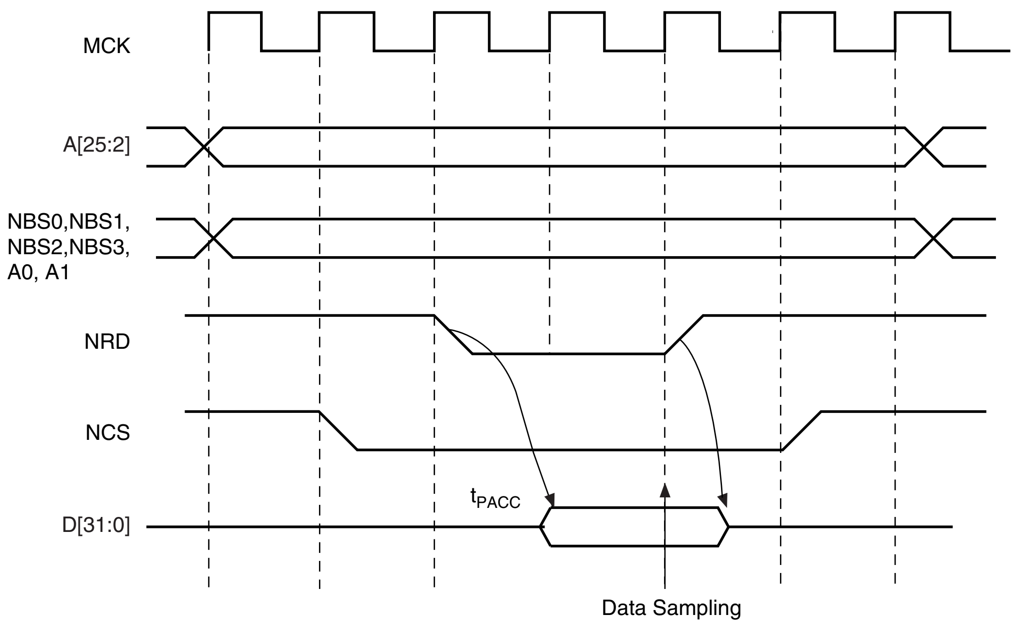
23.8.2 Read Mode

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.

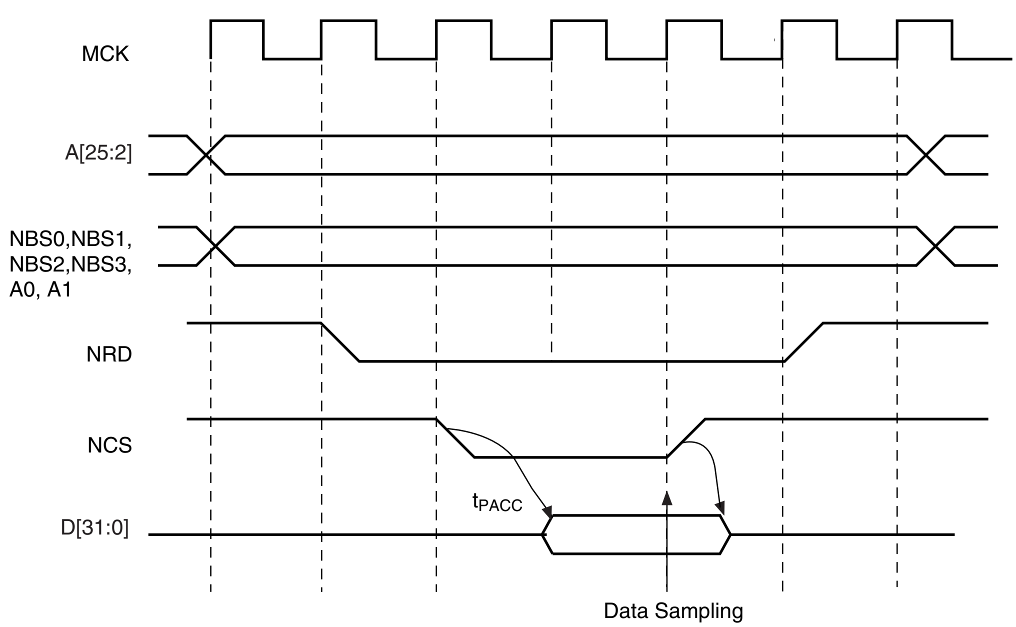
Figure 23-10. READ_MODE = 1: Data is sampled by SMC before the rising edge of NRD



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.

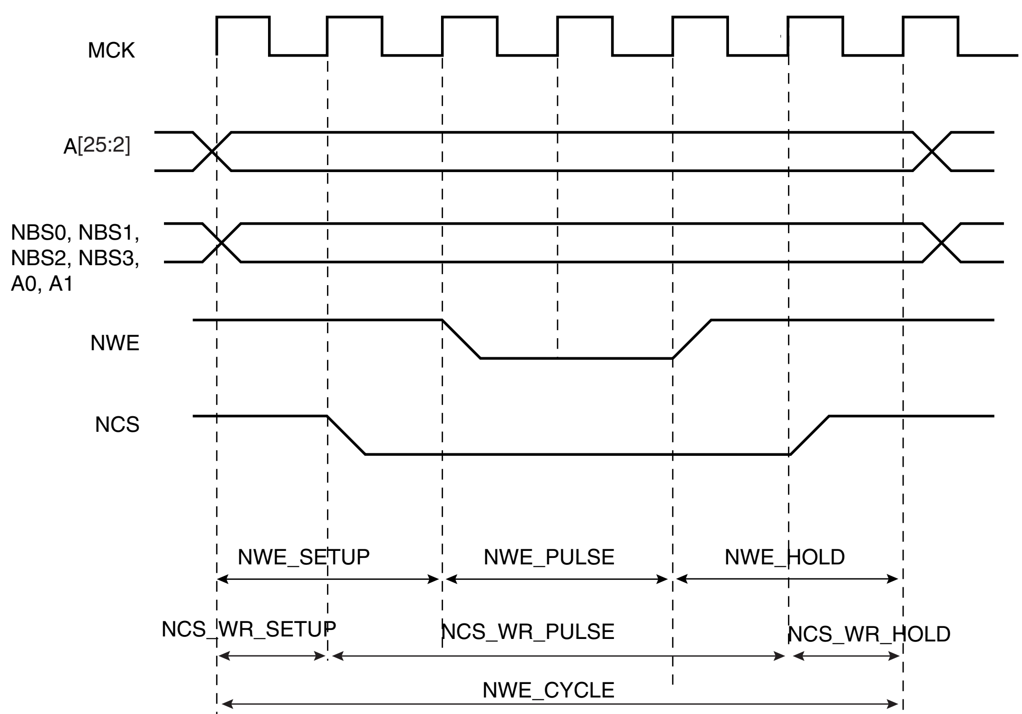
The NWE waveforms apply to all byte-write lines in Byte Write access mode: NWR0 to NWR3.

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:

NWE_CYCLE = NWE_SETUP + NWE_PULSE + NWE_HOLD

= NCS_WR_SETUP + NCS_WR_PULSE + NCS_WR_HOLD

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

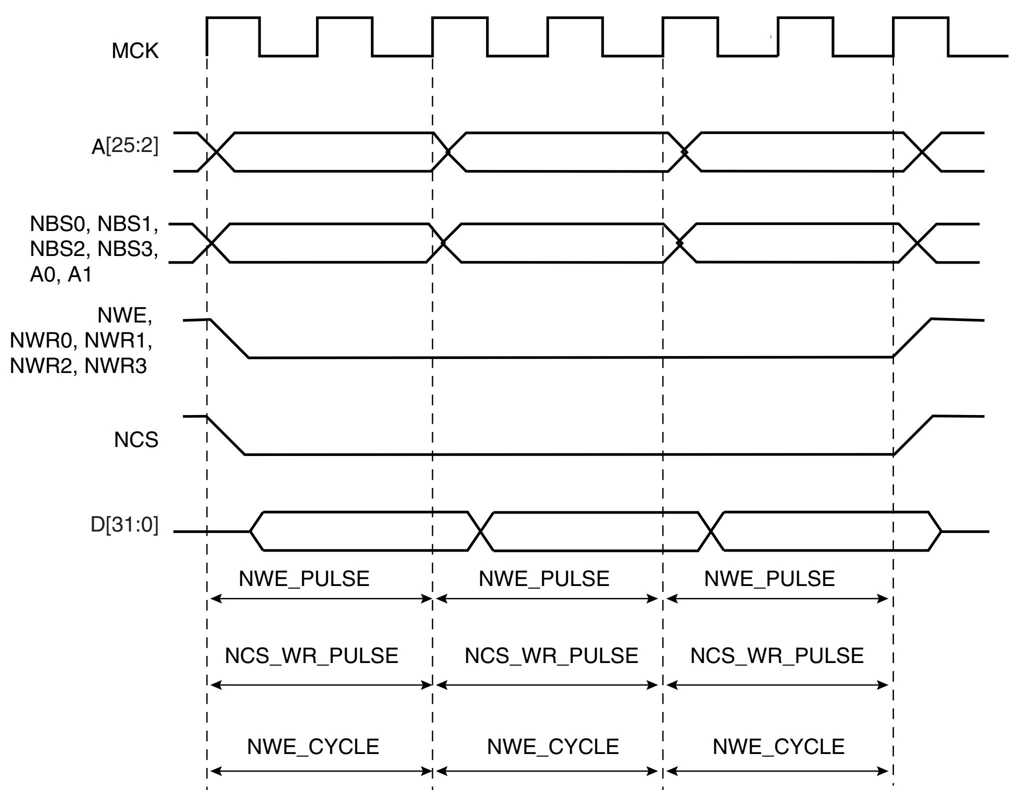
NWE_HOLD = NWE_CYCLE - NWE_SETUP - NWE_PULSE

NCS_WR_HOLD = NWE_CYCLE - NCS_WR_SETUP - NCS_WR_PULSE

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.

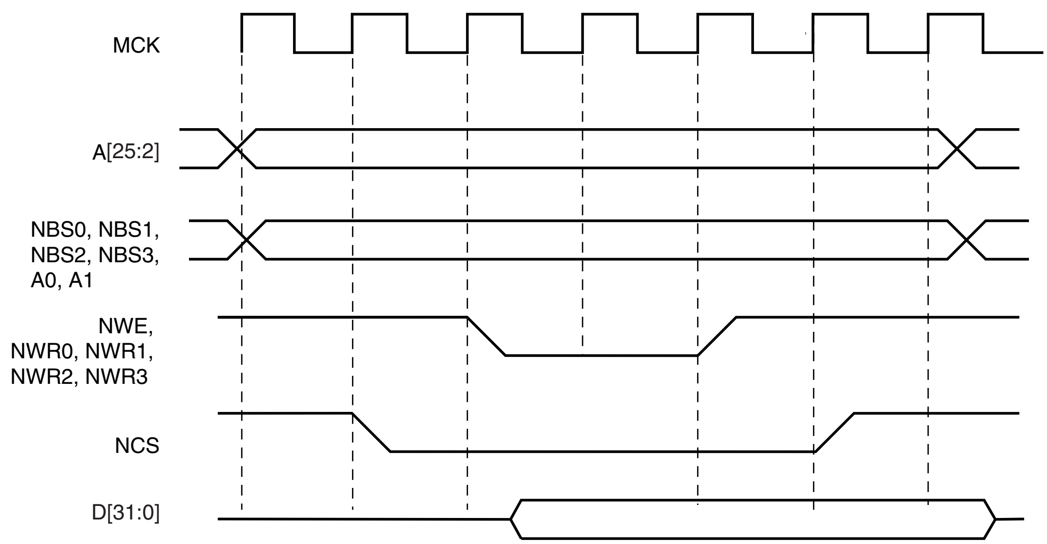
23.8.4 Write Mode

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.

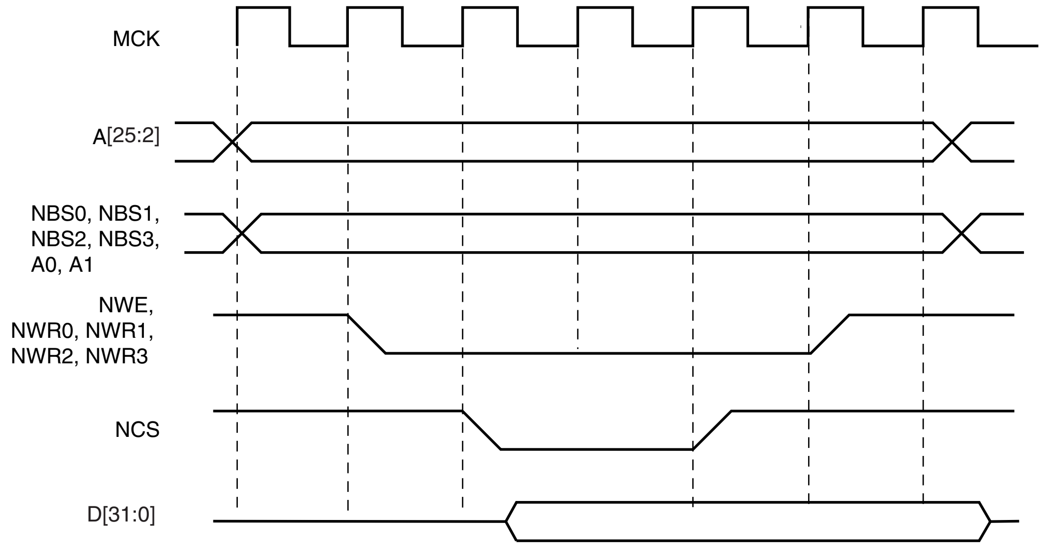
Figure 23-14. WRITE_MODE = 1. The write operation is controlled by NWE



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 x setup[5] + setup[4:0]	0 ≤≤31	0 ≤≤128+31
pulse [6:0]	7	256 x pulse[6] + pulse[5:0]	0 ≤≤63	0 ≤≤256+63
cycle [8:0]	9	256 x cycle[8:7] + cycle[6:0]	0 ≤≤127	0 ≤≤256+127 0 ≤≤512+127 0 ≤≤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	0x01010101	All setup timings are set to 1
SMC_PULSE	0x01010101	All pulse timings are set to 1
SMC_CYCLE	0x00030003	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 theses signals through external logic and pads. If positive setup and hold values must be verified, then it is strictly recommended to program non-null values so as to cover possible skews between address, NCS and NRD signals.

For write operations:

If a null hold value is programmed on NWE, the SMC can guarantee a positive hold of address, byte select lines, and NCS signal after the rising edge of NWE. This is true for WRITE_MODE = 1 only. See "Early Read Wait State" on page 197.

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.

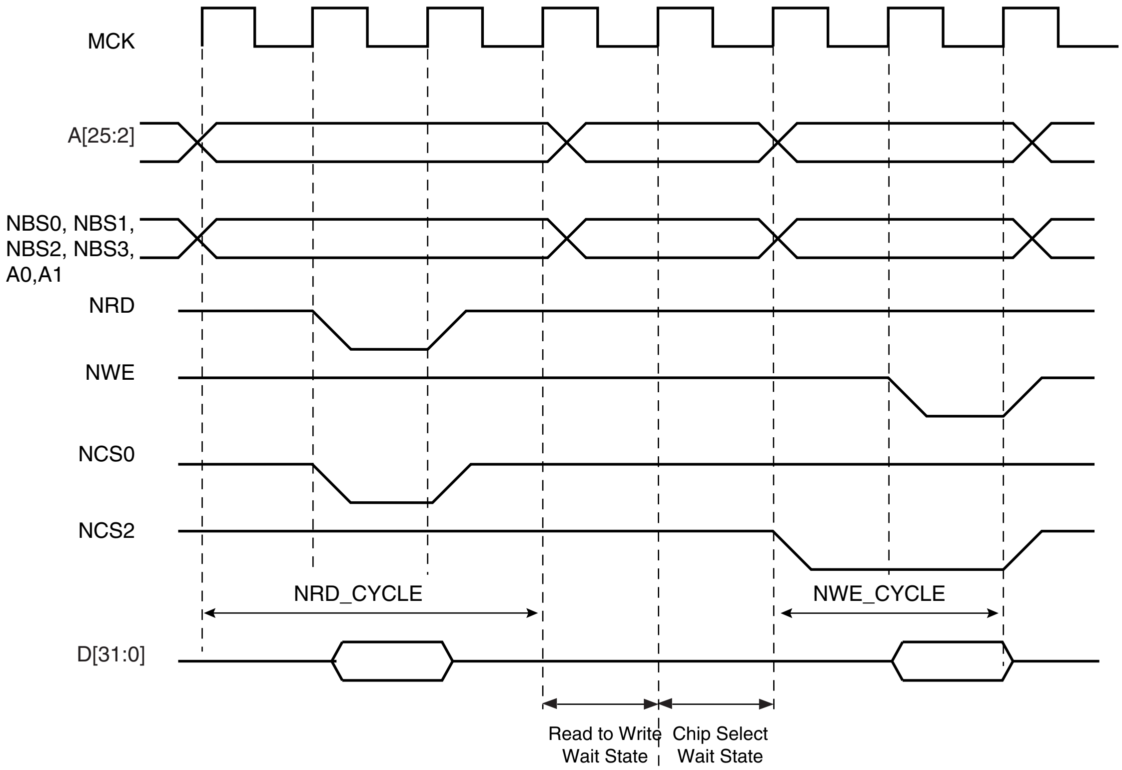
23.9.1 Chip Select Wait States

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

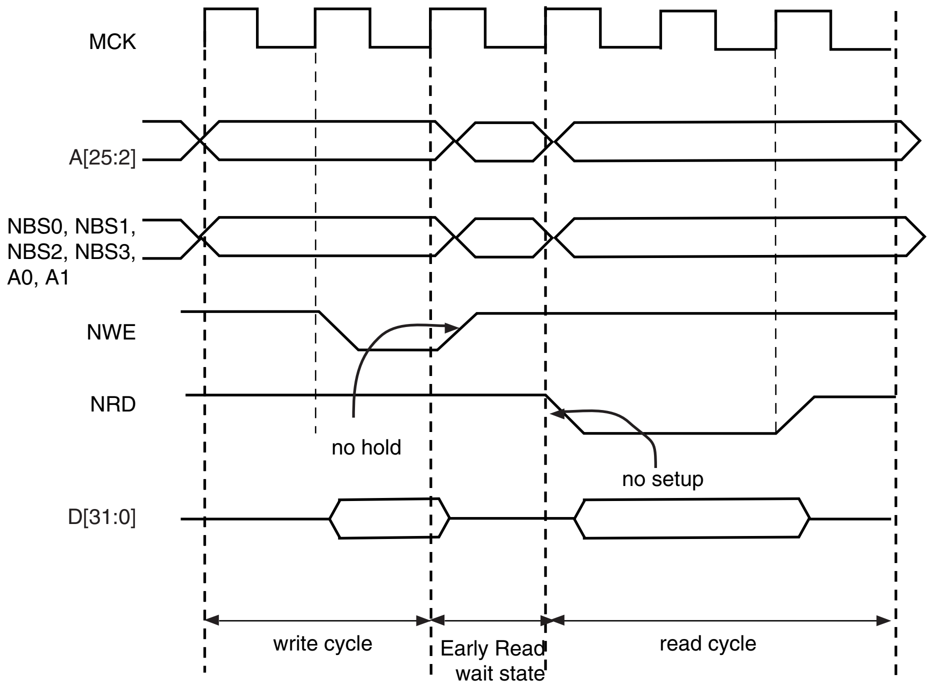


Figure 23-18. Early Read Wait State: NCS Controlled Write with No Hold Followed by a Read with No NCS Setup

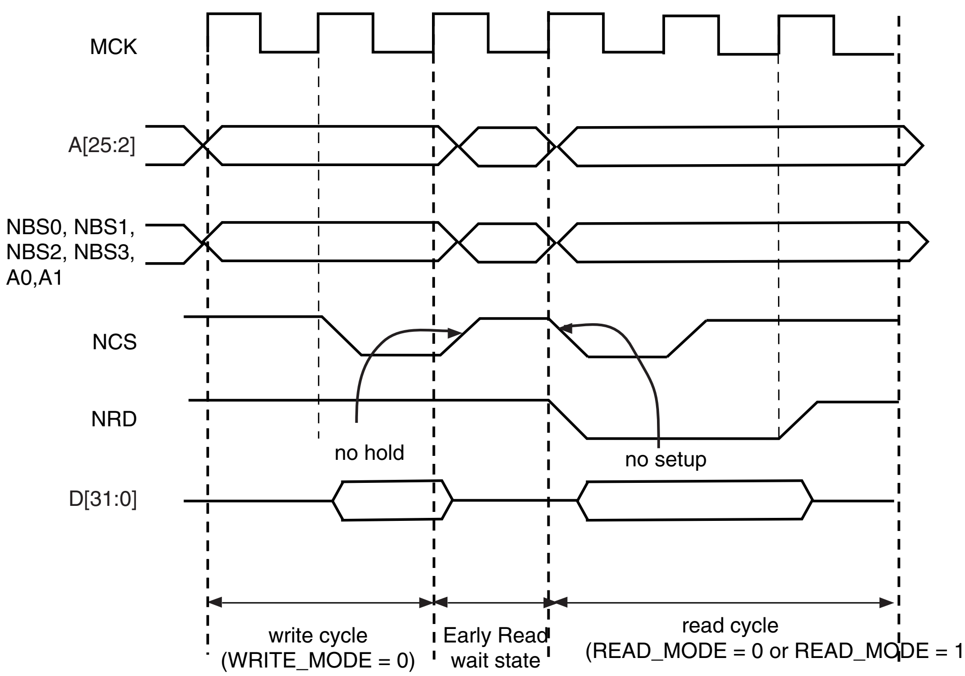
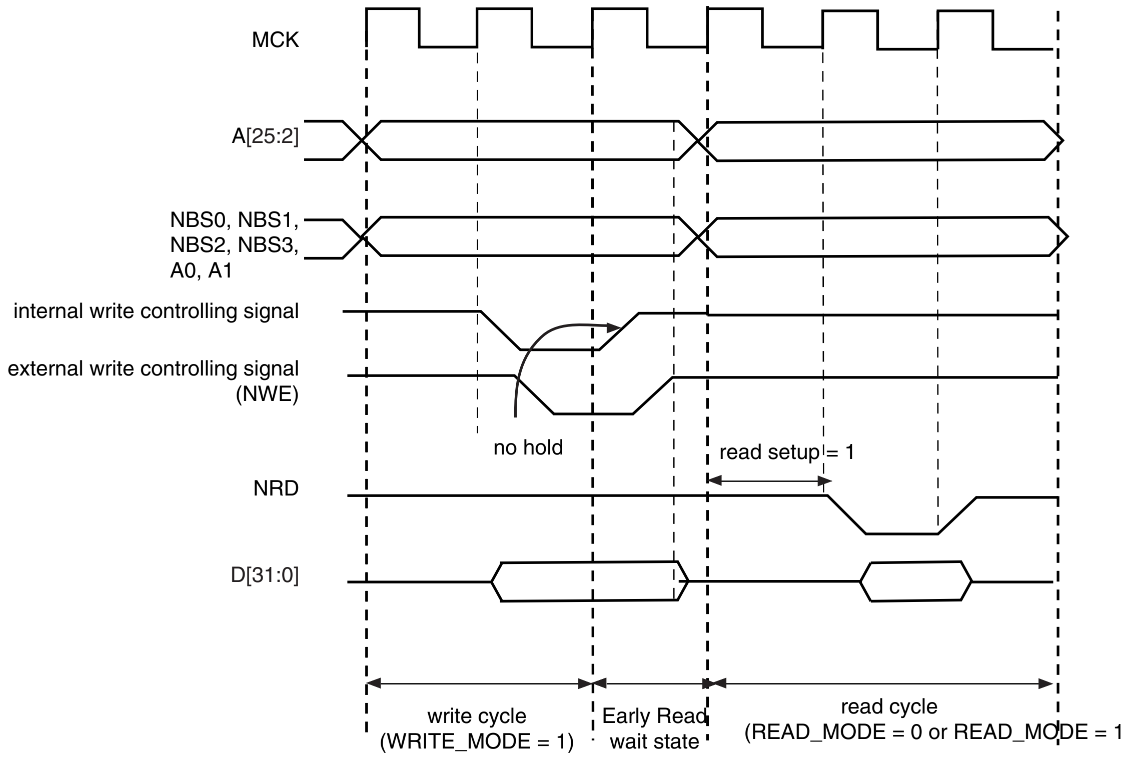


Figure 23-19. Early Read Wait State: NWE-controlled Write with No Hold Followed by a Read with one Set-up Cycle



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.

The user must not change the configuration parameters of an SMC Chip Select (Setup, Pulse, Cycle, Mode) if accesses are performed on this CS during the modification. Any change of the Chip Select parameters, while fetching the code from a memory connected on this CS, may lead to unpredictable behavior. The instructions used to modify the parameters of an SMC Chip Select can be executed from the internal RAM or from a memory connected to another CS.

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 211).

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 197.

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)

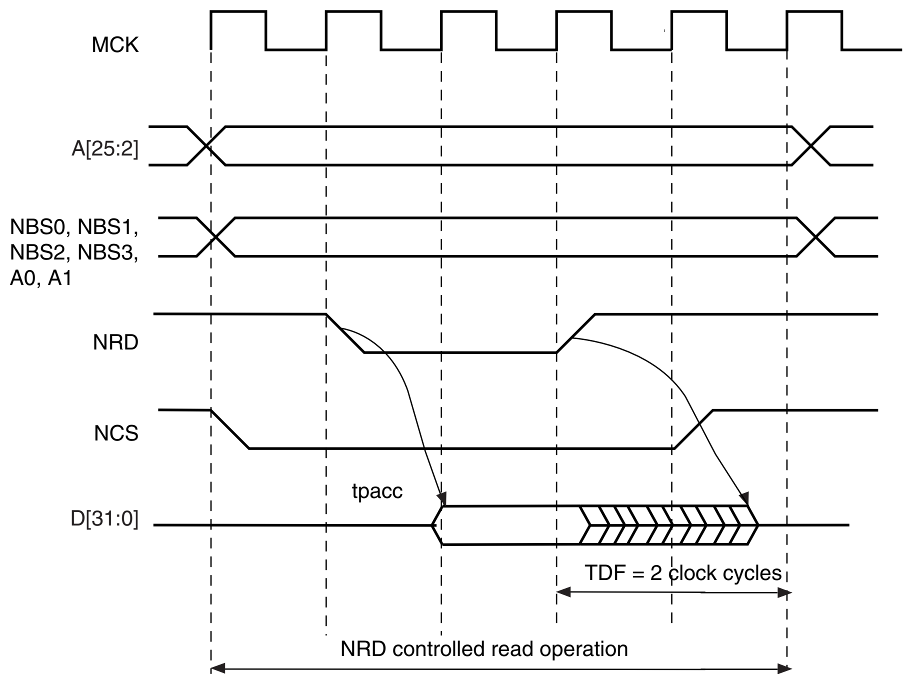
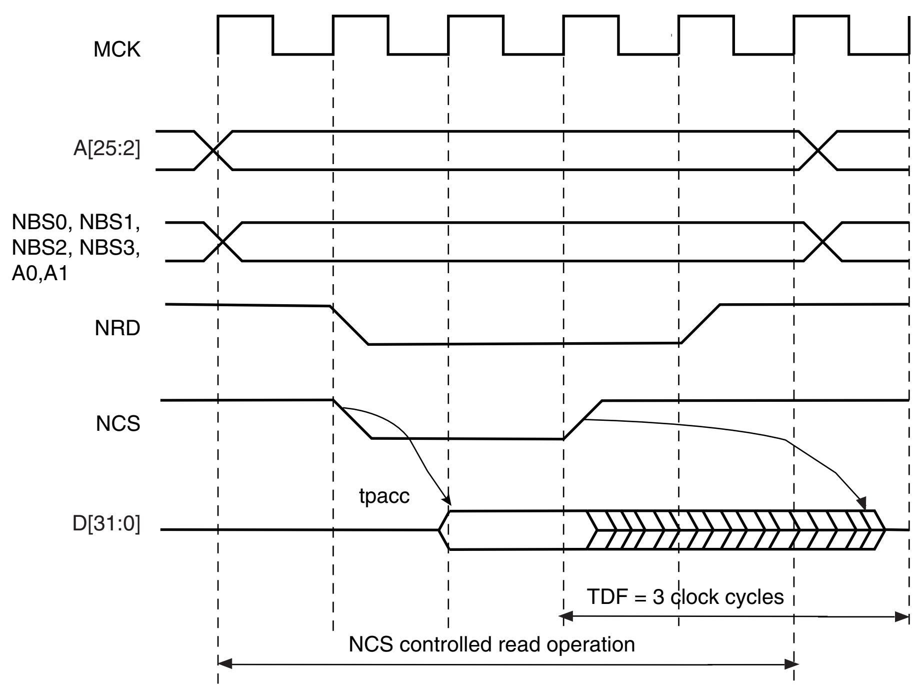


Figure 23-21. TDF Period in NCS Controlled Read Operation (TDF = 3)



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.

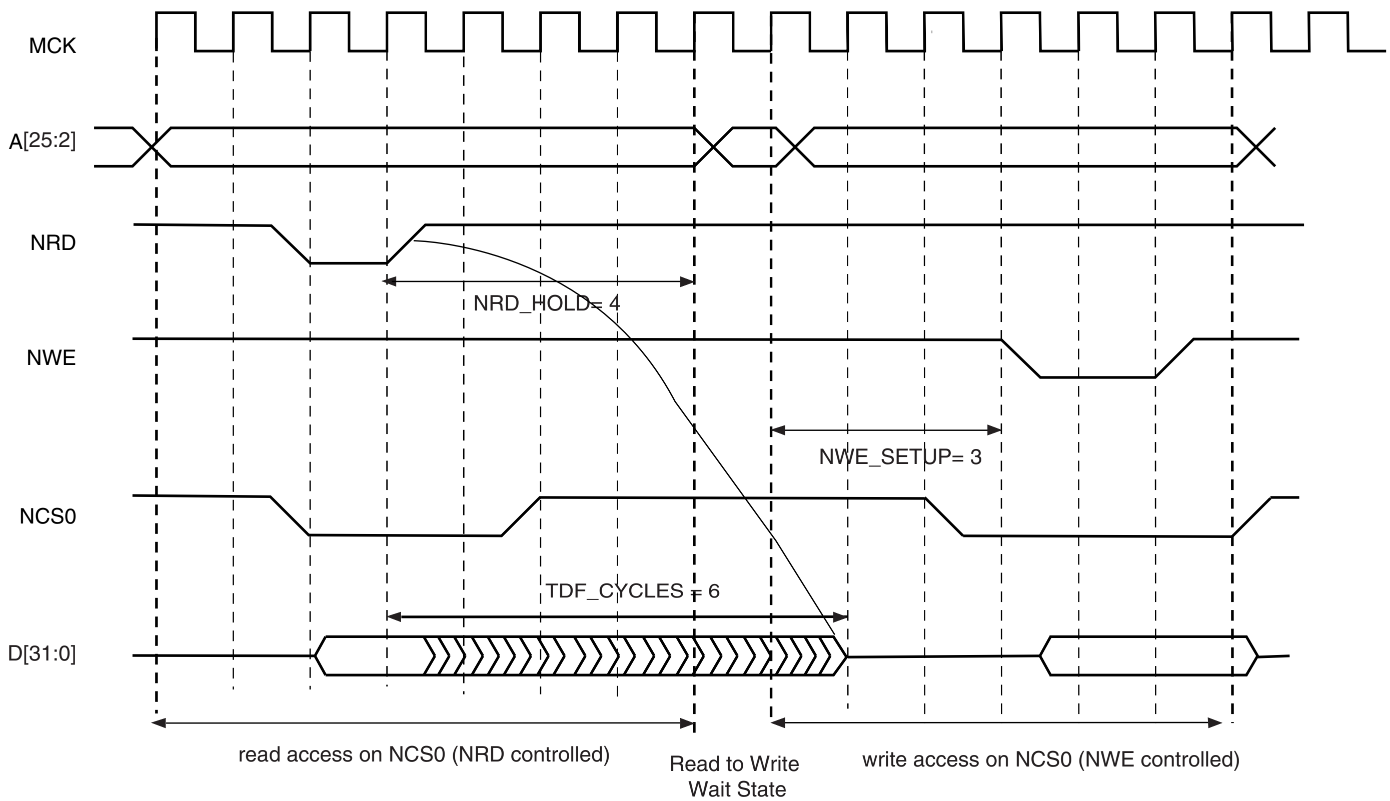
Figure 23-22 shows a read access controlled by NRD, followed by a write access controlled by NWE, on Chip Select 0. Chip Select 0 has been programmed with:

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

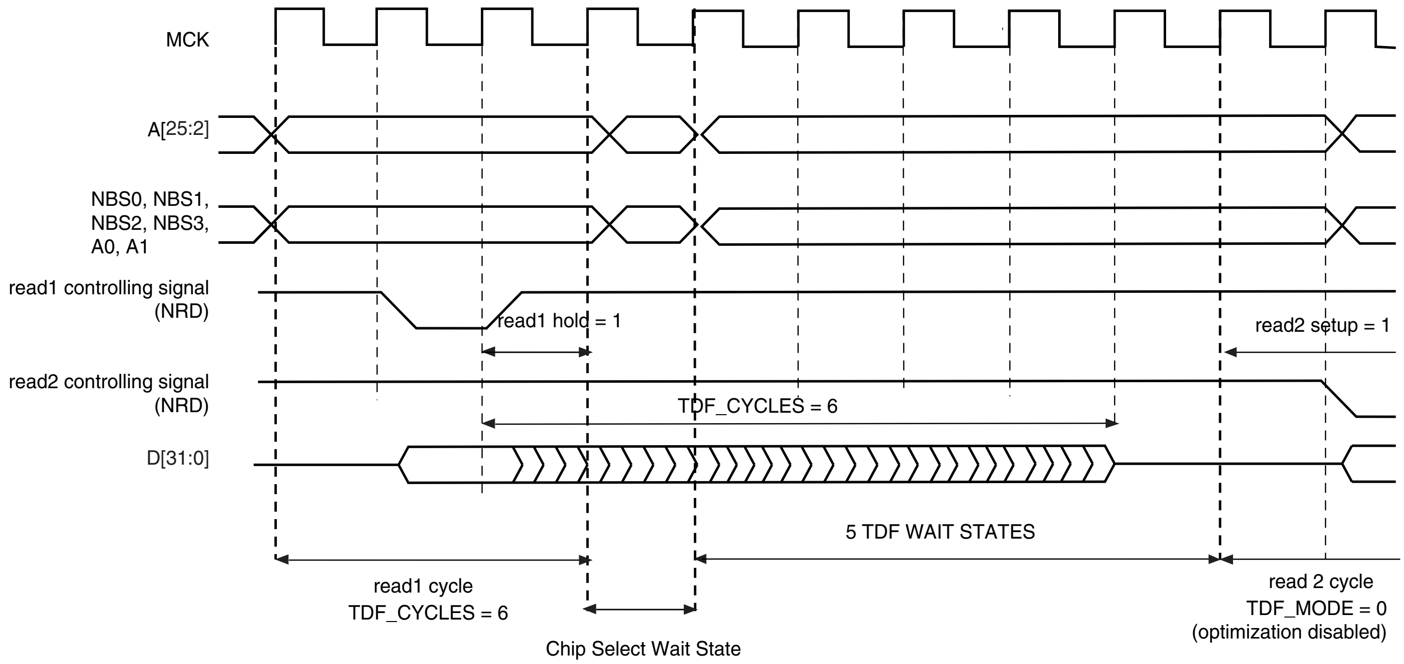


Figure 23-24. TDF Mode = 0: TDF wait states between a read and a write access on different chip selects

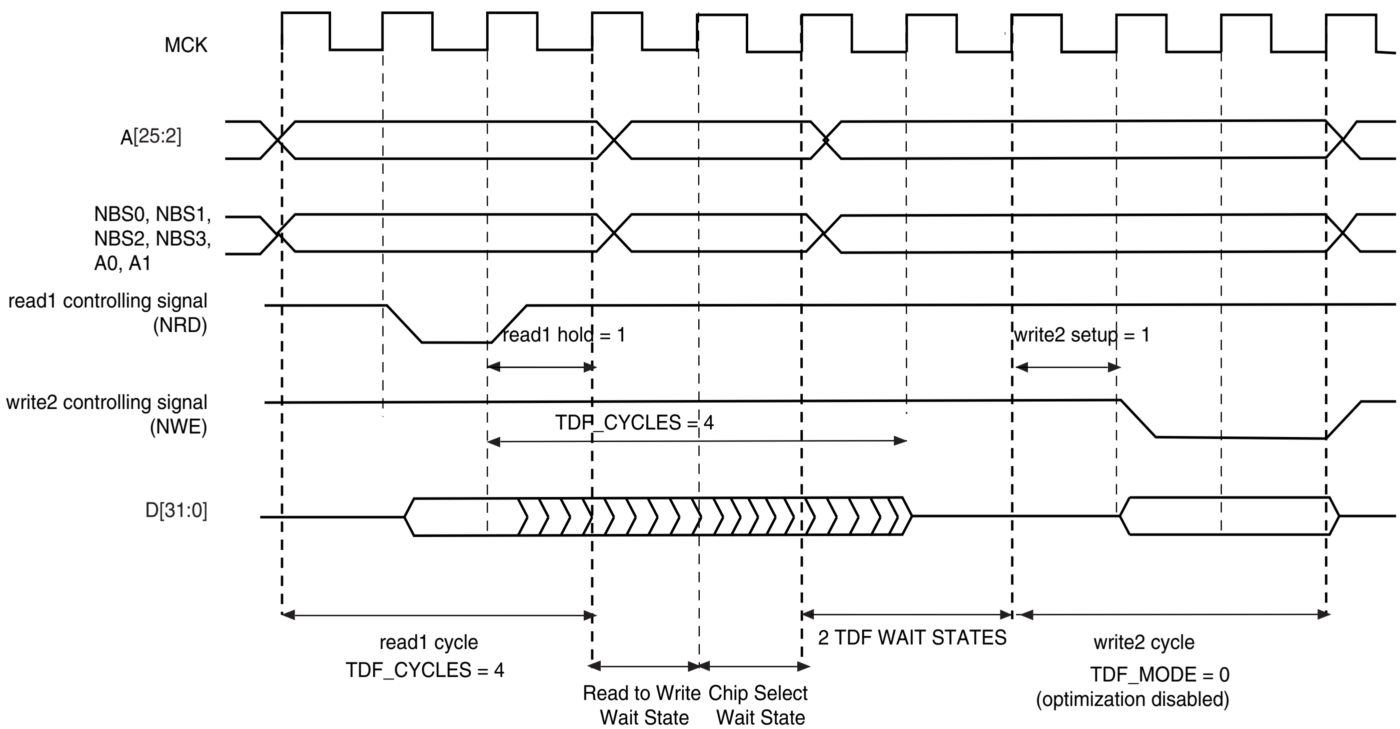
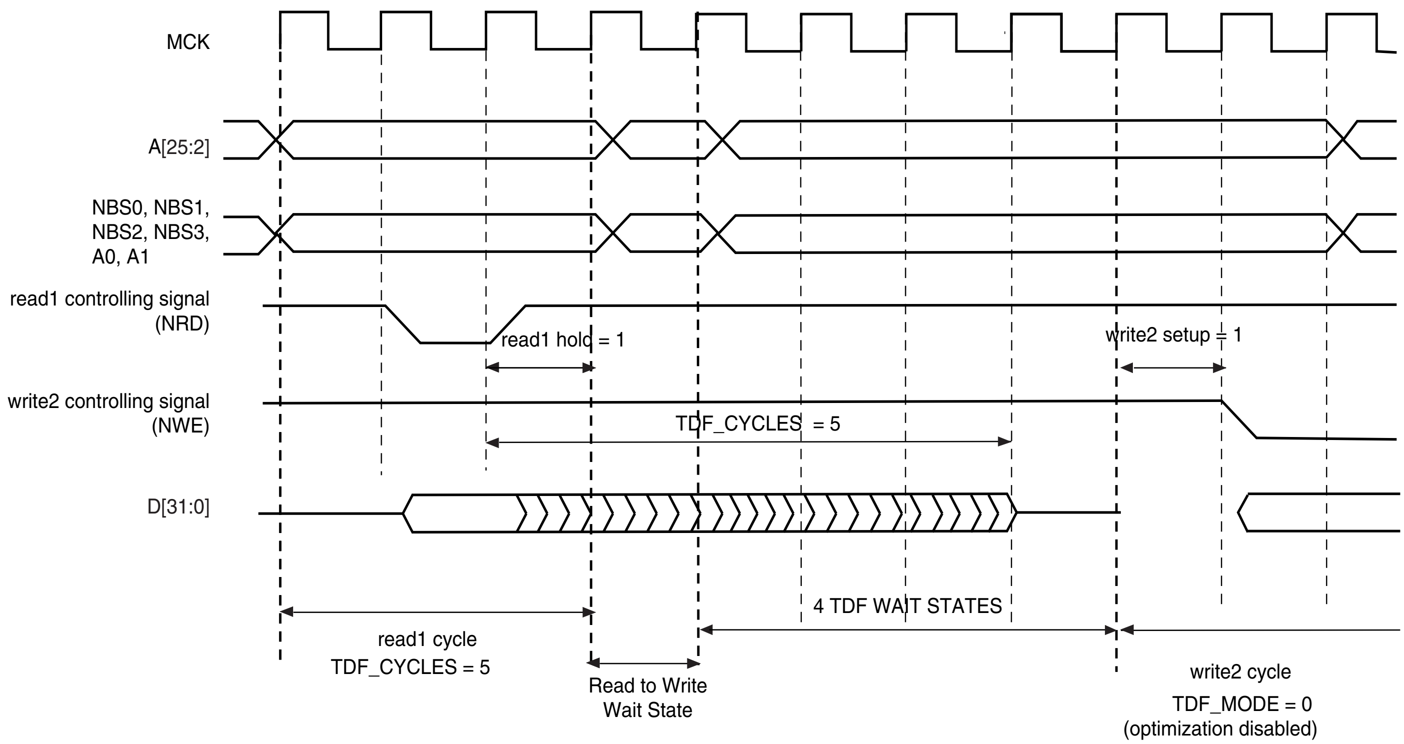


Figure 23-25. TDF Mode = 0: TDF wait states between read and write accesses on the same chip select



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 214), or in Slow Clock Mode (“Slow Clock Mode” on page 211).**

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)

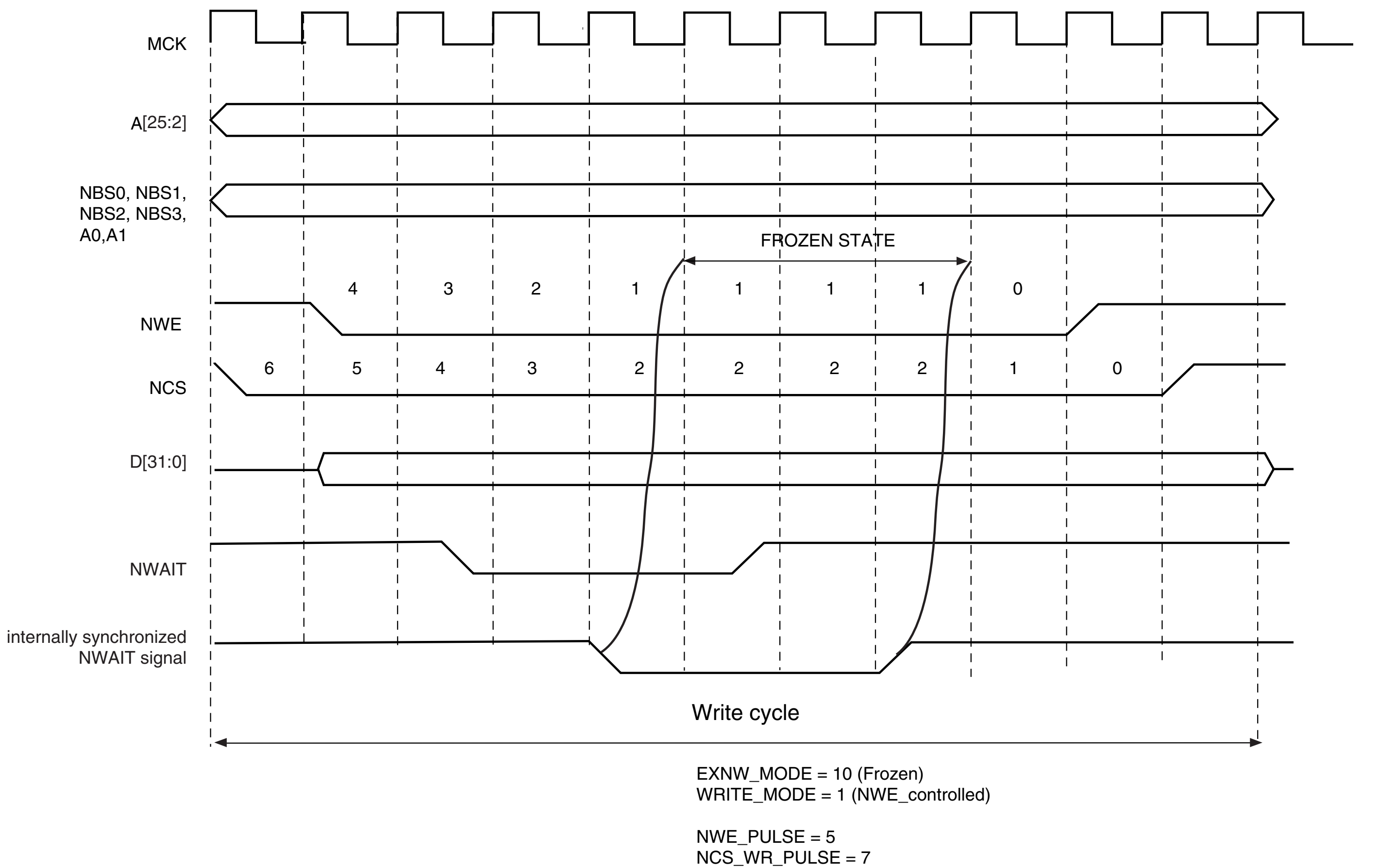
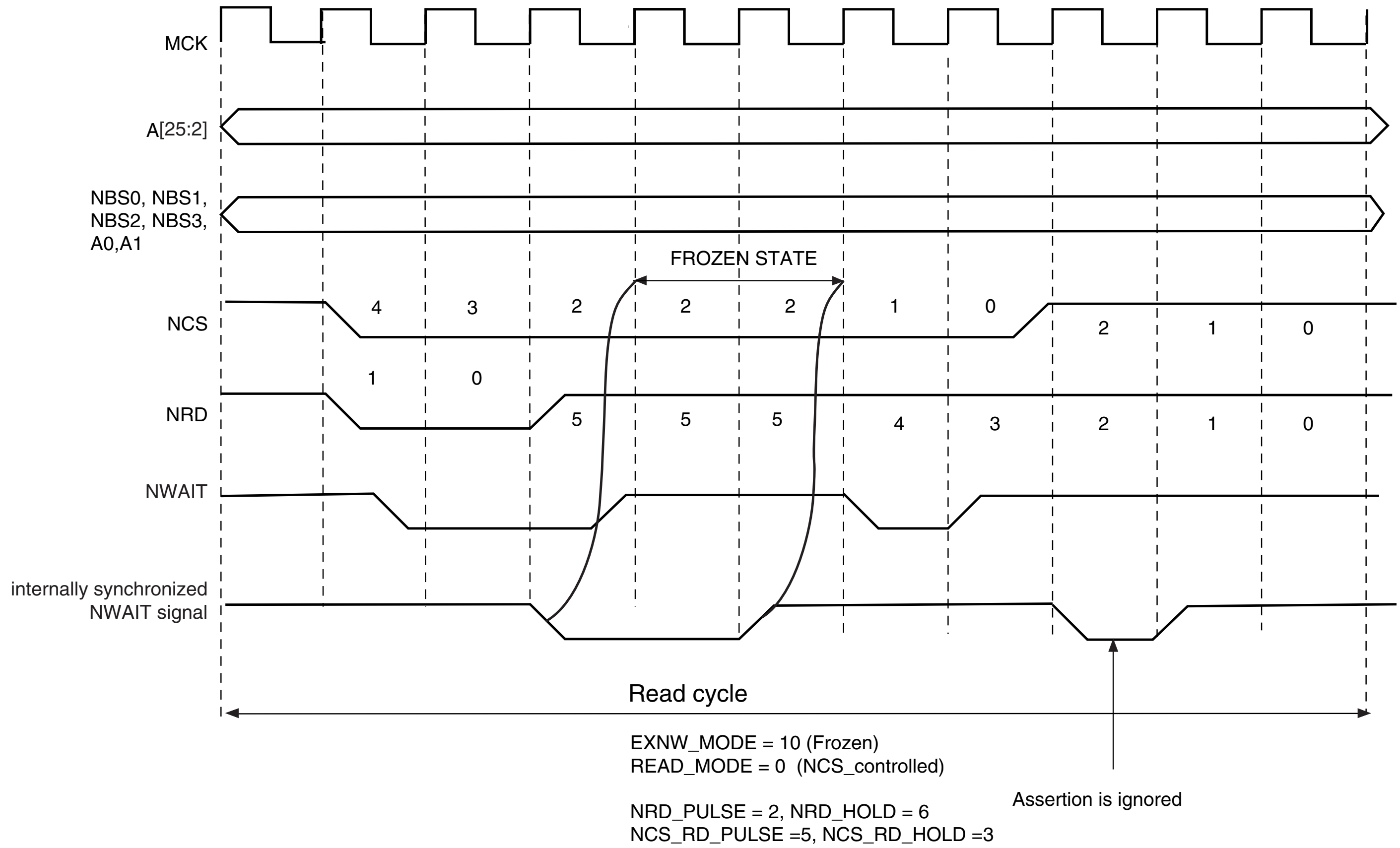


Figure 23-27. Read Access with NWAIT Assertion in Frozen Mode (EXNW_MODE = 10)



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)

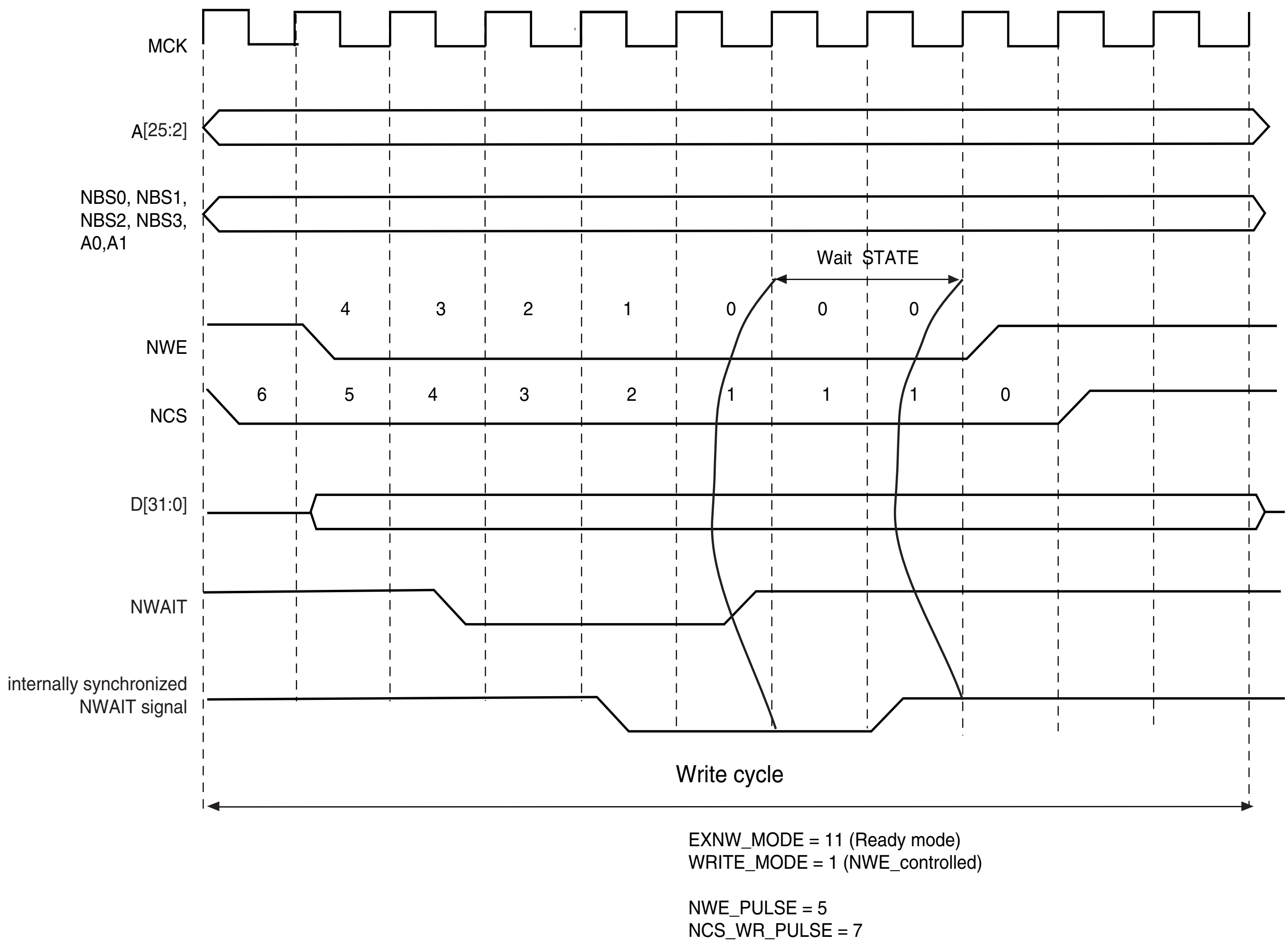
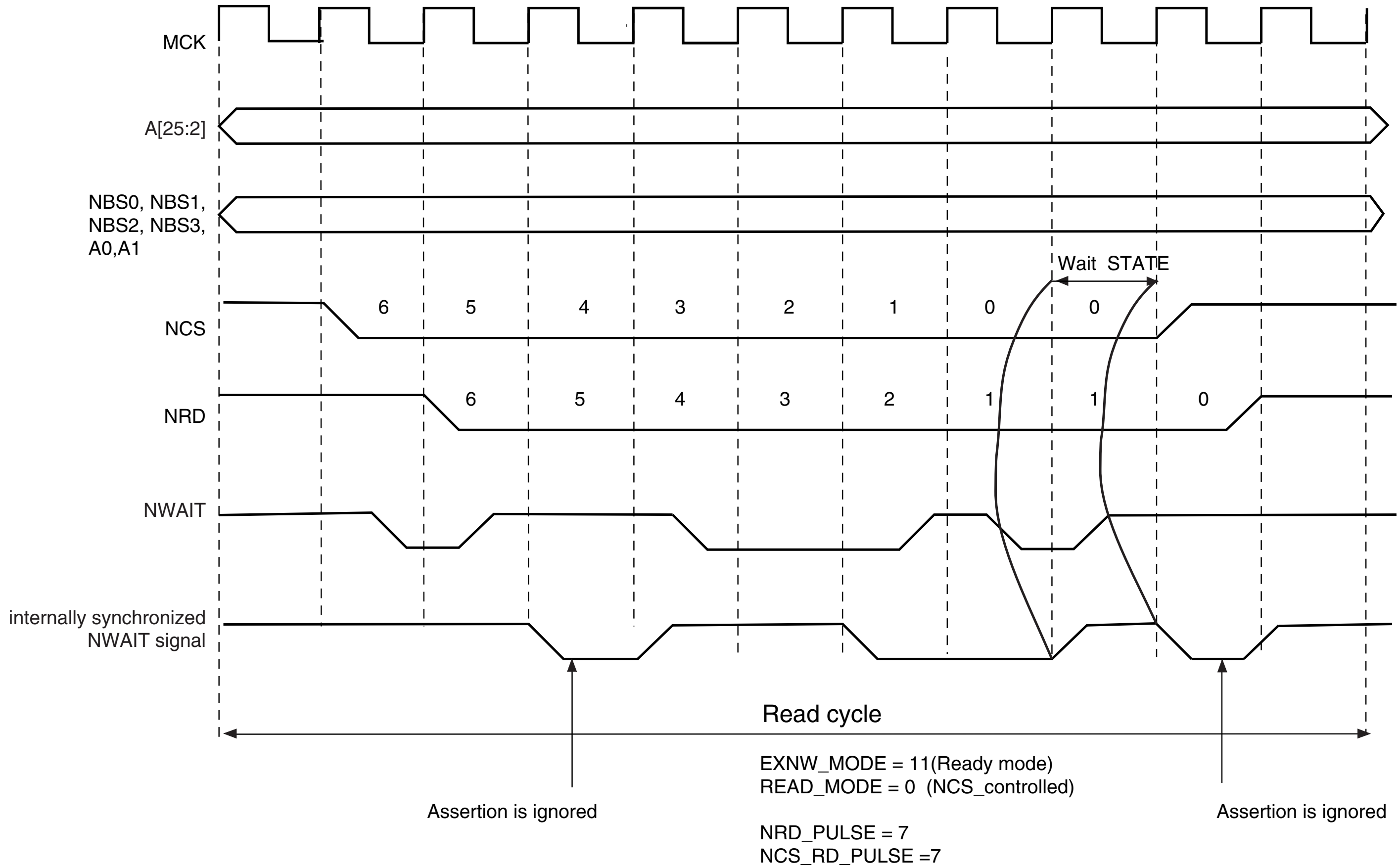


Figure 23-29. NWAIT Assertion in Read Access: Ready Mode (EXNW_MODE = 11)



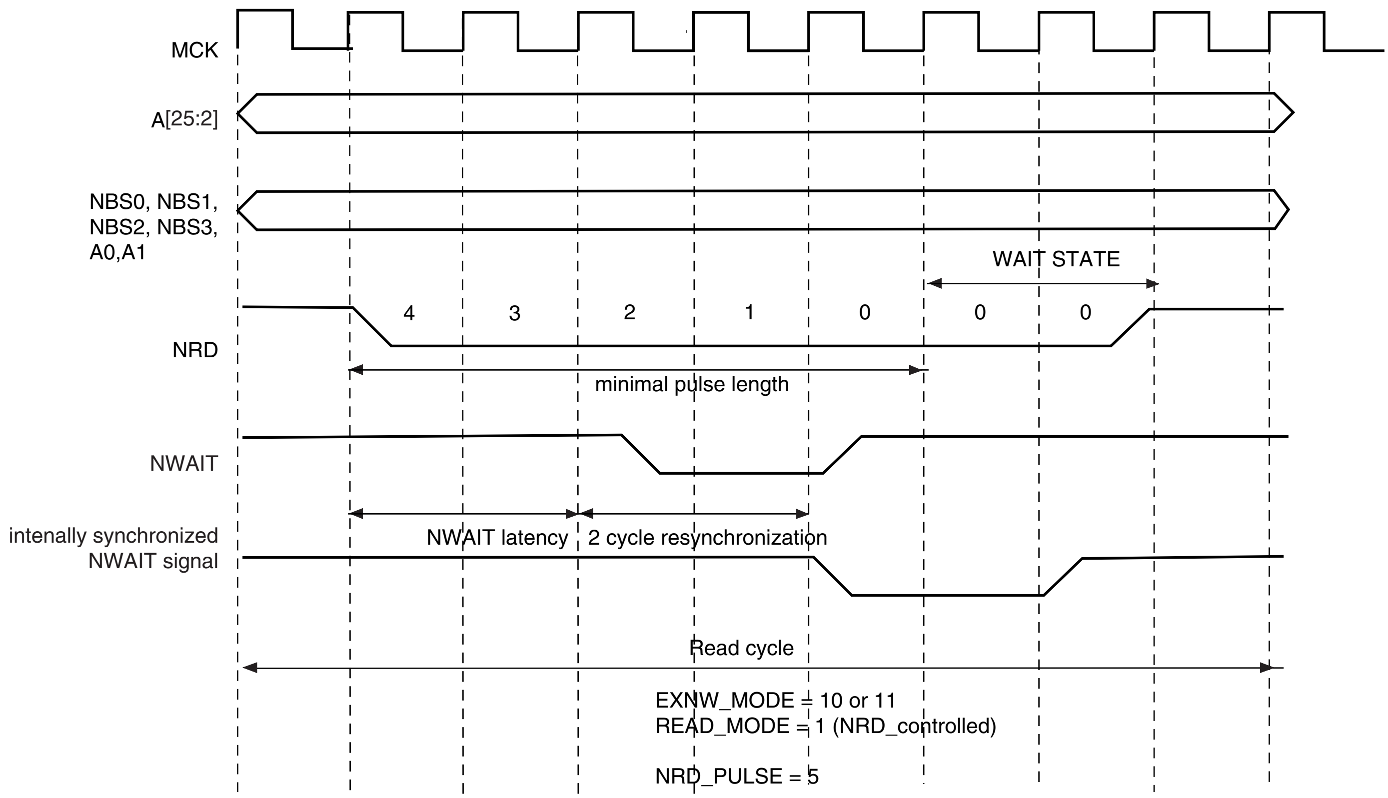
23.11.4 NWAIT Latency and Read/Write Timings

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:

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

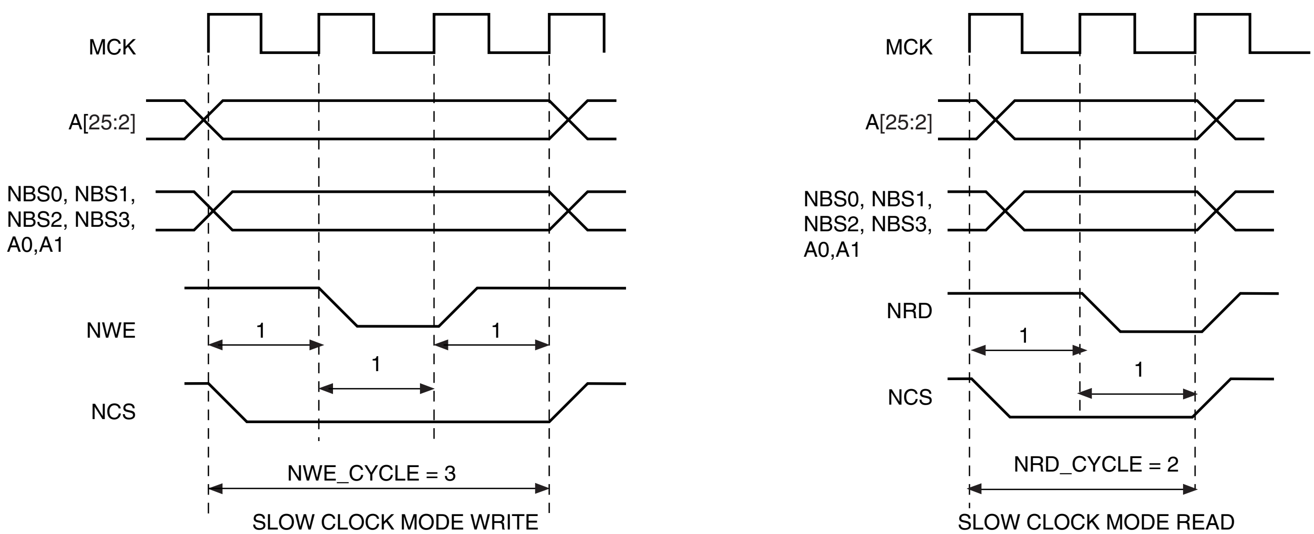


Table 23-6. Read and Write Timing Parameters in Slow Clock Mode

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 212. 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

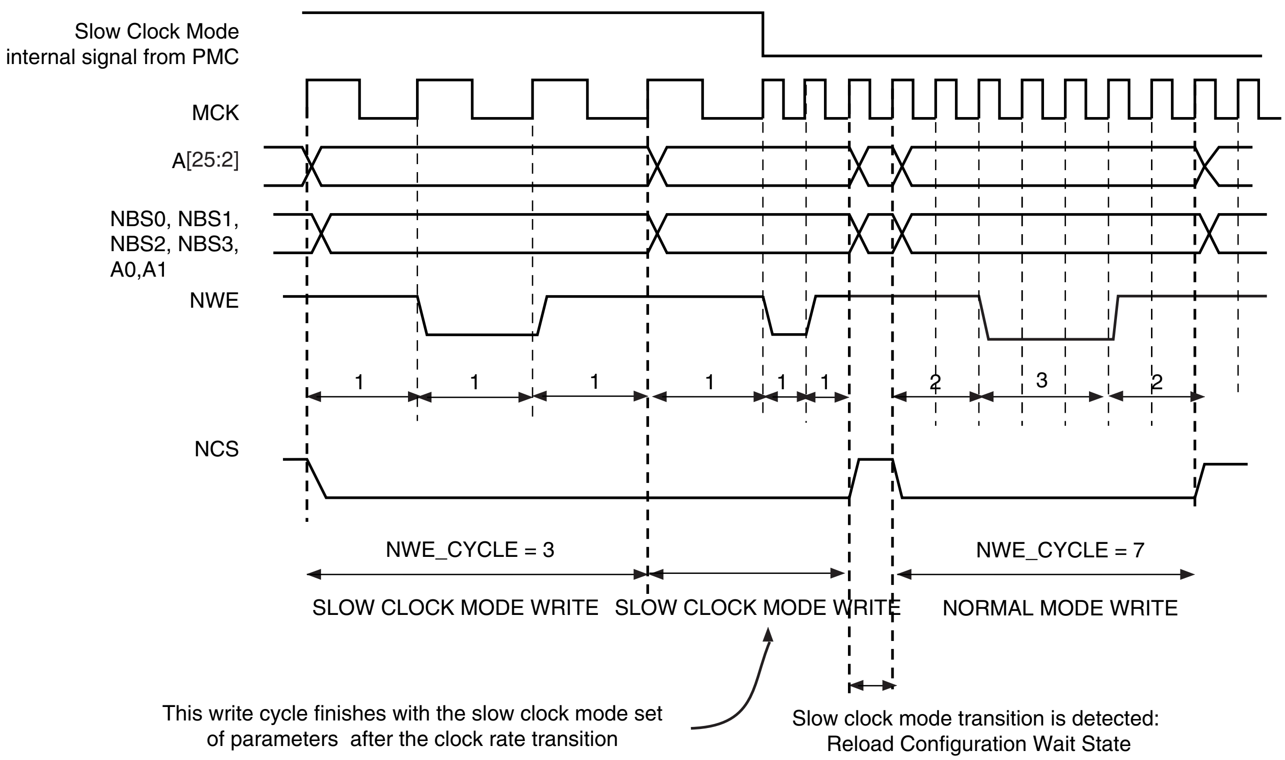
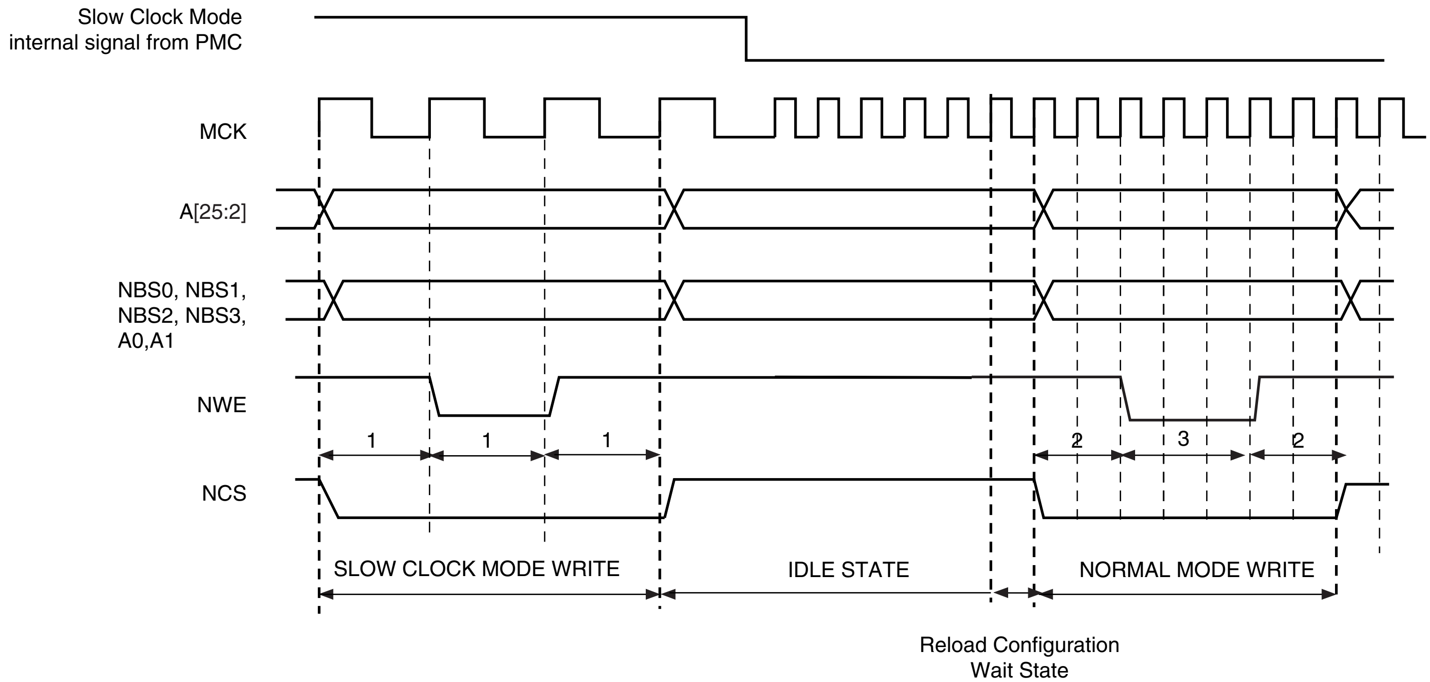


Figure 23-33. Recommended Procedure to Switch from Slow Clock Mode to Normal Mode or from Normal Mode to Slow Clock Mode



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

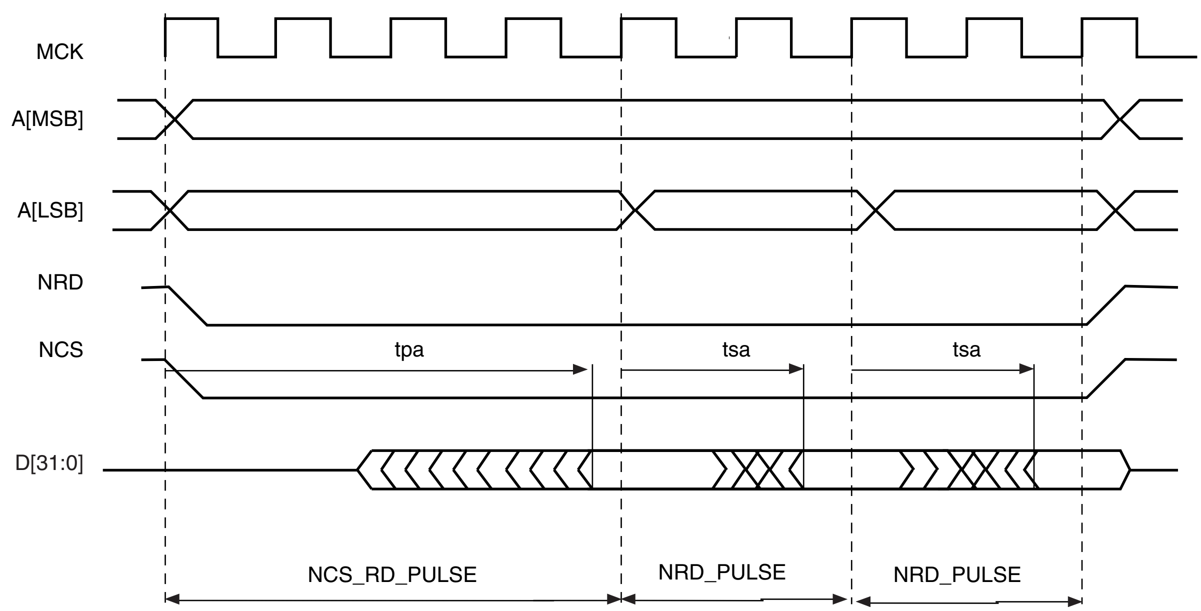
Notes: 1. A denotes the address bus of the memory device

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

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.

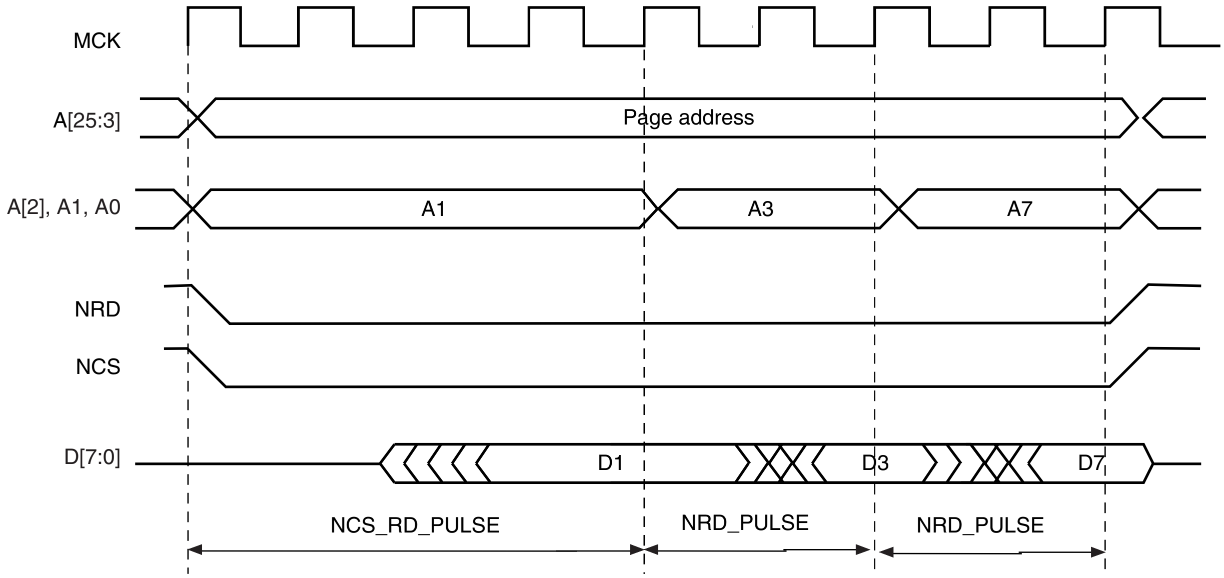
23.13.4 Sequential and Non-sequential Accesses

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. Register Mapping

Offset	Register	Name	Access	Reset
0x10 x CS_number + 0x00	SMC Setup Register	SMC_SETUP	Read-write	0x01010101
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	0x00030003
0x10 x CS_number + 0x0C	SMC Mode Register	SMC_MODE	Read-write	0x10001000
0xEC-0xFC	Reserved	-	-	-

23.14.1 SMC Setup Register

Name: SMC_SETUP[0..5]

Addresses: 0xFFFFE800 [0], 0xFFFFE810 [1], 0xFFFFE820 [2], 0xFFFFE830 [3], 0xFFFFE840 [4], 0xFFFFE850 [5]

Access: Read-write

31	30	29	28	27	26	25	24
–	–	NCS_RD_SETUP					

23	22	21	20	19	18	17	16
–	–	NRD_SETUP					

15	14	13	12	11	10	9	8
–	–	NCS_WR_SETUP					

7	6	5	4	3	2	1	0
–	–	NWE_SETUP					

- **NWE_SETUP: NWE Setup Length**

The NWE signal setup length is defined as:

NWE setup length = (128* NWE_SETUP[5] + NWE_SETUP[4:0]) clock cycles

- **NCS_WR_SETUP: NCS Setup Length in WRITE Access**

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

NCS setup length = (128* NCS_WR_SETUP[5] + NCS_WR_SETUP[4:0]) clock cycles

- **NRD_SETUP: NRD Setup Length**

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

NRD setup length = (128* NRD_SETUP[5] + NRD_SETUP[4:0]) clock cycles

- **NCS_RD_SETUP: NCS Setup Length in READ Access**

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

NCS setup length = (128* NCS_RD_SETUP[5] + NCS_RD_SETUP[4:0]) clock cycles

23.14.2 SMC Pulse Register

Name: SMC_PULSE[0..5]

Addresses: 0xFFFFE804 [0], 0xFFFFE814 [1], 0xFFFFE824 [2], 0xFFFFE834 [3], 0xFFFFE844 [4], 0xFFFFE854 [5]

Access: Read-write

31	30	29	28	27	26	25	24
–	NCS_RD_PULSE						

23	22	21	20	19	18	17	16
–	NRD_PULSE						

15	14	13	12	11	10	9	8
–	NCS_WR_PULSE						

7	6	5	4	3	2	1	0
–	NWE_PULSE						

- **NWE_PULSE: NWE Pulse Length**

The NWE signal pulse length is defined as:

NWE pulse length = (256* NWE_PULSE[6] + NWE_PULSE[5:0]) clock cycles

The NWE pulse length must be at least 1 clock cycle.

- **NCS_WR_PULSE: NCS Pulse Length in WRITE Access**

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

NCS pulse length = (256* NCS_WR_PULSE[6] + NCS_WR_PULSE[5:0]) clock cycles

The NCS pulse length must be at least 1 clock cycle.

- **NRD_PULSE: NRD Pulse Length**

In standard read access, the NRD signal pulse length is defined in clock cycles as:

NRD pulse length = (256* NRD_PULSE[6] + NRD_PULSE[5:0]) clock cycles

The NRD pulse length must be at least 1 clock cycle.

In page mode read access, the NRD_PULSE parameter defines the duration of the subsequent accesses in the page.

- **NCS_RD_PULSE: NCS Pulse Length in READ Access**

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

NCS pulse length = (256* NCS_RD_PULSE[6] + NCS_RD_PULSE[5:0]) clock cycles

The NCS pulse length must be at least 1 clock cycle.

In page mode read access, the NCS_RD_PULSE parameter defines the duration of the first access to one page.

23.14.3 SMC Cycle Register

Name: SMC_CYCLE[0..5]

Addresses: 0xFFFFE808 [0], 0xFFFFE818 [1], 0xFFFFE828 [2], 0xFFFFE838 [3], 0xFFFFE848 [4], 0xFFFFE858 [5]

Access: 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

Name: SMC_MODE[0..5]

Addresses: 0xFFFFE80C [0], 0xFFFFE81C [1], 0xFFFFE82C [2], 0xFFFFE83C [3], 0xFFFFE84C [4], 0xFFFFE85C [5]

Access: 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. DDR/SDR SDRAM Controller (DDRSDRC)

24.1 Description

The DDR/SDR SDRAM Controller (DDRSDRC) is a multiport memory controller. It comprises four slave AHB interfaces. All simultaneous accesses (four independent AHB ports) are interleaved to maximize memory bandwidth and minimize transaction latency due to SDRAM protocol.

The DDRSDRC extends the memory capabilities of a chip by providing the interface to an external 16-bit or 32-bit SDR-SDRAM device and external 16-bit DDR-SDRAM device. The page size supports ranges from 2048 to 16384 and the number of columns from 256 to 4096. It supports byte (8-bit), half-word (16-bit) and word (32-bit) accesses.

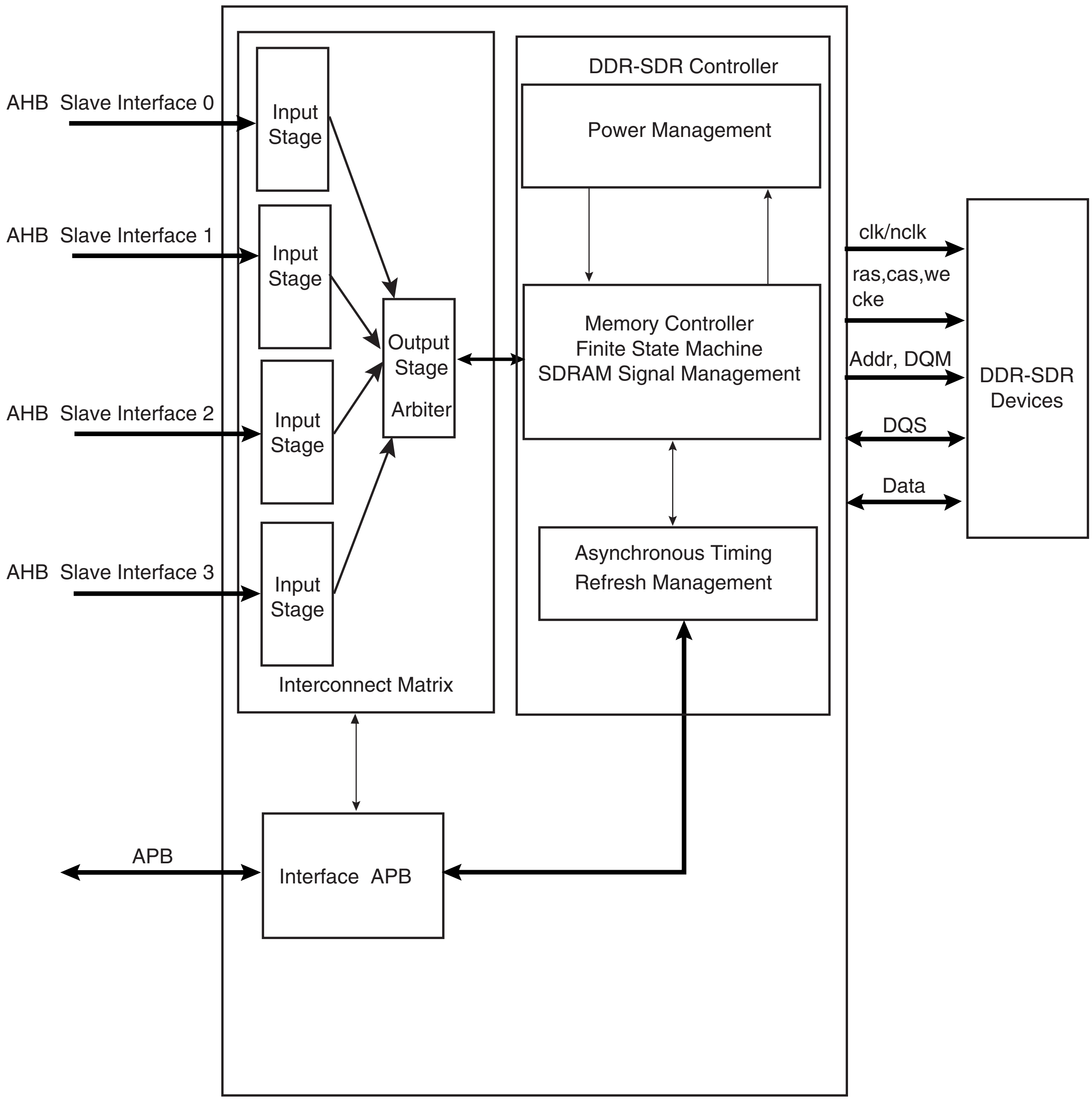
The DDRSDRC supports a read or write burst length of 8 locations which frees the command and address bus to anticipate the next command, thus reducing latency imposed by the SDRAM protocol and improving the SDRAM bandwidth. Moreover it keeps track of the active row in each bank, thus maximizing SDRAM 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 DDRSDRC supports a CAS latency of 2, 2.5 or 3 and optimizes the read access depending on the frequency.

The features of self refresh, power-down and deep power-down modes minimize the consumption of the SDRAM device.

The DDRSDRC user interface is compliant with ARM Advance Peripheral Bus (APB rev2).

24.2 DDRSDRC Module Diagram

Figure 24-1. DDRSDRC Module Diagram



DDRSDRC is partitioned in two blocks (see Figure 24-1):

- An Interconnect-Matrix that manages concurrent accesses on the AHB bus between four AHB masters and integrates an arbiter.
- A controller that translates AHB requests (Read/Write) in the SDRAM protocol.

24.3 Product Dependencies

24.3.1 SDR-SDRAM Initialization

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

1. Program the memory device type into the Memory Device Register (see Section 24.6.7 on page 254).
2. Program the features of the SDR-SDRAM device into the Timing Register (asynchronous timing (trc, tras, etc.)), and into the Configuration Register (number of columns, rows, banks, cas latency) (see Section 24.6.3 on page 247, Section 24.6.4 on page 249 and Section 24.6.5 on page 251).
3. For low-power SDRAM, temperature-compensated self refresh (TCSR), drive strength (DS) and partial array self refresh (PASR) must be set in the Low-power Register (see Section 24.6.6 on page 252).

A minimum pause of 200 µs is provided to precede any signal toggle.

4. A NOP command is issued to the SDR-SDRAM. Program NOP command into Mode Register, the application must set Mode to 1 in the Mode Register (See Section 24.6.1 on page 245). Perform a write access to any SDR-SDRAM address to acknowledge this command. Now the clock which drives SDR-SDRAM device is enabled.
5. An all banks precharge command is issued to the SDR-SDRAM. Program all banks precharge command into Mode Register, the application must set Mode to 2 in the Mode Register (See Section 24.6.1 on page 245). Perform a write access to any SDR-SDRAM address to acknowledge this command.
6. Eight auto-refresh (CBR) cycles are provided. Program the auto refresh command (CBR) into Mode Register, the application must set Mode to 4 in the Mode Register (see Section 24.6.1 on page 245).Performs a write access to any SDR-SDRAM location eight times to acknowledge these commands.
7. A Mode Register set (MRS) cycle is issued to program the parameters of the SDR-SDRAM devices, in particular CAS latency and burst length. The application must set Mode to 3 in the Mode Register (see Section 24.6.1 on page 245) and perform a write access to the SDR-SDRAM to acknowledge this command. The write address must be chosen so that BA[1:0] are set to 0. For example, with a 16-bit 128 MB SDR-SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be done at the address 0x20000000[1].

Note: 1. This address is for example purposes only. The real address is dependent on implementation in the product.

8. For low-power SDR-SDRAM initialization, an Extended Mode Register set (EMRS) cycle is issued to program the SDR-SDRAM parameters (TCSR, PASR, DS). The application must set Mode to 5 in the Mode Register (see Section 24.6.1 on page 245) and perform a write access to the SDR-SDRAM to acknowledge this command. 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.
9. The application must go into Normal Mode, setting Mode to 0 in the Mode Register (see Section 24.6.1 on page 245) and perform a write access at any location in the SDRAM to acknowledge this command.
10. Write the refresh rate into the count field in the SDRAMC Refresh Timer register (see page 246). (Refresh rate = delay between refresh cycles). The SDR-SDRAM device requires a refresh every 15.625 µs or 7.81 µs. With a 100 MHz frequency, the refresh

timer count register must to be set with (15.625 /100 MHz) = 1562 i.e. 0x061A or (7.81 /100 MHz) = 781 i.e. 0x030d

After initialization, the SDR-SDRAM device is fully functional.

24.3.2 DDR-SDRAM Initialization

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

1. Program the memory device type into the Memory Device Register (see Section 24.6.7 on page 254).
2. Program the features of DDR-SDRAM device into the Timing Register (asynchronous timing (trc, tras, etc.)), and into the Configuration Register (number of columns, rows, banks, cas latency and output drive strength) (see Section 24.6.3 on page 247, Section 24.6.4 on page 249 and Section 24.6.5 on page 251).
3. An NOP command is issued to the DDR-SDRAM. Program the NOP command into the Mode Register, the application must set Mode to 1 in the Mode Register (see Section 24.6.1 on page 245). Perform a write access to any DDR-SDRAM address to acknowledge this command. Now clocks which drive DDR-SDRAM device are enabled.

A minimum pause of 200 µs is provided to precede any signal toggle.

4. An NOP command is issued to the DDR-SDRAM. Program the NOP command into the Mode Register, the application must set Mode to 1 in the Mode Register (see Section 24.6.1 on page 245). Perform a write access to any DDR-SDRAM address to acknowledge this command. Now CKE is driven high.
5. An all banks precharge command is issued to the DDR-SDRAM. Program all banks precharge command into the Mode Register, the application must set Mode to 2 in the Mode Register (See Section 24.6.1 on page 245). Perform a write access to any DDR-SDRAM address to acknowledge this command
6. An Extended Mode Register set (EMRS) cycle is issued to enable DLL and to program output drive strength (DIC/DS). The application must set Mode to 5 in the Mode Register (see Section 24.6.1 on page 245) and perform a write access to the DDR-SDRAM to acknowledge this command. 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 DDR-SDRAM (12 rows, 9 columns, 4 banks) bank address, the DDR-SDRAM write access should be done at the address 0x20800000 or 0x20400000.

An additional 200 cycles of clock are required for locking DLL

7. Program DLL field into the Configuration Register (see Section 24.6.3 on page 247) to high (Enable DLL reset).
8. A Mode Register set (MRS) cycle is issued to reset DLL. The application must set Mode to 3 in the Mode Register (see Section 24.6.1 on page 245) and perform a write access to the DDR-SDRAM to acknowledge this command. The write address must be chosen so that BA[1:0] bits are set to 0. For example, with a 16-bit 128 MB DDR-SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be done at the address 0x20000000.
9. Two auto-refresh (CBR) cycles are provided. Program the auto refresh command (CBR) into the Mode Register, the application must set Mode to 4 in the Mode Register (see Section 24.6.1 on page 245). Performs a write access to any DDR-SDRAM location twice to acknowledge these commands.
10. Program DLL field into the Configuration Register (see Section 24.6.3 on page 247) to low (Disable DLL reset).

11. A Mode Register set (MRS) cycle is issued to program the parameters of the DDR-SDRAM devices, in particular CAS latency, burst length and to disable DLL reset. The application must set Mode to 3 in the Mode Register (see Section 24.6.1 on page 245) and perform a write access to the DDR-SDRAM to acknowledge this command. 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
12. A mode Normal command is provided. Program the normal mode into Mode Register (see Section 24.6.1 on page 245). Perform a write access to any DDR-SDRAM address to acknowledge this command.
13. Perform a write access to any DDR-SDRAM address.
14. Write the refresh rate into the count field in the Refresh Timer register (see page 246). (Refresh rate = delay between refresh cycles). The DDR-SDRAM device requires a refresh every 15.625 µs or 7.81 µs. With a 100 MHz frequency, the refresh timer count register must to be set with (15.625 /100 MHz) = 1562 i.e. 0x061A or (7.81 /100 MHz) = 781 i.e. 0x030d

After initialization, the DDR-SDRAM devices are fully functional.

24.3.3 Low-power DDR-SDRAM Initialization

The initialization sequence is generated by software. The low-power DDR-SDRAM devices are initialized by the following sequence:

1. Program the memory device type into the Memory Device Register (see Section 24.6.7 on page 254).
2. Program the features of the low-power DDR-SDRAM device into the Configuration Register: asynchronous timing (trc, DDRSDRAMC, etc.), number of columns, rows, banks, cas latency. See Section 24.6.3 on page 247, Section 24.6.4 on page 249 and Section 24.6.5 on page 251.
3. Program temperature compensated self refresh (tcr), Partial array self refresh (pasr) and Drive strength (ds) into the Low-power Register. See Section 24.6.6 on page 252.

A minimum pause of 200 µs will be provided to precede any signal toggle.

4. An NOP command will be issued to the DDR-SDRAM. Program NOP command into the Mode Register, the application must set Mode to 1 in the Mode Register (see Section 24.6.1 on page 245). Perform a write access to any DDR-SDRAM address to acknowledge this command. Now clocks which drive DDR-SDRAM device are enabled.
5. An all banks precharge command is issued to the DDR-SDRAM. Program all banks precharge command into the Mode Register, the application must set Mode to 2 in the Mode Register (See Section 24.6.1 on page 245). Perform a write access to any DDR-SDRAM address to acknowledge this command
6. Two auto-refresh (CBR) cycles are provided. Program the auto refresh command (CBR) into the Mode Register, the application must set Mode to 4 in the Mode Register (see Section 24.6.1 on page 245). Perform a write access to any DDR-SDRAM location twice to acknowledge these commands.
7. An Extended Mode Register set (EMRS) cycle is issued to program the DDR-SDRAM parameters (TCSR, PASR, DS). The application must set Mode to 5 in the Mode Register (see Section 24.6.1 on page 245) and perform a write access to the SDRAM to acknowledge this command. 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.

8. A Mode Register set (MRS) cycle is issued to program the parameters of the DDR-SDRAM devices, in particular CAS latency, burst length. The application must set Mode to 3 in the Mode Register (see Section 24.6.1 on page 245) and perform a write access to the DDR-SDRAM to acknowledge this command. The write address must be chosen so that BA[1:0] bits are set to 0. For example, with a 16-bit 128 MB DDR-SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be done at the address 0x20000000
9. The application must go into Normal Mode, setting Mode to 0 in the Mode Register (see Section 24.6.1 on page 245) and performing a write access at any location in the DDR-SDRAM to acknowledge this command.
10. Perform a write access to any DDR-SDRAM address.
11. Write the refresh rate into the count field in the SDRAMC Refresh Timer register (see page 246). (Refresh rate = delay between refresh cycles). The DDR-SDRAM device requires a refresh every 15.625 µs or 7.81 µs. With a 100 MHz frequency, the refresh timer count register must to be set with (15.625 /100 MHz) = 1562 i.e. 0x061A or (7.81 /100 MHz) = 781 i.e. 0x030d
12. After initialization, the DDR-SDRAM device is fully functional.

24.4 Functional Description

24.4.1 SDRAM Controller Write Cycle

The DDRSDRC allows burst access or single access in normal mode (mode = 000). Whatever the access type, the DDRSDRC keeps track of the active row in each bank, thus maximizing performance.

The SDRAM device is programmed with a burst length equal to 8. This determines the length of a sequential data input by the write command that is set to 8. The latency from write command to data input is fixed to 1 in the case of DDR-SDRAM devices. In the case of SDR-SDRAM devices, there is no latency from write command to data input.

To initiate a single access, the DDRSDRC checks if the page access is already open. If row/bank addresses match with the previous row/bank addresses, the controller generates a write command. If the bank addresses are not identical or if bank addresses are identical but the row addresses are not identical, the 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) command. As the burst length is fixed to 8, in the case of single access, it has to stop the burst, otherwise seven invalid values may be written. In the case of SDR-SDRAM devices, a Burst Stop command is generated to interrupt the write operation. In the case of DDR-SDRAM devices, Burst Stop command is not supported for the burst write operation. In order to then interrupt the write operation, Dm must be set to 1 to mask invalid data (see Figure 24-2 on page 229 and Figure 24-4 on page 230) and DQS must continue to toggle.

To initiate a burst access, the DDRSDRC 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 non-sequential access, then an automatic access break is inserted, the DDRSDRC 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 (tRP) commands and active/write (tRCD) commands.

For a definition of timing parameters, refer to Section 24.6.4 "DDRSDRC Timing 0 Parameter Register" on page 249.

Write accesses to the SDRAM devices are burst oriented and the burst length is programmed to 8. It determines the maximum number of column locations that can be accessed for a given write command. When the write command is issued, 8 columns are selected. All accesses for that burst take place within these eight columns, thus the burst wraps within these 8 columns if a boundary is reached. These 8 columns are selected by addr[13:3]. addr[2:0] is used to select the starting location within the block.

In the case of incrementing burst (INCR/INCR4/INCR8/INCR16), the addresses can cross the 16-byte boundary of the SDRAM device. For example, in the case of DDR-SDRAM devices, when a transfer (INCR4) starts at address 0x0C, the next access is 0x10, but since the burst length is programmed to 8, the next access is at 0x00. Since the boundary is reached, the burst is wrapping. The DDRSDRC takes this feature of the SDRAM device into account. In the case of transfer starting at address 0x04/0x08/0x0C (DDR-SDRAM devices) or starting at address 0x10/0x14/0x18/0x1C, two write commands are issued to avoid to wrap when the boundary is reached. The last write command is subject to DM input logic level. If DM is registered high, the corresponding data input is ignored and write access is not done. This avoids additional writing being done.

Figure 24-2. Single Write Access, Row Closed, DDR-SDRAM Device

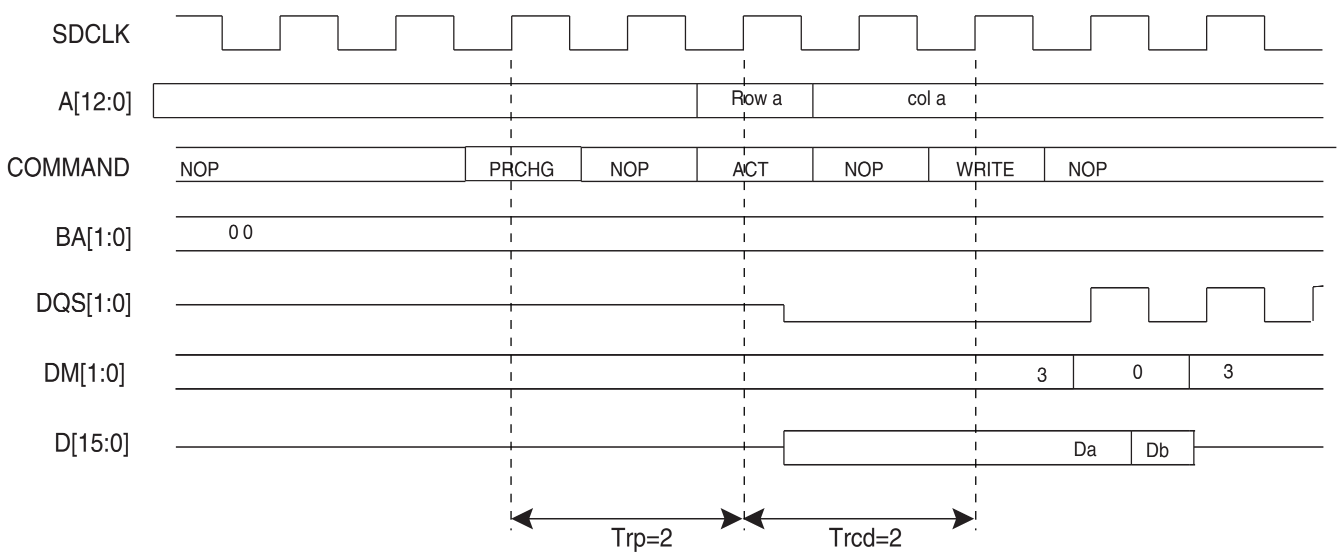


Figure 24-3. Single Write Access, Row Closed, SDR-SDRAM Device

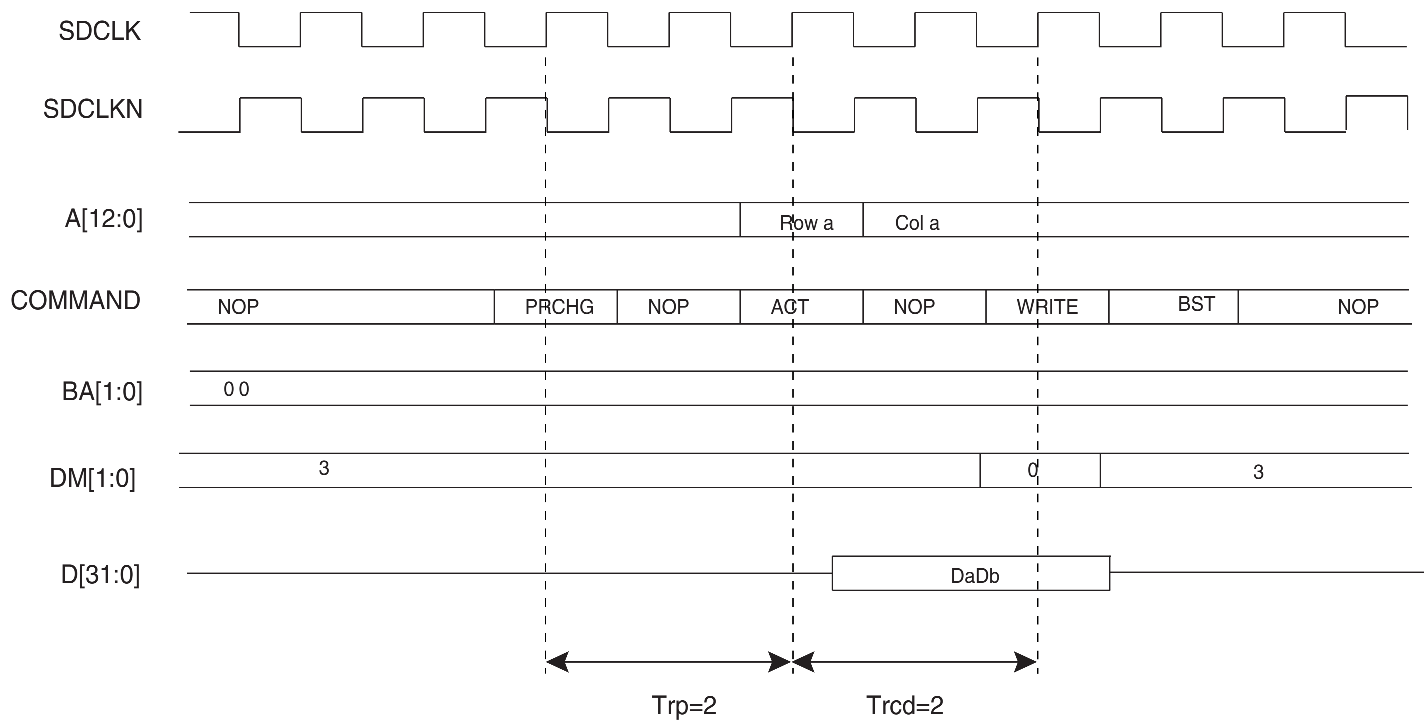


Figure 24-4. Burst Write Access, Row Closed, DDR-SDRAM Devices

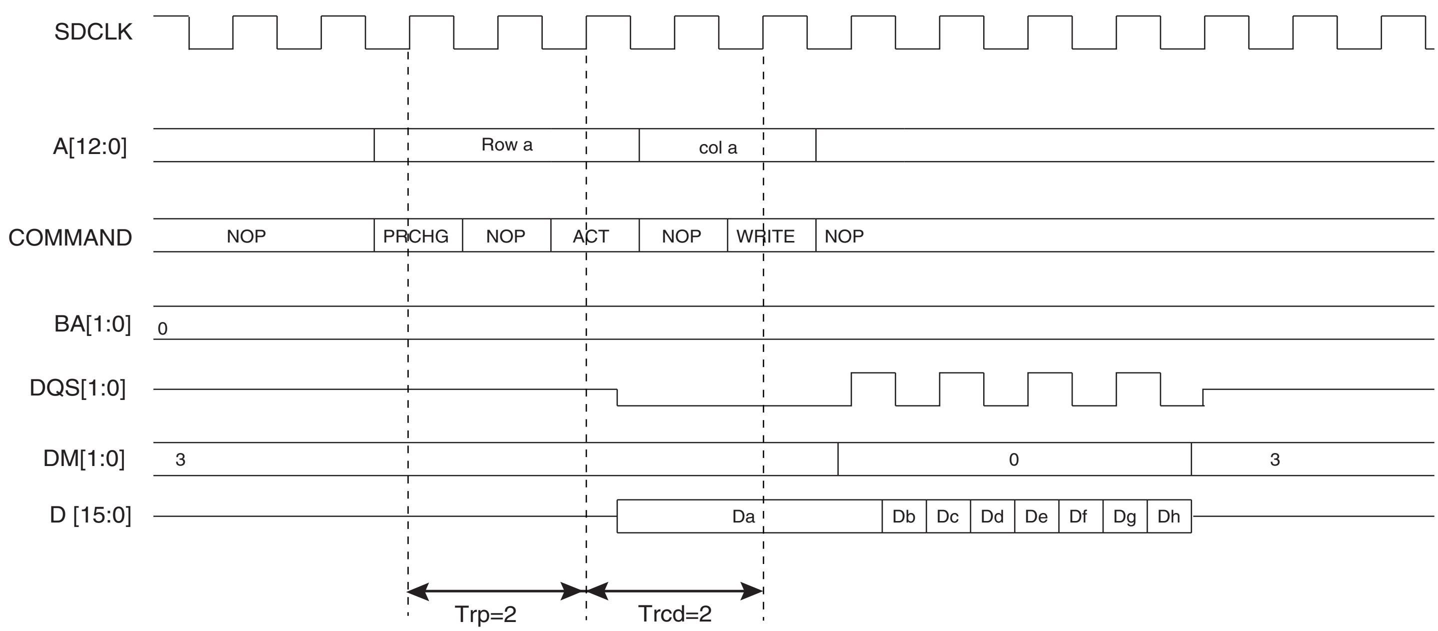
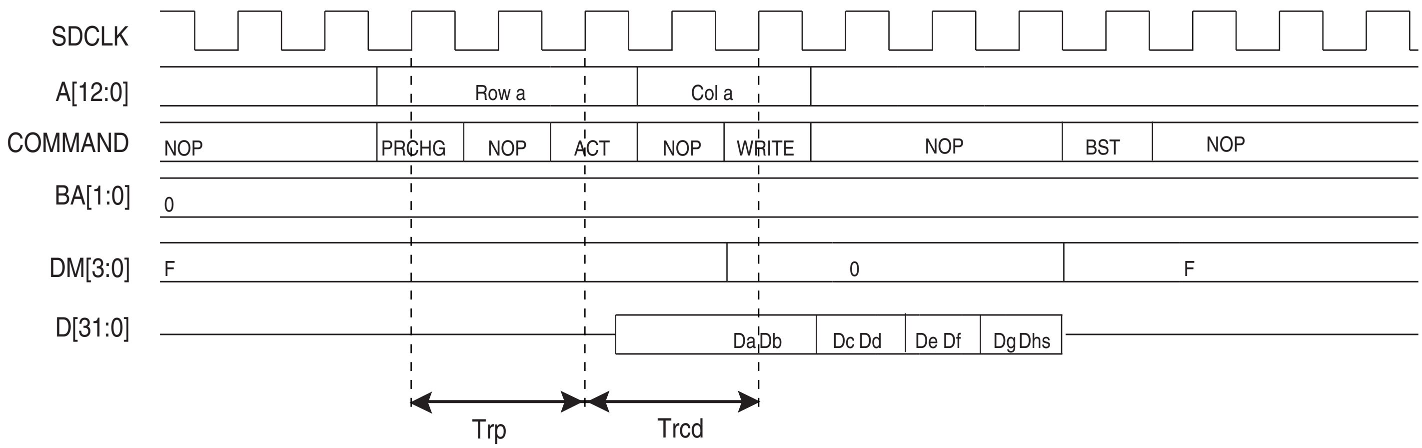
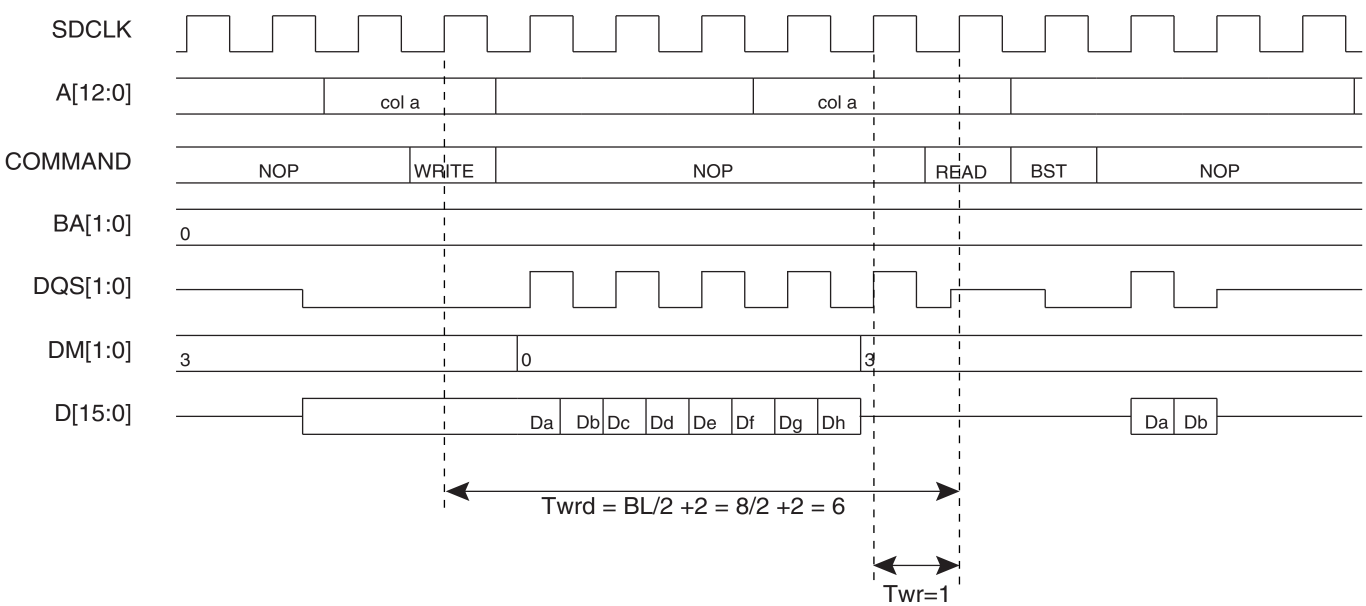


Figure 24-5. Burst Write Access, Row Closed, SDR-SDRAM Devices



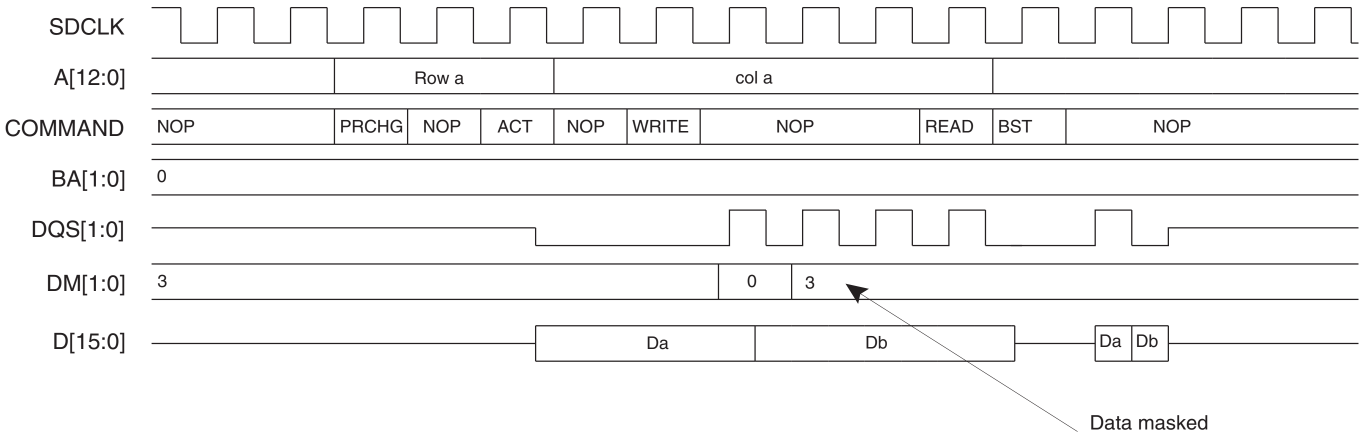
A write command can be followed by a read command. To avoid breaking the current write burst, Twtr/twrd (bl/2 + 2 = 6 cycles) should be met. See Figure 24-6 on page 231.

Figure 24-6. Write Command Followed By A Read Command without Burst Write Interrupt, DDR-SDRAM Devices t



In the case of a single write access, write operation should be interrupted by a read access but DM must be input 1 cycle prior to the read command to avoid writing invalid data. See Figure 24-7 on page 232.

Figure 24-7. SINGLE Write Access Followed By a Read Access, DDR-SDRAM Devices



24.4.2 SDRAM Controller Read Cycle

The DDRSDRC allows burst access or single access in normal mode (mode =000). Whatever access type, the DDRSDRC keeps track of the active row in each bank, thus maximizing performance of the DDRSDRC.

The SDRAM devices are programmed with a burst length equal to 8 which determines the length of a sequential data output by the read command that is set to 8. The latency from read command to data output is equal to 2, 2.5 or 3. This value is programmed during the initialization phase (see Section 24.3.1 “SDR-SDRAM Initialization” on page 225).

To initiate a single access, the DDRSDRC checks if the page access is already open. If row/bank addresses match with the previous row/bank addresses, the controller generates a read command. If the bank addresses are not identical or if bank addresses are identical but the row addresses are not identical, the controller generates a precharge command, activates the new row and initiates a read command. To comply with SDRAM timing parameters, additional clock cycles are inserted between precharge/active (Trp) commands and active/read (Trcd) command. After a read command, additional wait states are generated to comply with cas latency. The DDRSDRC supports a cas latency of two, two and half, and three (2 or 3 clocks delay). As the burst length is fixed to 8, in the case of single access or burst access inferior to 8 data requests, it has to stop the burst otherwise seven or X values could be read. Burst Stop Command (BST) is used to stop output during a burst read.

To initiate a burst access, the DDRSDRC checks the transfer type signal. If the next accesses are sequential read accesses, reading to the SDRAM device is carried out. If the next access is a read non-sequential access, then an automatic page break can be inserted. If the bank addresses are not identical or if bank addresses are identical but the row addresses are not identical, the controller generates a precharge command, activates the new row and initiates a read command. In the case where the page access is already open, a read command is generated.

To comply with SDRAM timing parameters, additional clock cycles are inserted between precharge/active (Trp) commands and active/read (Trcd) commands. The DDRSDRC supports a cas latency of two, two and half, and three (2 or 3 clocks delay). During this delay, the controller uses internal signals to anticipate the next access and improve the performance of the controller. Depending on the latency(2/2.5/3), the DDRSDRC anticipates 2 or 3 read accesses. In the case of burst of specified length, accesses are not anticipated, but if the burst is broken (border, busy mode, etc.), the next access is treated as an incrementing burst of unspecified length, and in function of the latency(2/2.5/3), the DDRSDRAMC anticipates 2 or 3 read accesses.

For a definition of timing parameters, refer to Section 24.6.3 “DDRSDRC Configuration Register” on page 247.

Read accesses to the SDRAM are burst oriented and the burst length is programmed to 8. It determines the maximum number of column locations that can be accessed for a given read command. When the read command is issued, 8 columns are selected. All accesses for that burst take place within these eight columns, meaning that the burst wraps within these 8 columns if the boundary is reached. These 8 columns are selected by addr[13:3]; addr[2:0] is used to select the starting location within the block.

In the case of incrementing burst (INCR/INCR4/INCR8/INCR16), the addresses can cross the 16-byte boundary of the SDRAM device. For example, when a transfer (INCR4) starts at address 0x0C, the next access is 0x10, but since the burst length is programmed to 8, the next access is 0x00. Since the boundary is reached, the burst wraps. The DDRSDRC takes into account this feature of the SDRAM device. In the case of DDR-SDRAM devices, transfers start at address 0x04/0x08/0x0C. In the case of SDR-SDRAM devices, transfers start at address 0x14/0x18/0x1C. Two read commands are issued to avoid wrapping when the boundary is reached. The last read command may generate additional reading (1 read cmd = 4 DDR words or 1 read cmd = 8 SDR words).

To avoid additional reading, it is possible to use the burst stop command to truncate the read burst and to decrease power consumption.

Figure 24-8. Single Read Access, Row Close, Latency= 2, DDR-SDRAM Devices

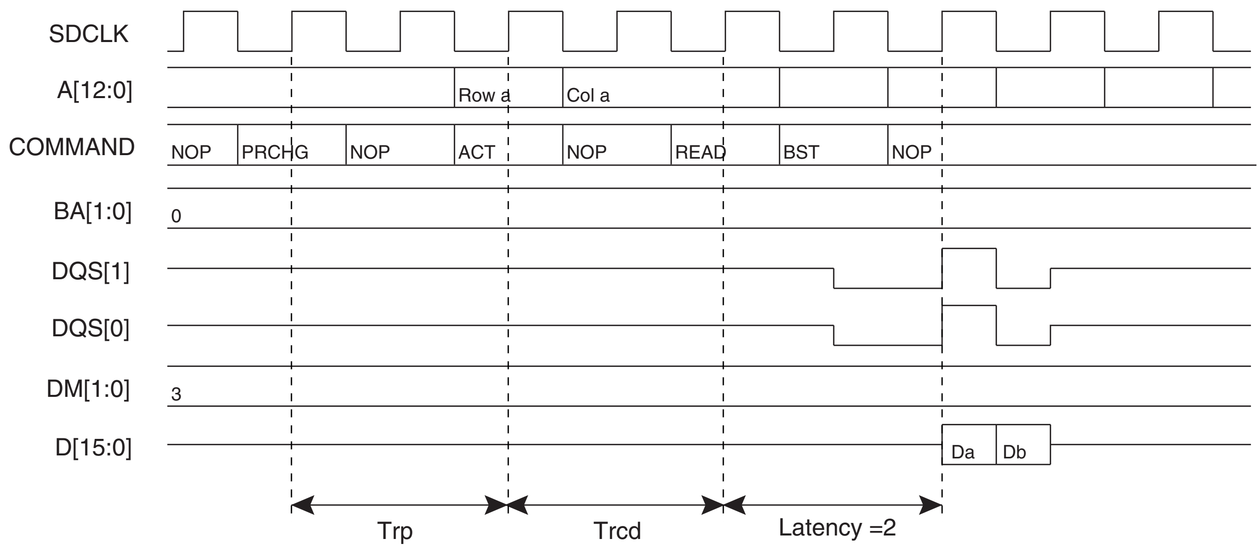


Figure 24-9. Single Read Access, Row Close, Latency= 2, SDR-SDRAM Devices

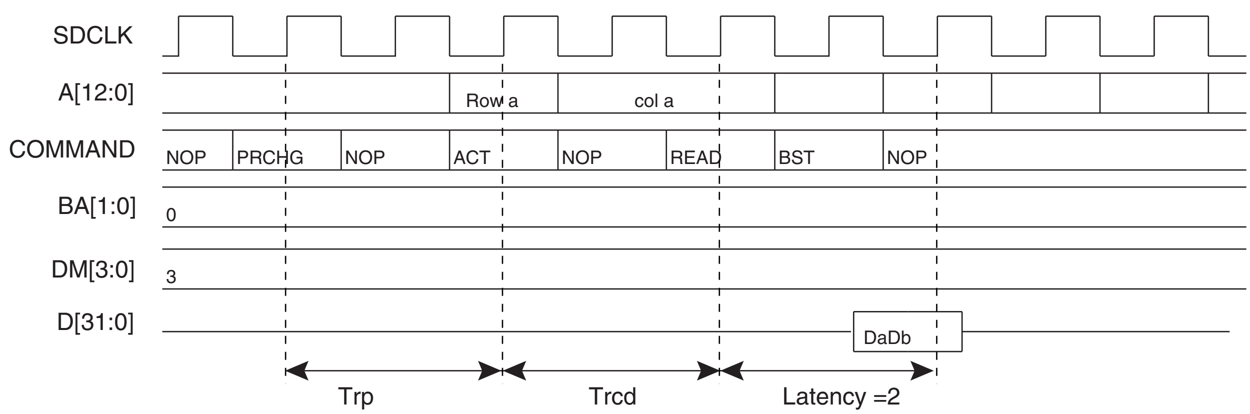


Figure 24-10. Burst Read Access, Latency =2, DDR-SDRAM Devices

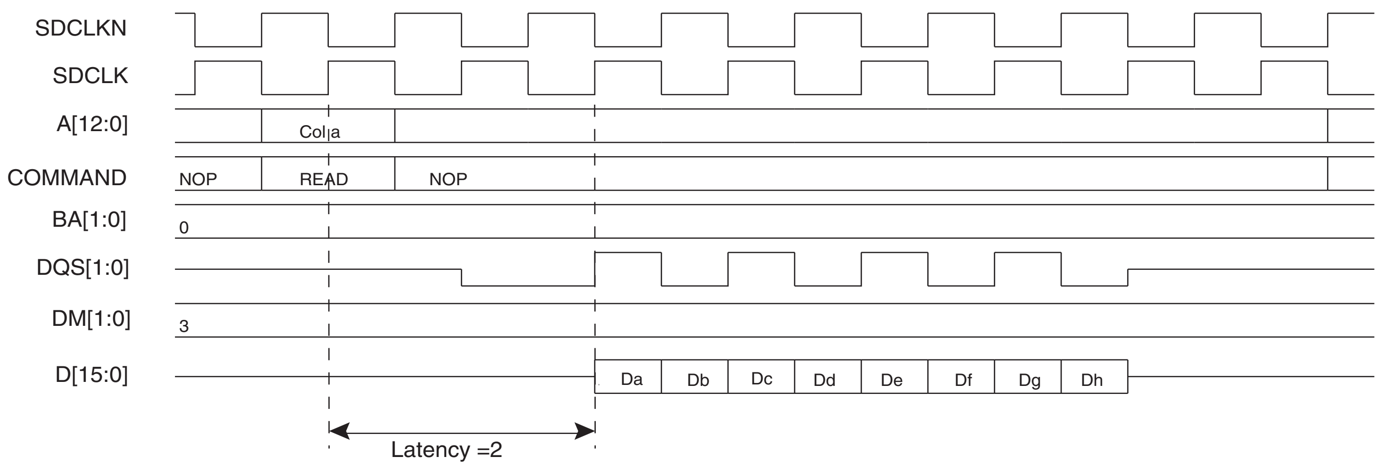
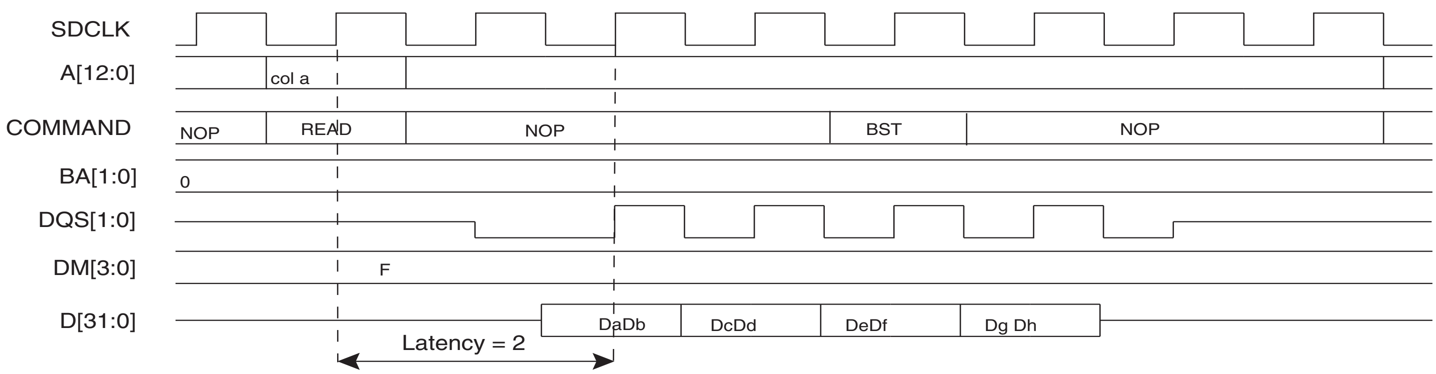


Figure 24-11. Burst Read Access, Latency =2, SDR-SDRAM Devices



24.4.3 Refresh (Auto-refresh Command)

An auto-refresh command is used to refresh the DDRSDRC. Refresh addresses are generated internally by the SDRAM device and incremented after each auto-refresh automatically. The DDRSDRC generates these auto-refresh commands periodically. A timer is loaded with the value in the register DDRSDRAMC_TR that indicates the number of clock cycles between refresh cycles. When the DDRSDRC initiates a refresh of an SDRAM device, internal memory accesses are not delayed. However, if the CPU tries to access the SDRAM device, the slave indicates that the device is busy. A request of refresh does not interrupt a burst transfer in progress.

24.4.4 Power Management

24.4.4.1 Self Refresh Mode

This mode is activated by setting low-power command bits [LPCB] to ‘01’ in the DDRSDRAMC_LPR Register

Self refresh mode is used in power-down mode, i.e., when no access to the SDRAM device is possible. In this case, power consumption is very low. 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 CKE, which remains low. As soon as the SDRAM device is selected, the DDRSDRC provides a sequence of commands and exits self refresh mode.

The DDRSDRC re-enables self refresh mode as soon as the SDRAM device is not selected. It is possible to define when self refresh mode will be enabled by setting the register LPR (see Section 24.6.6 “DDRSDRC Low-power Register” on page 252), timeout command bit:

- 00 = Self refresh mode is enabled as soon as the SDRAM device is not selected
- 01 = Self refresh mode is enabled 64 clock cycles after completion of the last access
- 10 = Self refresh mode is enabled 128 clock cycles after completion of the last access

This controller also interfaces low-power SDRAM. These devices add a new feature: A single quarter, one half quarter or all banks of the SDRAM array can be enabled in self refresh mode. Disabled banks will be not refreshed in self refresh mode. This feature permits to reduce the self refresh current. The extended mode register controls this new feature, it include Temperature Compensated Self Refresh (TSCR), Partial Array Self refresh (PASR) parameters and drives strength (DS). These parameters are set during the initialization phase. 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 any entry into self refresh mode.

The low-power SDR-SDRAM must remain in self refresh mode for a minimum period of TRAS periods and may remain in self refresh mode for an indefinite period. (See Figure 24-12 on page 237)

The low-power DDR-SDRAM must remain in self refresh mode for a minimum of TRFC periods and may remain in self refresh mode for an indefinite period.

Figure 24-12. Self Refresh Mode Entry, Timeout =0

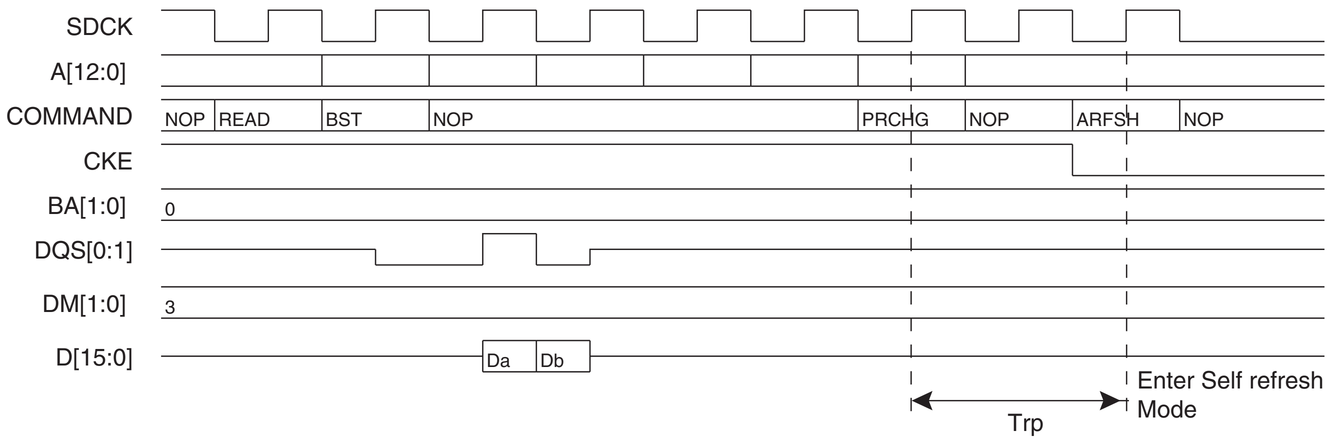


Figure 24-13. Self Refresh Mode Entry, Timeout =1 or 2

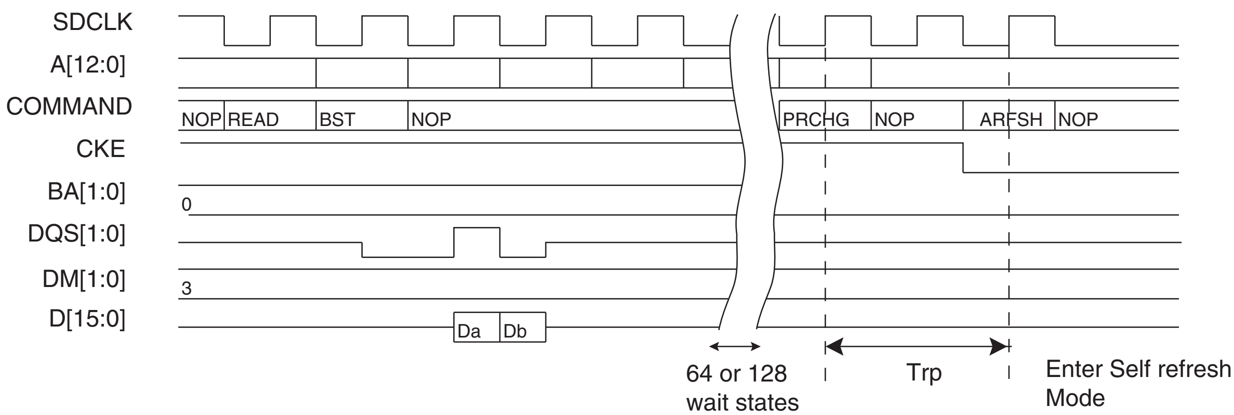
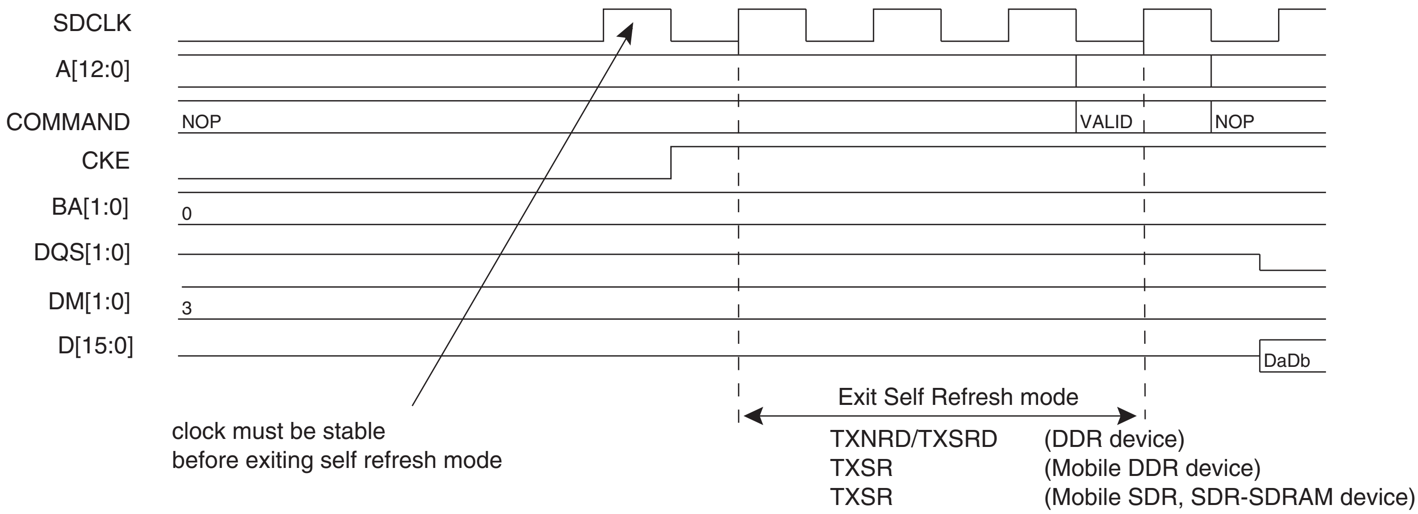


Figure 24-14. Self Refresh Mode Exit



24.4.4.2 Power-down Mode

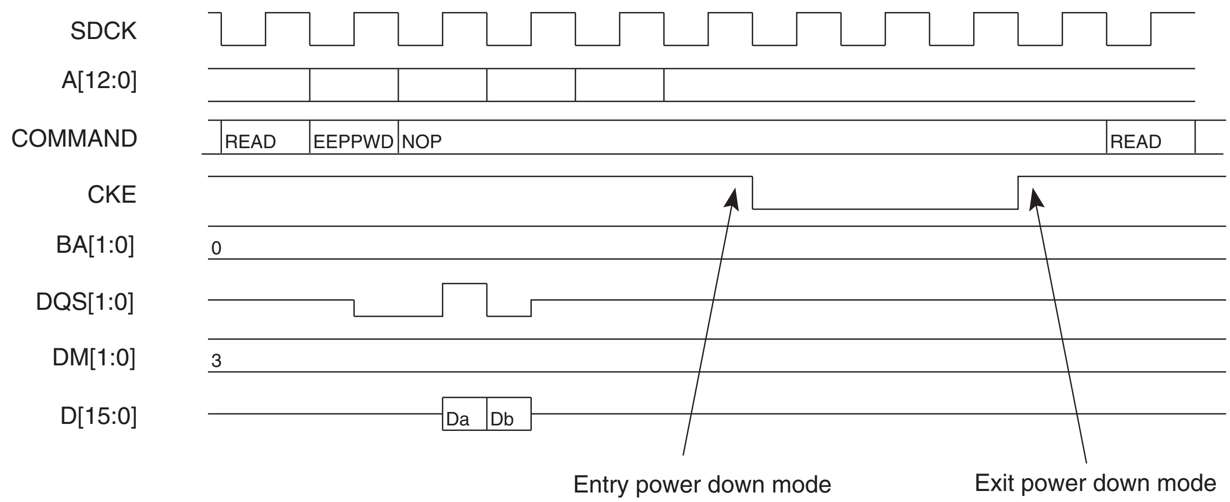
This mode is activated by setting the low-power command bits [LPCB] to '10'.

Power-down mode is used when no access to the SDRAM device is possible. In this mode, power consumption is greater than in self refresh mode. This state is similar to normal mode (No low-power mode/No self refresh mode), but the CKE pin is low and the input and output buffers are deactivated as soon the SDRAM device is no longer accessible. In contrast to self refresh mode, the SDRAM device cannot remain in low-power mode longer than the refresh period (64 ms). As no auto-refresh operations are performed in this mode, the DDRSDRC carries out the refresh operation. In order to exit low-power mode, a NOP command is required in the case of low-power SDR-SDRAM, SDR-SDRAM and DDR-SDRAM devices. In the case of low-power DDR-SDRAM devices, the controller generates a NOP command during a delay of at least TXP. In addition, low-power DDR-SDRAM must remain in power-down mode for a minimum period of TCKE periods.

The exit procedure is faster than in self refresh mode. See Figure 24-15 on page 239. The DDRSDRC returns to power-down mode as soon as the SDRAM device is not selected. It is possible to define when power-down mode is enabled by setting the register LPR, timeout command bit.

- 00 = Power-down mode is enabled as soon as the SDRAM device is not selected
- 01 = Power-down mode is enabled 64 clock cycles after completion of the last access
- 10 = Power-down mode is enabled 128 clock cycles after completion of the last access

Figure 24-15. Power-down Entry/Exit, Timeout =0

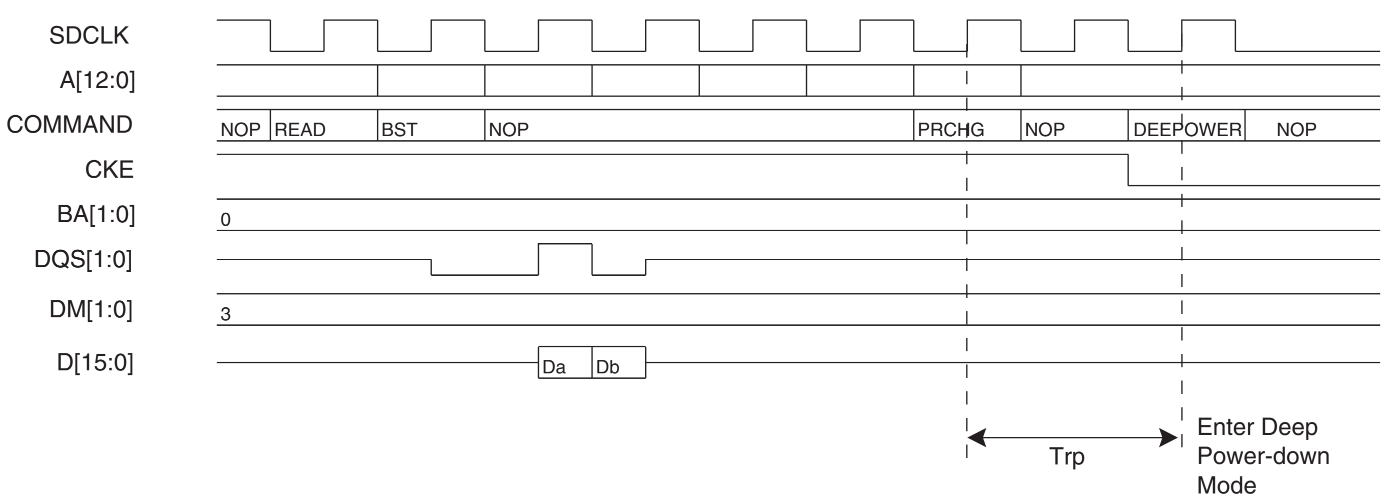


24.4.4.3 Deep Power-down Mode

The deep power-down mode is a new feature of the low-power SDRAM. When this mode is activated, all internal voltage generators inside the device are stopped and all data is lost.

This mode is activated by setting the low-power command bits [LPCB] to '11'. When this mode is enabled, the DDRSDRC leaves normal mode (mode == 000) and the controller is frozen. To exit deep power-down mode, the low-power command bits (LPCB) must be set to "00", an initialization sequence must be generated by software: (see Section 24.3.3 "Low-power DDR-SDRAM Initialization" on page 227).

Figure 24-16. Deep Power-down Mode Entry

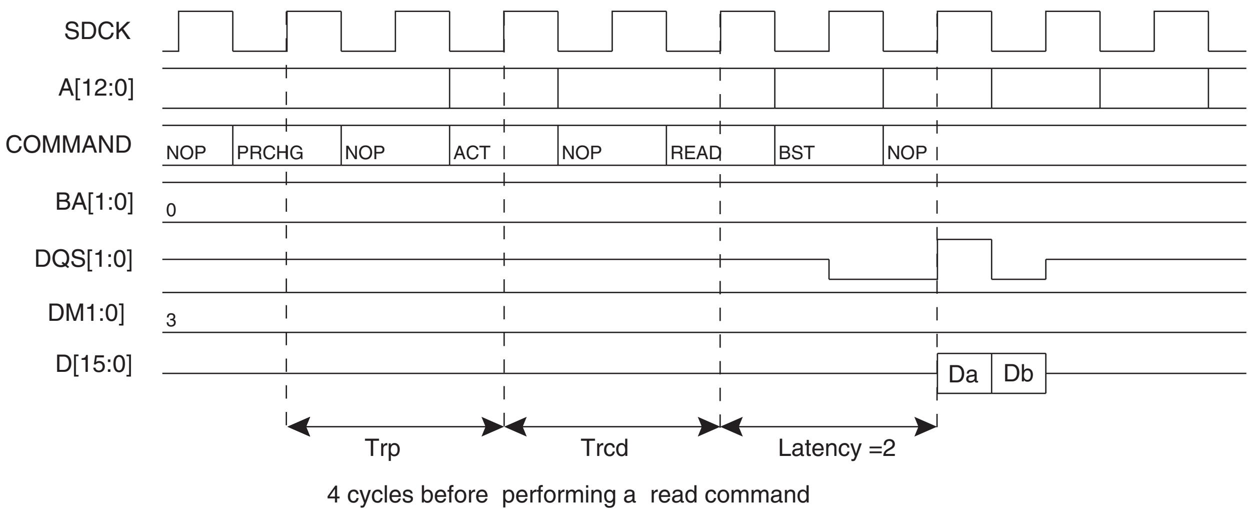


24.4.4.4 Multi-port Functionality

The SDRAM protocol imposes a check of timings prior to performing a read or a write access, thus decreasing the performance of systems. An access to SDRAM is performed if banks and rows are open (or active). To activate a row in a particular bank, it has to de-active the last open row and open the new row. Two SDRAM commands must be performed to open a bank: Pre-charge and Active command with respect to Trp timing. Before performing a read or write command, Trcd timing must checked.

This operation represents a significative loss. (see Figure 24-17)

Figure 24-17. Trp and Trcd Timings



The multi-port controller has been designed to mask these timings and thus improve the bandwidth of the system.

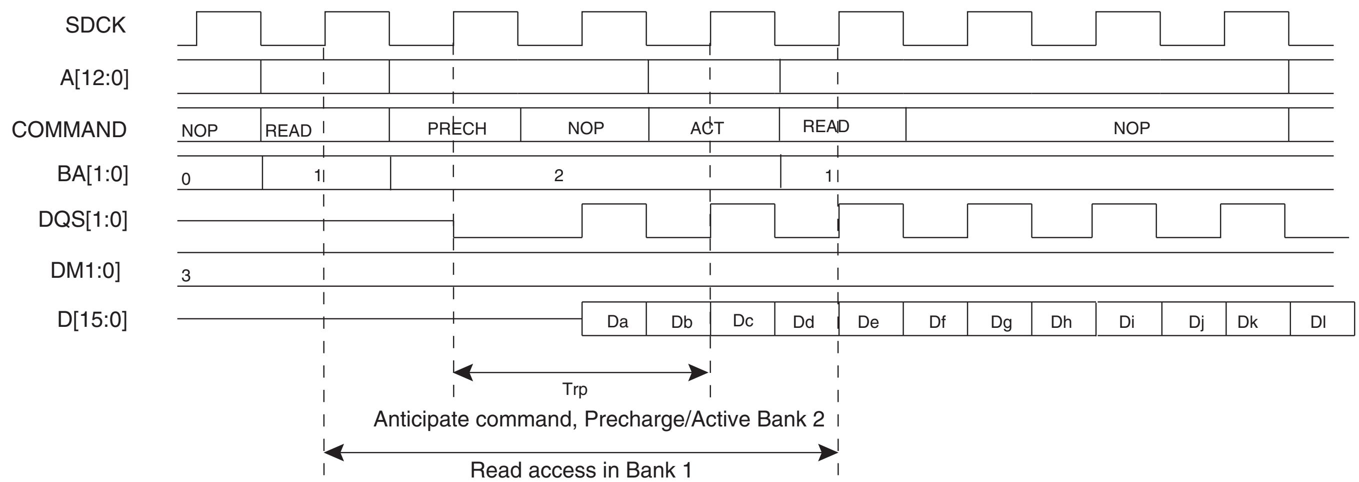
DDRSDRC is a multi-port controller since four masters can simultaneously reach the controller. This feature improves the bandwidth of the system because it can detect four requests on the AHB slave inputs and thus anticipate the commands that follow, PRECHARGE and ACTIVE commands in bank X during current access in bank Y. This allows Trp and Trcd timings to be masked (see Figure 24-18). In the best case, all accesses are done as if the banks and rows were already open. The best condition is met when the four masters work in different banks. In the case of four simultaneous read accesses, when the four banks and associated rows are open, the controller reads with a continuous flow and masks the cas latency for each different access. To allow a continuous flow, the read command must be set at 2 or 3 cycles (cas latency) before the end of current access. This requires that the scheme of arbitration changes since the round-robin arbitration cannot be respected. If the controller anticipates a read access, and thus before the end of current access a master with a high priority arises, then this master will not serviced.

The arbitration mechanism reduces latency when conflicts occur, i.e., when two or more masters try to access the SDRAM device at the same time.

The arbitration type is round-robin arbitration. This algorithm dispatches the requests from different masters to the SDRAM device in a round-robin manner. If two or more master requests arise at the same time, the master with the lowest number is serviced first, then the others are serviced in a round-robin manner. To avoid burst breaking and to provide the maximum throughput for the SDRAM device, arbitration may only take place during the following cycles:

1. Idle cycles: When no master is connected to the SDRAM device.
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 bursts of defined length, predicted end of burst matches the size of the transfer. For bursts of undefined length, predicted end of burst is generated at the end of each four beat boundary inside the INCR transfer.
4. Anticipated Access: When an anticipate read access is done while current access is not complete, the arbitration scheme can be changed if the anticipated access is not the next access serviced by the arbitration scheme.

Figure 24-18. Anticipate Precharge/Active Command in Bank 2 during Read Access in Bank 1



24.5 Software Interface/SDRAM Organization, Address Mapping

The SDRAM address space is organized into banks, rows and columns. The DDRSDRC maps different memory types depending on the values set in the DDRSDRC Configuration Register. See Section 24.6.3 "DDRSDRC Configuration Register" on page 247. The following figures illustrate the relation between CPU addresses and columns, rows and banks addresses for 16-bit memory data bus widths and 32-bit memory data bus widths.

The DDRSDRC supports address mapping in linear mode.

Linear mode is a method for address mapping where banks alternate at each last SDRAM page of current bank.

The DDRSDRC makes the SDRAM devices access protocol transparent to the user. Table 24-1 to Table 24-7 illustrate the SDRAM device memory mapping seen by the user in correlation with the device structure. Various configurations are illustrated.

24.5.1 SDRAM Address Mapping for 16-bit Memory Data Bus Width[(1)]

Table 24-1. Linear Mapping for SDRAM Configuration, 2K Rows, 512/1024/2048/4096 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[8:0]									M0
				Bk[1:0]		Row[10:0]											Column[9:0]										M0
			Bk[1:0]		Row[10:0]											Column[10:0]											M0
		Bk[1:0]		Row[10:0]											Column[11:0]												M0

Table 24-2. Linear Mapping for SDRAM Configuration: 4K Rows, 512/1024/2048/4096 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[8:0]									M0
			Bk[1:0]		Row[11:0]												Column[9:0]										M0
		Bk[1:0]		Row[11:0]												Column[10:0]											M0
	Bk[1:0]		Row[11:0]												Column[11:0]												M0

Table 24-3. Linear Mapping for SDRAM Configuration: 8K Rows, 512/1024/2048/4096 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[8:0]									M0
		Bk[1:0]		Row[12:0]													Column[9:0]										M0
	Bk[1:0]		Row[12:0]													Column[10:0]											M0
Bk[1:0]		Row[12:0]													Column[11:0]												M0

Table 24-4. Linear Mapping for SDRAM Configuration: 16K Rows, 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[13:0]														Column[8:0]									M0
	Bk[1:0]		Row[13:0]														Column[9:0]										M0
Bk[1:0]		Row[13:0]														Column[10:0]											M0

Note: 1. SDR-SDRAM devices with eight columns in 16-bit mode are not supported.

24.5.2 SDR-SDRAM Address Mapping for 32-bit Memory Data Bus Width

Table 24-5. SDR-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 24-6. SDR-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 24-7. SDR-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

24.6 DDR/SDR SDRAM Controller (DDRSDRC) User Interface

The User Interface is connected to the APB bus.

The DDRSDRC is programmed using the registers listed in Table 24-8.

Table 24-8. Register Mapping

Offset	Register	Name	Access	Reset
0x00	DDRSDRC Mode Register	DDRSDRC_MR	Read-write	0x00000000
0x04	DDRSDRC Refresh Timer Register	DDRSDRC_RTR	Read-write	0x00000000
0x08	DDRSDRC Configuration Register	DDRSDRC_CR	Read-write	0x024
0x0C	DDRSDRC Timing0 Register	DDRSDRC_T0PR	Read-write	0x20227225
0x10	DDRSDRC Timing1 Register	DDRSDRC_T1PR	Read-write	0x3c80808
0x18	DDRSDRC Low-power Register	DDRSDRC_LPR	Read-write	0x0
0x1C	DDRSDRC Memory Device Register	DDRSDRC_MD	Read-write	0x10
0x20	DDRSDRC DLL Information Register	DDRSDRC_DLL	Read	0x00000001
0x28-0xE8	Reserved			

24.6.1 DDRSDRC Mode Register

Name: DDRSDRC_MR

Address: 0xFFFFE600

Access: Read-write

Reset: See Table 24-8

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: DDRSDRAMC Command Mode**

This field defines the command issued by the DDRSDRC when the SDRAM device is accessed. This register is used to initialize the SDRAM device and to activate deep power-down mode.

MODE	Description
000	Normal Mode. Any access to the DDRSDRAMC will be decoded normally. To activate this mode, command must be followed by a write to the SDRAM.
001	The DDRSDRC issues a **NOP** Command when the SDRAM device is accessed regardless of the cycle. To activate this mode, command must be followed by a write to the SDRAM.
010	The DDRSDRC issues an **All Banks Precharge** Command when the SDRAM device is accessed regardless of the cycle. To activate this mode, command must be followed by a write to the SDRAM.
011	The DDRSDRC issues a **Load mode Register** command when the SDRAM device is accessed regardless of the cycle. To activate this mode, command must be followed by a write to the SDRAM.
100	The DDRSDRC issues an **Auto Refresh** Command when the SDRAM device is accessed regardless of the cycle. Previously, an All Banks Precharge Command must be issued. To activate this mode, command must be followed by a write to the SDRAM.
101	The DDRSDRC issues an **Extended Load Mode** register command when the SDRAM device is accessed regardless of the cycle. To activate this mode, the "Extended Load Mode" register command must be followed by a write to the SDRAM. The write in the SDRAM must be done in the appropriate bank.
110	**Deep power mode**: Access to deep power-down mode
111	Reserved

24.6.2 DDRSDRC Refresh Timer Register

Name: DDRSDRC_RTR

Address: 0xFFFFE604

Access: Read-write

Reset: See Table 24-8

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: DDRSDRC Refresh Timer Count**

This 12-bit field is loaded into a timer which generates the refresh pulse. Each time the refresh pulse is generated, a refresh sequence is initiated.

SDRAM devices require a refresh of all rows every 64 ms. The value to be loaded depends on the DDRSDRC clock frequency (MCK: Master Clock) and the number of rows in the device.

For example, for an SDRAM with 8192 rows and a 100 MHz Master clock, the value of Refresh Timer Count bit is programmed: $(((64 \times 10^{-3})/8192) \times 100 \times 10^{6} = 781$ or 0x030D.

24.6.3 DDRSDRC Configuration Register

Name: DDRSDRC_CR

Address: 0xFFFFE608

Access: Read-write

Reset: See Table 24-8

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	DIC/DS

7	6	5	4	3	2	1	0
DLL	CAS			NR		NC	

- **NC: Number of Column Bits.**

The reset value is 9 column bits.

SDR-SDRAM devices with eight columns in 16-bit mode (b16mode ==1) are not supported.

NC	DDR - Column bits	SDR - Column bits
00	9	8
01	10	9
10	11	10
11	12	11

- **NR: Number of Row Bits**

The reset value is 12 row bits.

NR	Row bits
00	11
01	12
10	13
11	14

- **CAS: CAS Latency**

The reset value is 2 cycles.

cas	DDR-SDRAM Cas Latency	SDR-SDRAM Cas Latency
000	Reserved	Reserved
001	Reserved	Reserved
010	2	2

cas	DDR-SDRAM Cas Latency	SDR-SDRAM Cas Latency
011	3	3
100	Reserved	Reserved
101	Reserved	Reserved
110	2.5	Reserved
111	Reserved	Reserved

- **DLL: Reset DLL**

Reset value is 0.

This field defines the value of Reset DLL.

0: Disable DLL reset

1: Enable DLL reset

This value is used during the power-up sequence. **This field is found only in DDR-SDRAM devices.**

- **DIC/DS: Output Driver Impedance Control**

Reset value is 0.

This field defines the output drive strength.

0: Normal driver strength

1: Weak driver strength

This value is used during the power-up sequence. This parameter is found in the datasheet as DIC or DS.

This field is found only in DDR-SDRAM devices.

24.6.4 DDRSDRC Timing 0 Parameter Register

Name: DDRSDRC_T0PR

Address: 0xFFFFE60C

Access: Read-write

Reset: See Table 24-8

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

23	22	21	20	19	18	17	16
TRRD				TRP			

15	14	13	12	11	10	9	8
TRC				TWR			

7	6	5	4	3	2	1	0
TRCD				TRAS			

- **TRAS: Active to Precharge Delay**

Reset Value is 5 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.

- **TRCD: Row to Column Delay**

Reset Value is 2 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.

- **TWR: Write Recovery Delay**

Reset value is 2.

This field defines the Write Recovery Time in number of cycles. Number of cycles is between 1 and 15.

- **TRC: Row Cycle Delay**

Reset value is 7 cycles.

This field defines the delay between an Activate command and Refresh command in number of cycles. Number of cycles is between 0 and 15

- **TRP: Row Precharge Delay**

Reset Value is 2 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.

- **TRRD Active bankA to Active bankB**

Reset value is 2.

This field defines the delay between an Active command in BankA and an active command in bankB in number of cycles. Number of cycles is between 1 and 15.

- **TWTR: Internal Write to Read Delay**

Reset value is 0.

This field defines the internal write to read command Time in number of cycles. Number of cycles is between 1 and 2.

- **TMRD: Load Mode Register Command to Active or Refresh Command**

Reset Value is 2 cycles.

This field defines the delay between an Load mode register command and an active or refresh command in number of cycles. Number of cycles is between 0 and 15.

24.6.5 DDRSDRC Timing 1 Parameter Register

Name: DDRSDRC_T1PR

Address: 0xFFFFE610

Access: Read-write

Reset: See Table 24-8

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

23	22	21	20	19	18	17	16
TXSRD							

15	14	13	12	11	10	9	8
TXSNR							

7	6	5	4	3	2	1	0
–	–	–	TRFC				

- **TRFC: Row Cycle Delay**

Reset Value is 8 cycles.

This field defines the delay between a Refresh and an Activate command or Refresh command in number of cycles. Number of cycles is between 0 and 31

- **TXSNR: Exit Self Refresh Delay to Non Read Command**

Reset Value is 8 cycles.

This field defines the delay between cke set high and a non Read Command in number of cycles. Number of cycles is between 0 and 15. This field is used for SDR-SDRAM and DDR-SDRAM devices. In the case of SDR-SDRAM devices and low-power DDR-SDRAM, this field is equivalent to TXSR timing.

- **TXSRD: Exit Self Refresh Delay to Read Command**

Reset Value is C8.

This field defines the delay between cke set high and a Read Command in number of cycles. Number of cycles is between 0 and 255 cycles.This field is unique to DDR-SDRAM devices.

- **TXP: Exit Power-down Delay to First Command**

Reset Value is 3.

This field defines the delay between cke set high and a Valid Command in number of cycles. Number of cycles is between 0 and 15 cycles. This field is unique to low-power DDR-SDRAM devices.

24.6.6 DDRSDRC Low-power Register

Name: DDRSDRC_LPR

Address: 0xFFFFE618

Access: Read-write

Reset: See Table 24-8

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		TCR	

7	6	5	4	3	2	1	0
–	PASR				CLK_FR	LPCB	

- **LPCB: Low-power Command Bit**

Reset value is “00”.

00: Low-power Feature is inhibited: no power-down, self refresh and Deep power mode are issued to the SDRAM device.

01: The DDRSDRC issues a Self Refresh Command to the SDRAM device, the clock(s) is/are de-activated and the CKE signal is set low. The SDRAM device leaves the self refresh mode when accessed and enters it after the access.

10: The DDRSDRC issues a Power-down Command to the SDRAM device after each access, the CKE signal is set low. The SDRAM device leaves the power-down mode when accessed and enters it after the access.

11: The DDRSDRC issues a Deep Power-down Command to the low-power SDRAM device.**This mode is unique to low-power SDRAM devices.**

- **CLK_FR: Clock Frozen Command Bit**

Reset value is “0”.

This field sets the clock low during power-down mode or deep power-down mode. Some SDRAM devices do not support freezing the clock during power-down mode or deep power-down mode. Refer to the SDRAM device datasheet for details on this.

1: Clock(s) is/are frozen.

0: Clock(s) is/are not frozen.

- **PASR: Partial Array Self Refresh**

Reset value is “0”.

This field is unique to low-power SDRAM. It is used 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.

The values of this field are dependant on low-power SDRAM devices.

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

- **TCR: Temperature Compensated Self Refresh**

Reset value is “0”.

This field is unique to low-power SDRAM. It is used to program the refresh interval during self refresh mode, depending on the case temperature of the low-power SDRAM.

The values of this field are dependent on low-power SDRAM devices.

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

- **DS: Drive Strength**

Reset value is “0”.

This field is unique to low-power SDRAM. It selects the driver strength of SDRAM output.

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

- **TIMEOUT**

Reset value is “00”.

This field defines 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

24.6.7 DDRSDRC Memory Device Register

Name: DDRSDRC_MD

Address: 0xFFFFE61C

Access: Read-write

Reset: See Table 24-8

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

- **MD Memory Device**

Indicates the type of memory used.

Reset value is for SDR-SDRAM device.

00: SDR-SDRAM

01: Low-power SDR-SDRAM 1

10: DDR-SDRAM

11: Low-power DDR-SDRAM

- **DBW Data Bus Width**

Reset value is 16 bits.

0: Data bus width is 32 bits (reserved for SDR-SDRAM device).

1: Data bus width is 16 bits.

24.6.8 DDRSDRC DLL Information

Name: DDRSDRC_DLL

Address: 0xFFFFE620

Access: Read

Reset: See Table 24-8

31	30	29	28	27	26	25	24
SDCVAL							

23	22	21	20	19	18	17	16
SDVAL							

15	14	13	12	11	10	9	8
MDVAL							

7	6	5	4	3	2	1	0
–	–	SDERF	SDCUDF	SDCOVF	MDOVF	MDDEC	MDINC

The DLL logic is internally used by the controller in order to delay DQS inputs.This is necessary to center the strobe time and the data valid window.

- **MDINC: DLL Master Delay Increment**

0: The DLL is not incrementing the Master delay counter.

1: The DLL is incrementing the Master delay counter.

- **MDDEC: DLL Master Delay Decrement**

0: The DLL is not decrementing the Master delay counter.

1: The DLL is decrementing the Master delay counter.

- **MDOVF: DLL Master Delay Overflow Flag**

0: The Master delay counter has not reached its maximum value, or the Master is not locked yet.

1: The Master delay counter has reached its maximum value, the Master delay counter increment is stopped and the DLL forces the Master lock. If this flag is set, it means the DDRSDRC clock frequency is too low compared to Master delay line number of elements.

- **SDCOVF: DLL Slave Delay Correction Overflow Flag.**

0: Due to the correction, the Slave delay counter has not reached its maximum value, or the Slave is not locked yet.

1: Due to the correction, the Slave delay counter has reached its maximum value, the correction is not optimal because it is not applied entirely.

- **SDCUDF: DLL Slave Delay Correction Underflow Flag**

0: Due to the correction, the Slave delay counter has not reached its minimum value, or the Slave is not locked yet.

1: Due to the correction, the Slave delay counter has reached its minimum value, the correction is not optimal because it has not been entirely applied.

- **SDERF**: **DLL Slave Delay Correction Error Flag**

0: The DLL has succeeded in computing the Slave delay correction, or the Slave is not locked yet.

1: The DLL has not succeeded in computing the Slave delay correction.

- **MDVAL**: **DLL Master Delay Value**

Value of the Master delay counter.

- **SDVAL**: **DLL Slave Delay Value**

Value of the Slave delay counter.

- **SDCVAL**: **DLL Slave Delay Correction Value**

Value of the correction applied to the Slave delay.

25. Burst Cellular RAM Controller (BCRAMC)

25.1 Description

The Burst Cellular RAM Controller (BCRAMC) is a synchronous pseudo-static RAM memory controller, it supports Cellular RAM device version 1.0, 1.5 and 2.0.

The BCRAMC extends the memory capabilities of a chip by providing the interface to an external 16- or 32-bit Cellular RAM device. The page size support ranges from 64 to 512. It supports byte, half-word and word accesses.

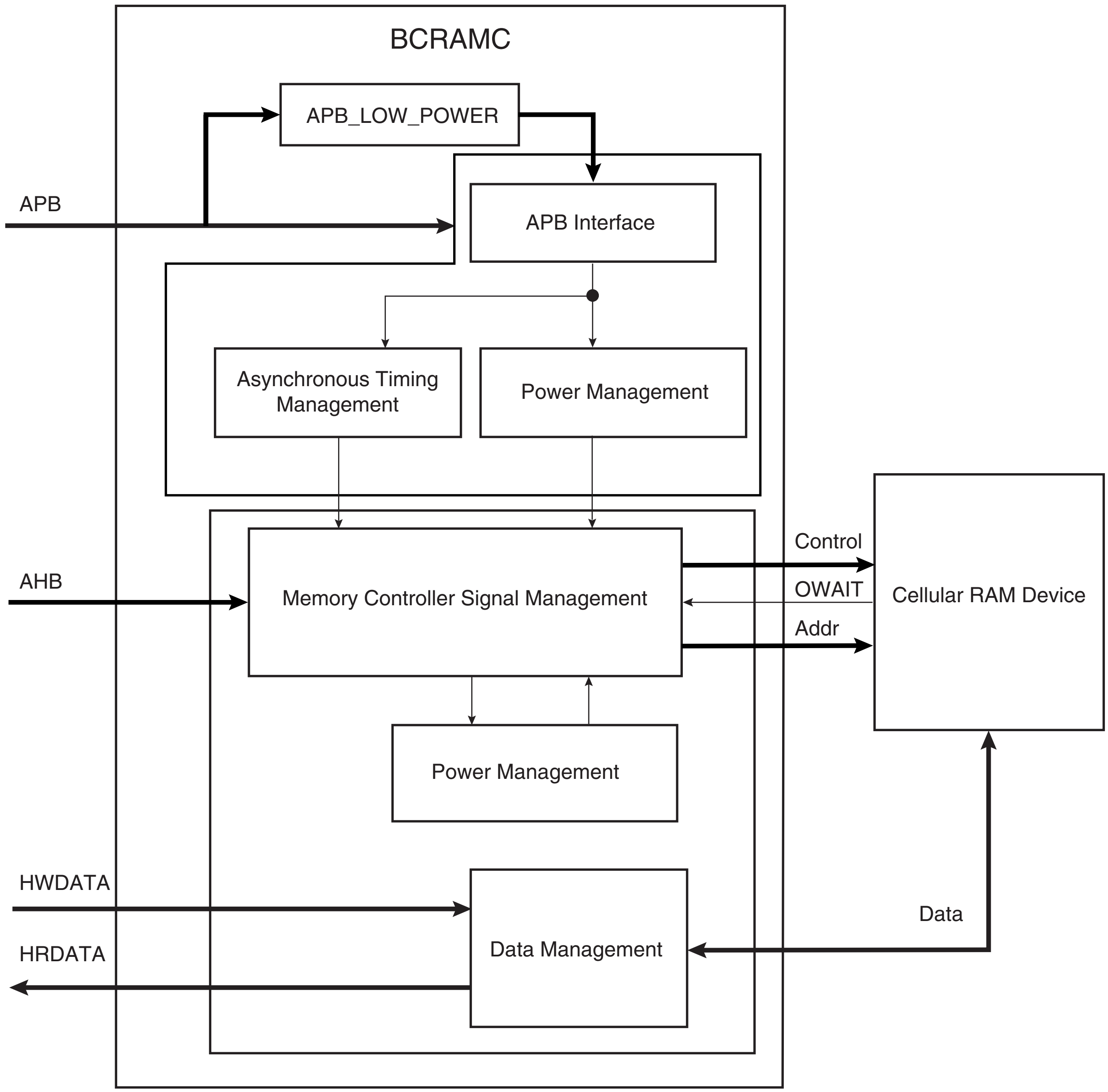
The BCRAMC supports continuous read or write burst. It supports a latency of 2, 3 for Cellular RAM version 1.0 and a latency of 2, 3, 4, 5, 6 for Cellular RAM version 1.5 and 2.0 and optimizes the read/write access depending on the frequency.

Standby and deep power down modes minimize power consumption on the Cellular RAM device.

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

25.2 BCRAMC Block Diagram

Figure 25-1. BCRAMC Block Diagram



25.3 Product Dependencies

25.3.1 Cellular RAM Initialization

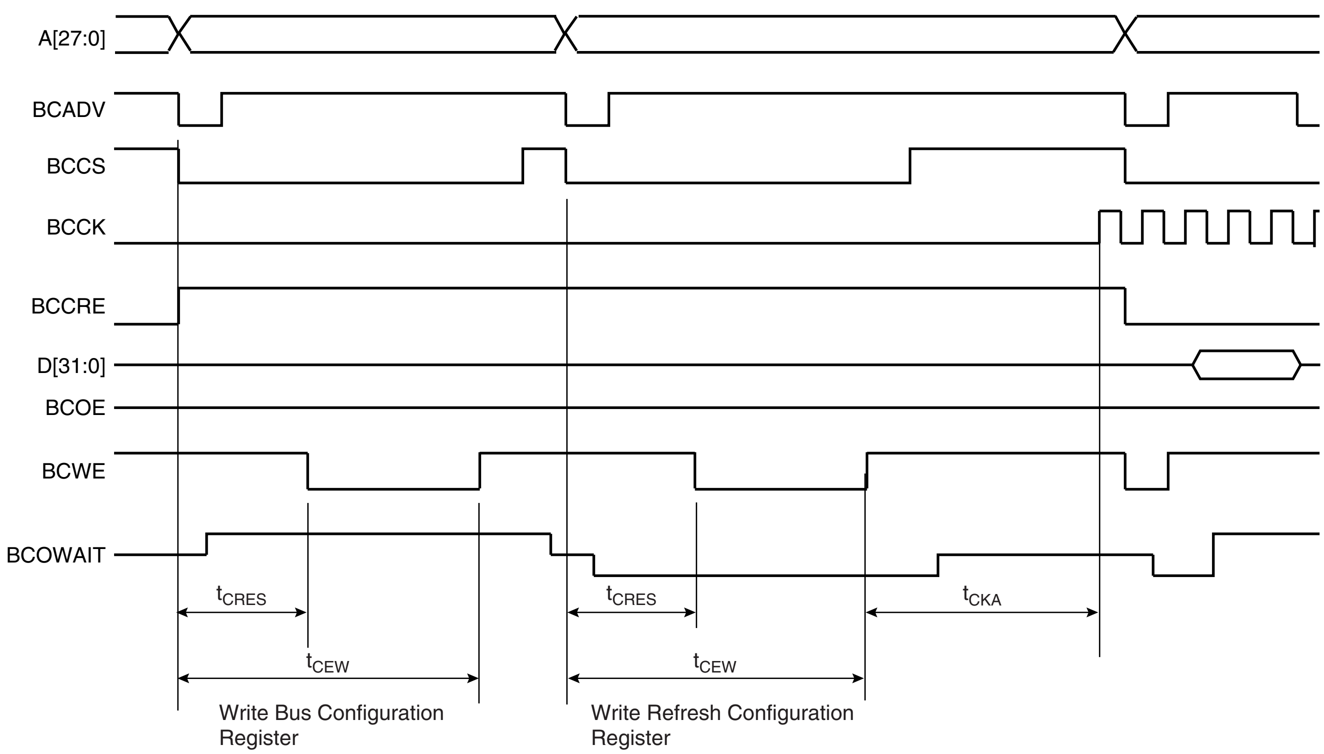
The Cellular RAM devices are initialized by the following sequence:

5. Minimum pause of 150 µs is provided to precede any signal toggle.
6. The Cellular RAM memory type must be set in the BCRAMC Memory Device Register.
7. Temperature-compensated self refresh (TCSR) and partial array self refresh (PASR) must be set in the BCRAMC Low Power register.
8. Asynchronous timings (TCKA, TCRE) must be set in the BCRAMC Timing Register.
9. Cellular RAM features must be set in the HBCRAMC Configuration Register:
   – number rows, latency, drive strength (DS), the data bus width and cram_enabled bit must be high.

Perform a write to the Cellular RAM device and the Bus Configuration Register (BCR) and Refresh Configuration Register (RCR) are programmed automatically.

After initialization, the Cellular RAM devices are fully functional.

Figure 25-2. Initialization Sequence



25.4 Functional Description

25.4.1 BCRAMC Overview

The BCRAMC is a synchronous cellular RAM controller, it does not support asynchronous access and mode page. Some version 1.0 devices which support only these features cannot be driven.

The BCRAMC drives 16-bit memory devices but, in this mode, it does not support byte read/write bursts. All byte burst accesses are treated as a single access because BCRAMC is set in continuous burst where16-bit data are accessed sequentially. To support byte read/write bursts, complex logic should be added to transform byte burst to half-word burst.

The BCRAMC drives 32-bit memory devices but, in this mode, it does not support byte/half-word read/write bursts. All byte or half-word burst accesses are treated as single access because BCRAMC is set in continuous burst where 32-bit data are accessed sequentially. To support byte/half-word read/write bursts, complex logic should be added to transform byte/half-word bursts to word bursts.

The BCRAMC supports busy transfer. This kind of access is treated as early burst termination. The controller performances are decreased because after a busy transfer, a new initial burst operation (adv is low) will be generated.

25.4.2 BCRAMC Write Cycle

The BCRAMC provides burst access or single access.

The Cellular RAM device is programmed with a continuous burst length.

The latency from write command to data input is a function of the Cellular RAM device.

Version 1.0 write latency is equal to the latency programmed in the bus configuration register during the initialization sequence or in the worst case, it is equal to the refresh collision delay. With version 1.0, the BCRAMC must monitor owait signal to detect any conflict of refresh collision during write accesses. The write latency is not constant.

In the case of version 1.5 and 2.0, write latency is equal to the latency programmed in the bus configuration register. Write latency always uses fixed latency. The BCRAMC does not monitor owait signal during write accesses.

To initiate a single access, the BCRAMC generates an initial burst write command. To comply with Cellular RAM timing parameters, additional clock cycles are inserted to check programmed latency. In the case of Cellular RAM version 1.0, the owait signal is monitored to detect a refresh collision. As soon as owait signal is high, data is accepted into the device and write access is achieved. In the case of Cellular RAM version 1.5 and 2.0, owait signal is not monitored and write access is performed as soon as latency is checked.

As the burst length is fixed to continuous, in the case of single access, it has to stop the burst else invalid values can be written. To interrupt the write operation, chip select (CS) must be set to 1 or an initial burst read/write command can be initiated to interrupt current access if the next access is a Cellular RAM access.

To initiate a burst access, the BCRAMC uses the transfer type signal provided by the master requesting the access. If the next access is a sequential write access, writing to the Cellular RAM device is carried out. If the next access is a write sequential access, but the current access is to a boundary page, then an automatic page break is inserted and the Cellular RAM controller generates an initial burst write command to finish access. To comply with Cellular RAM timing parameters, additional clock cycles are inserted to check programmed latency. A single access owait signal is monitored, or not, in function of the Cellular RAM version.

Write accesses to the Cellular RAM are burst oriented, the programmed burst length is continuous burst. This feature makes it possible to start at a specified address and burst through the entire memory. It is very useful for incrementing bursts (INCR/INCR4/INCR8/INCR16), as soon as the burst command init (latch burst start address) is initiated and latency is checked, at each BCCK rising a data is written.

In the case of a wrapping burst (WRAP4/WRAP8/WRAP16), the addresses can cross the boundary of the current transfer. For example, when a transfer (WRAP4) starts to address 0x0C, the next access will be 0x00, but the burst length being programmed to continuous burst in the next access should be 0x10, the burst does not wrap automatically. The BCRAMC takes account of this feature and in the case of a transfer starting from address 0x04/0x08/0x0C, two initial burst write commands will be issued to wrap when boundary is reached. The last initial burst write command will be interrupted by a BCCS (chip select) set to high or by another initial burst read/write command to do the next access in the Cellular RAM device, if an access is pending.

Figure 25-3. Single Write Access, No Refresh Collision

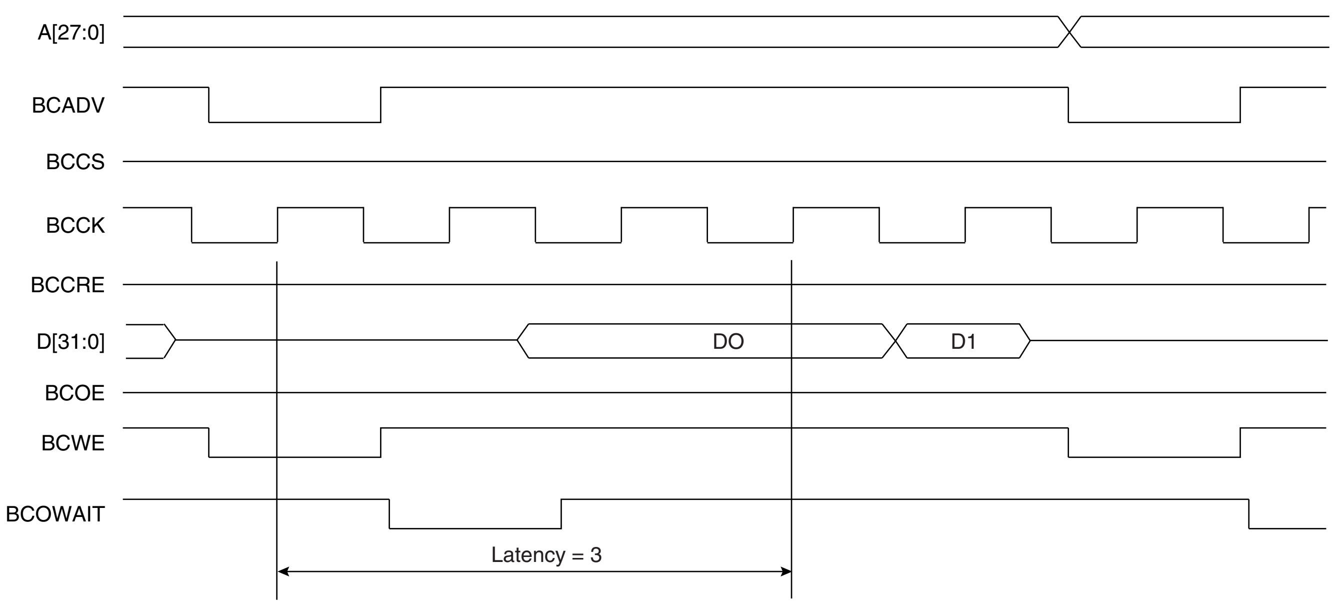


Figure 25-4. Single Write Access with Refresh Collision

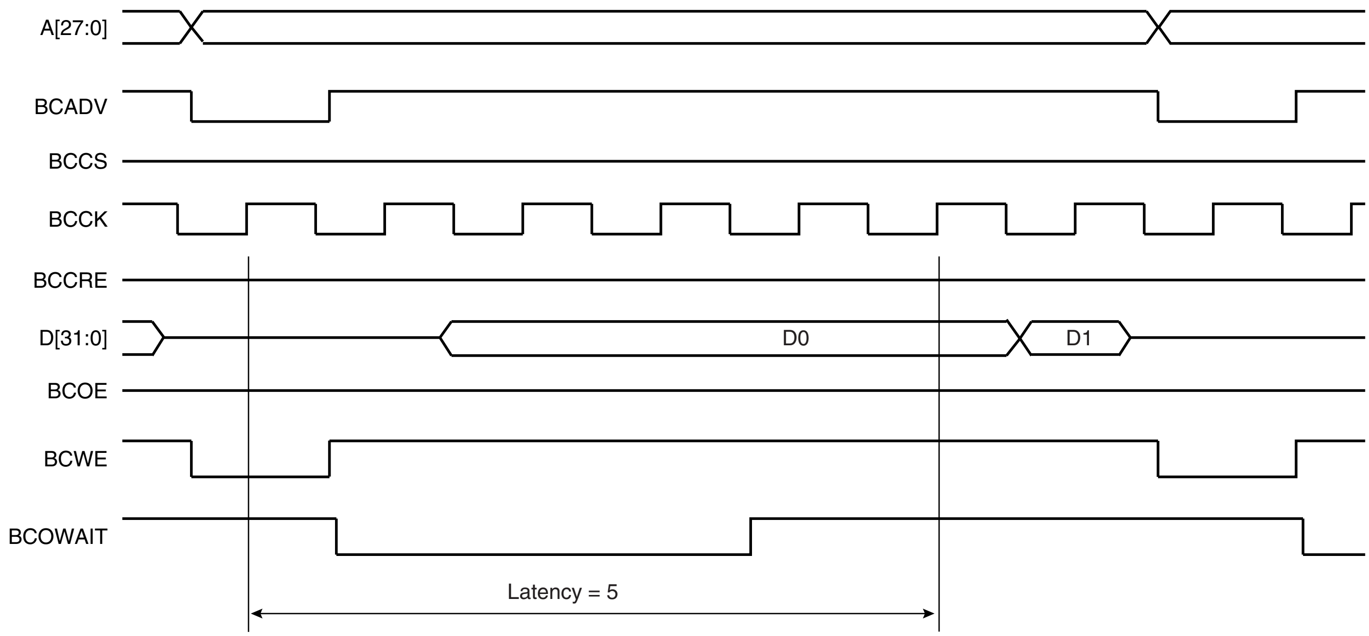


Figure 25-5. Burst Write Access with No Refresh Collision

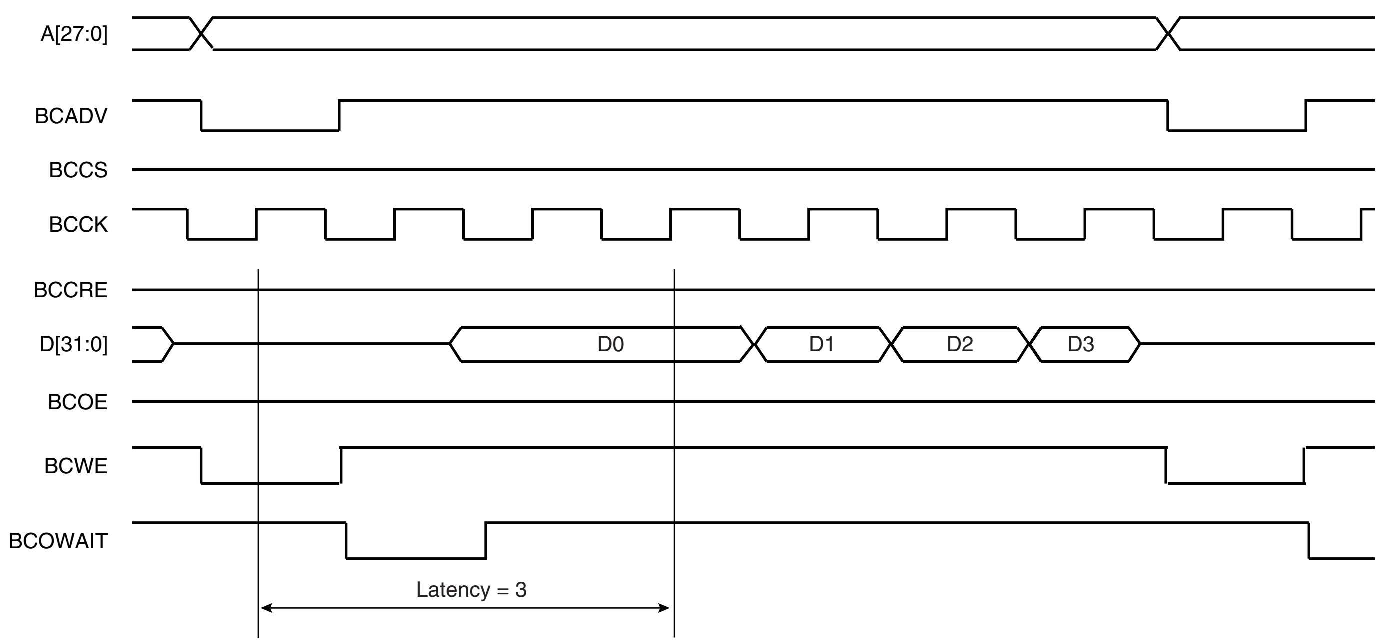


Figure 25-6. Four Beat Wrapping Burst With Address Starting at 0x0C

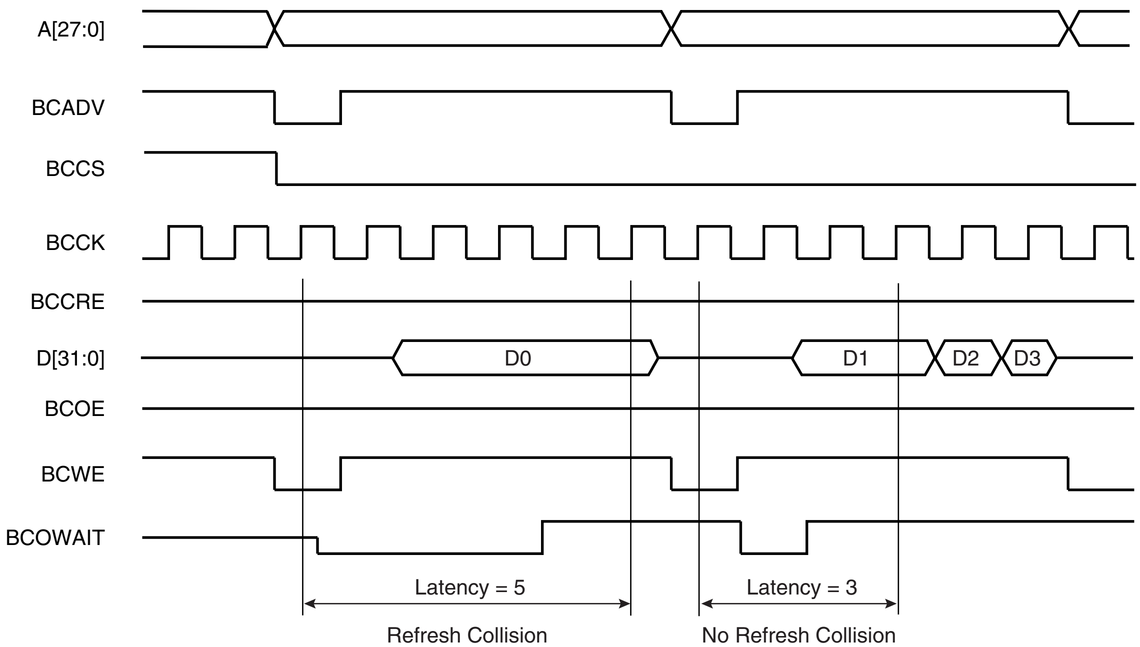
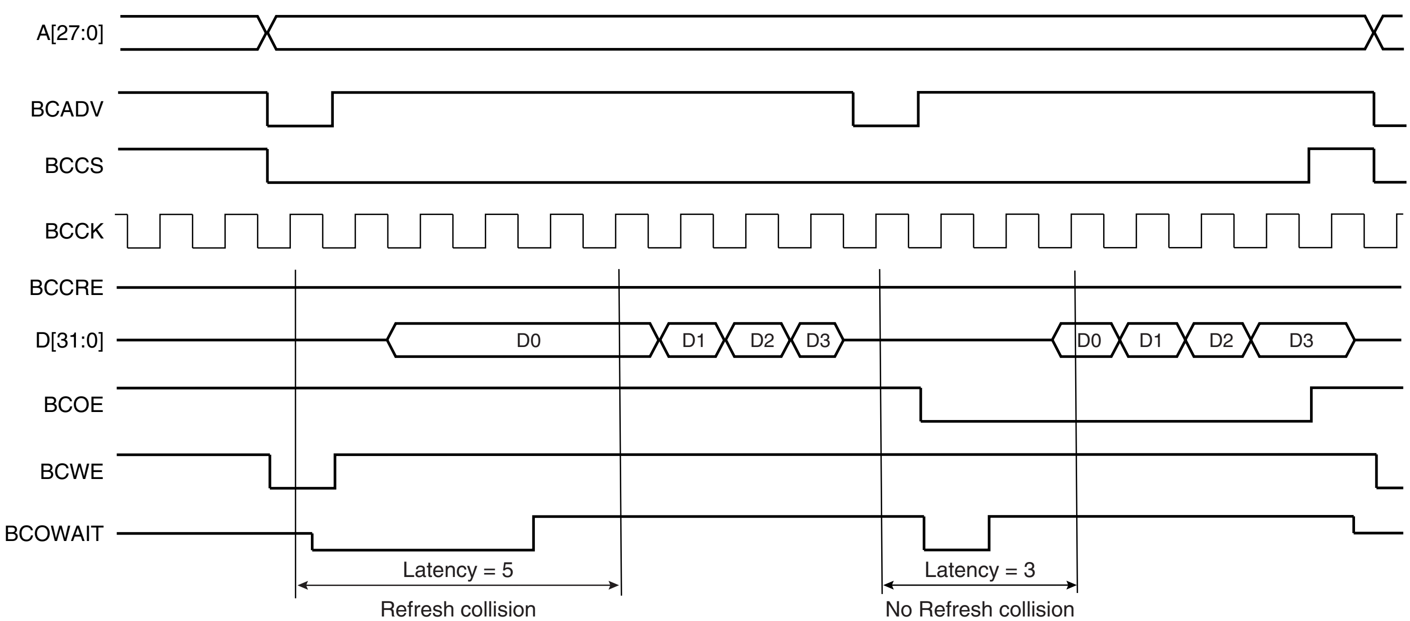


Figure 25-7. Write Command Followed by a Read Command then Interrupt Write Burst



25.4.3 BCRAMC Read Cycle

The BCRAMC allows burst access or single access.

The Cellular RAM device is programmed with a continuous burst length.

The latency from read command to data output is dependant on the Cellular RAM version. The owait signal is monitored to detect any conflict of refresh collision.

To initiate a single access, the BCRAMC generates an initial burst read command. To comply with Cellular RAM timing parameters, additional clock cycles are inserted to check programmed latency. The BCRAMC supports latency value which is a function of the Cellular RAM version. The owait signal is monitored to detect a refresh collision. As soon as owait signal is high, data is transferred out of the device and first data can be read. As the burst length is fixed to continuous, in the case of single access, it has to stop the burst else invalid values could be read. To interrupt the read operation, BCCS (chip select) must be set to 1 or an initial burst read/write command can be initiated to interrupt current access if the next access is a Cellular RAM access.

To initiate a burst access, the BCRAMC uses the transfer type signals provided by the master requesting the access. If the next access is a sequential read access, reading to the Cellular RAM device is carried out. If the next access is a read sequential access, but the current access is to a boundary page, then an automatic page break is inserted and the Cellular RAM controller generates an initial burst read command to finish the access. To comply with Cellular RAM timing parameters, additional clock cycles are inserted to check programmed latency. Like a single access, the owait signal is monitored to detect refresh collision.

The BCRAMC can anticipate 1 or 2 read accesses. In this case tdf_fr_cram is generated to alert the EBI that data is floating on the bus and that the next external access will be delayed. In the case of a burst of specified length, accesses are not anticipated, but, if the burst is broken (i.e., border, busy mode...)the next access will be treated as an incrementing burst of unspecified length, and the BCRAMC can anticipate 1 or 2 read accesses. In this case tdf_from_cram is generated to alert the EBI that data is floating on the bus and that the next external access will be delayed.

Read accesses to the Cellular RAM are burst oriented, the burst length programmed is continuous burst. This feature makes it possible to start at a specified address and burst through the entire memory. It is very useful for incrementing burst (INCR/INCR4/INCR8/INCR16), as soon as the burst command init (latch burst start address) is initiated and latency is checked, at each BCCK rising a data is read.

In the case of wrapping burst (WRAP4/WRAP8/WRAP16), the addresses can cross the boundary of the current transfer. For example, when a transfer (WRAP4) starts at address 0x0C, the next access will be 0x00, but for the burst length being programmed to continuous burst, the next access should be 0x10. The burst does not wrap automatically. The BCRAMC takes this feature into account and in the case of a transfer starting at address 0x04/0x08/0x0C, two initial burst read commands will be issued to wrap when the boundary is reached. The last initial burst read command will be interrupted by BCCS set to high or by another initial burst read/write command to do the next access in the Cellular RAM device if an access is pending. tdf_fr_cram will be generated to alert the EBI that data is floating on the bus.

Figure 25-8. Single Read Access with Refresh Collision

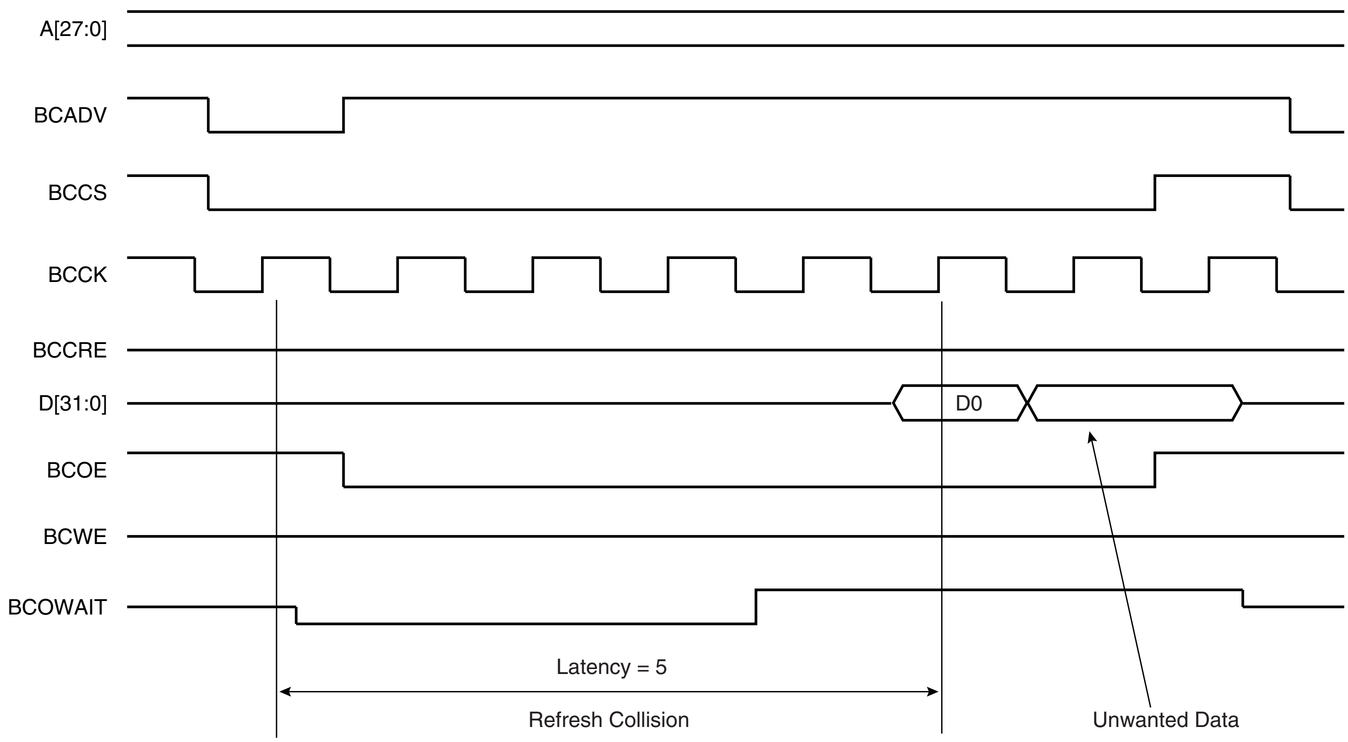


Figure 25-9. Single Read Access with No Refresh Collision

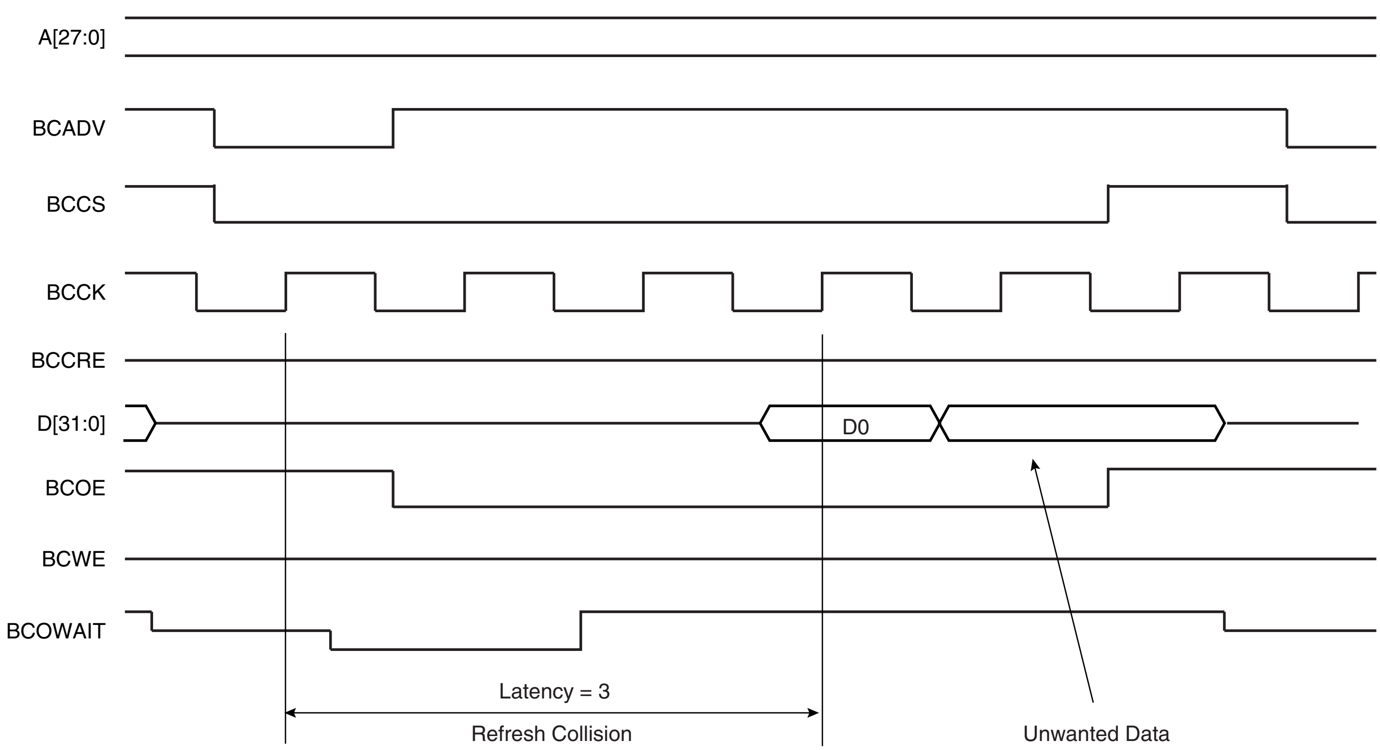


Figure 25-10. Burst Read Access with No Refresh Collision

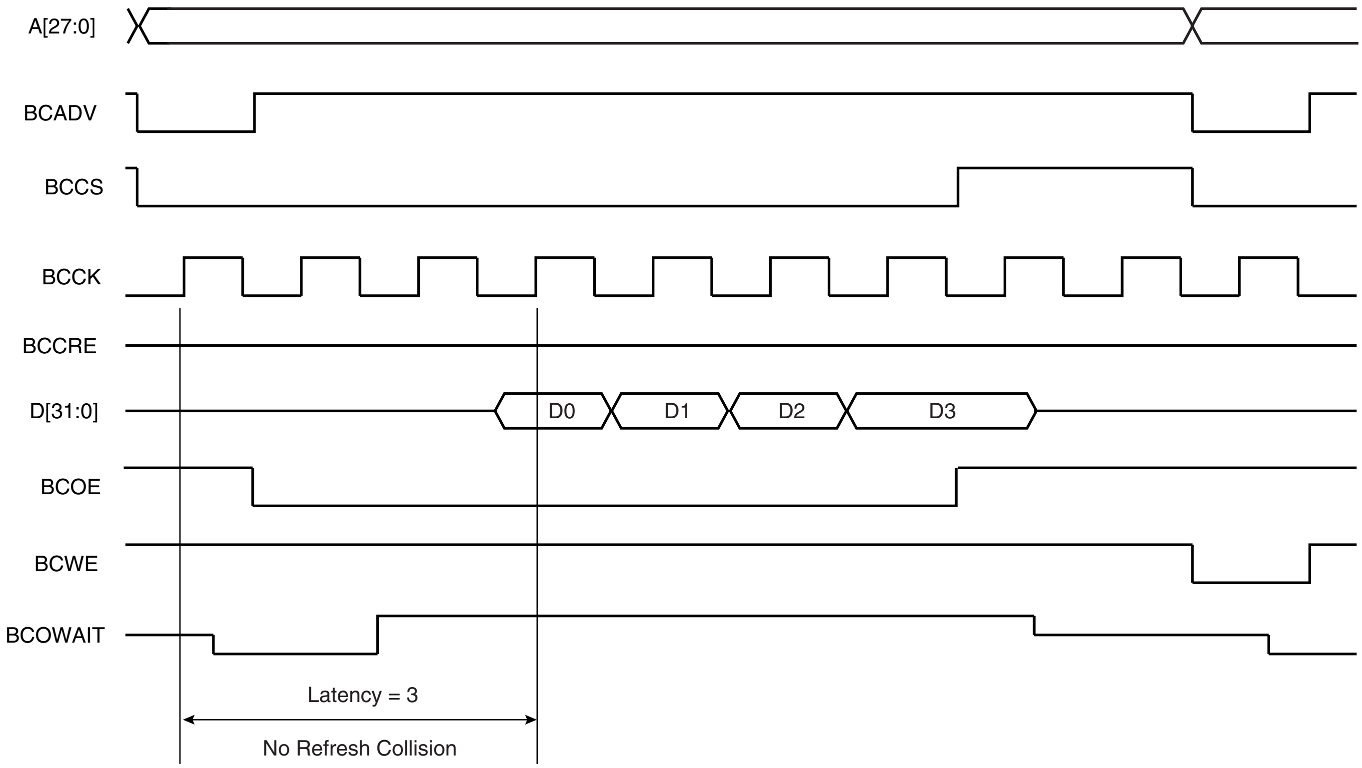
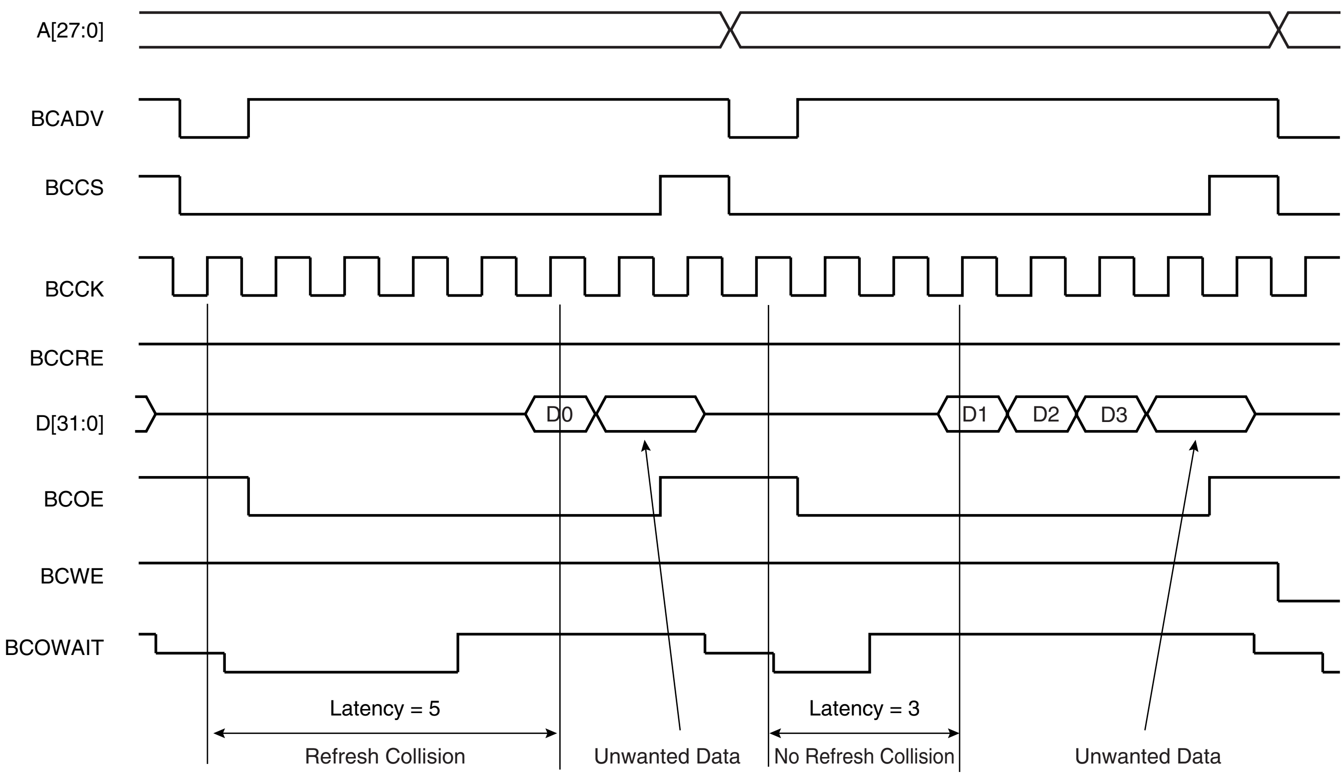


Figure 25-11. Four Beat Wrapping Burst with Address Starting at 0x0C



25.4.4 Power Management

25.4.4.1 Standby Mode

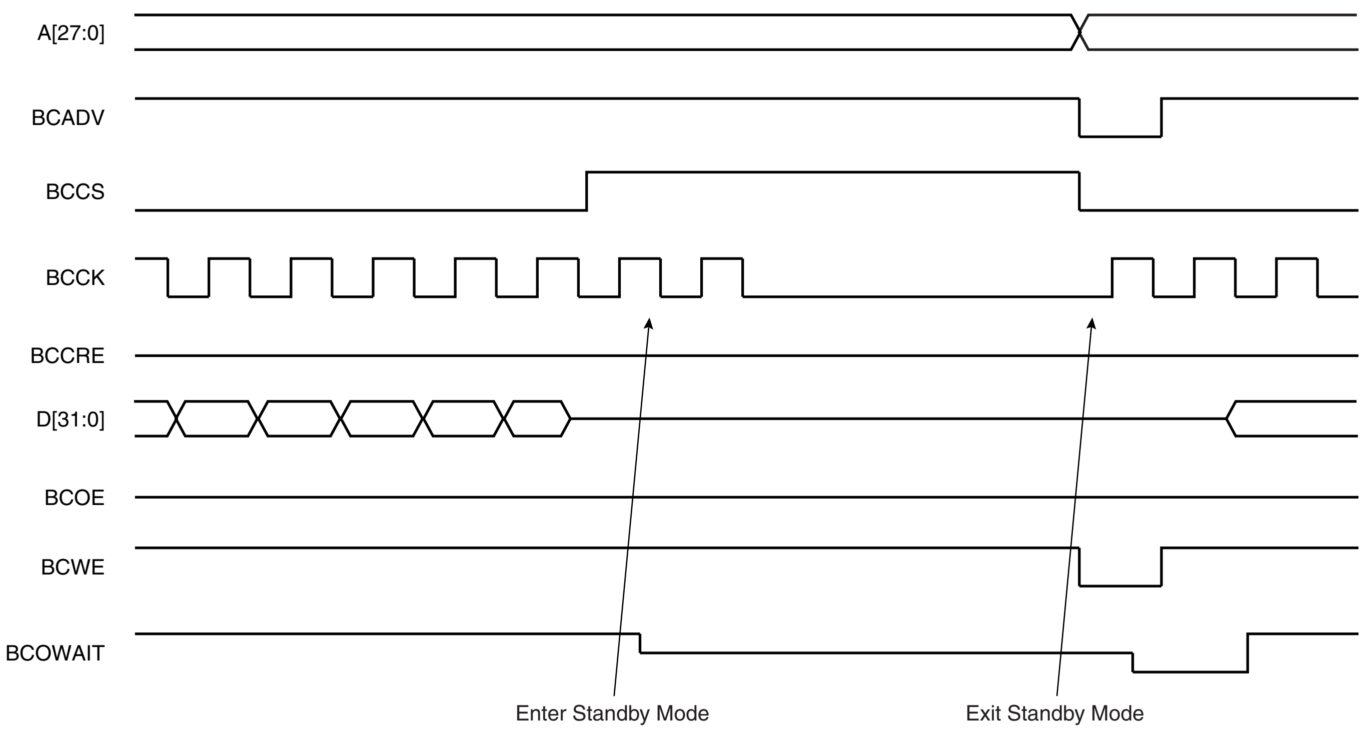
This mode is activated by programming low power command bits (LPCB) to 1 (See "LPCB: Low Power Command Bit" on page 273.)

Standby mode is used when there is no access to the Cellular RAM device. In this mode, power consumption is reduced, BCCK pin is low and ce pin is high.

When the device is in standby mode, address inputs and data inputs/outputs are internally isolated from external influence.

The BCRAMC leaves standby mode as soon as the Cellular RAM device is selected and returns to standby mode as soon as the Cellular RAM device is no longer selected.

Figure 25-12. Standby Mode Entry/Exit



25.4.4.2 Deep Power Down Mode (DPD)

This mode is selected by programming the LPCB field to 2 in the BCRAMC Low Power Register (See "LPCB: Low Power Command Bit" on page 273.). As soon as LPCB field is programmed and no access in Cellular RAM is pending, the Refresh Configuration Register is automatically accessed and DPD mode is enabled.

When this mode is activated, all internal voltage generators inside the Cellular RAM are stopped and all data is lost. Only the register values of the Bus Configuration Register and Refresh Configuration Register are kept valid during deep power down mode.

To leave this mode, the LPCB field must be programmed to 0 in the BCRAMC Low Power Register. As soon as the LPCB field is programmed and Cellular RAM access is pending, the Refresh Configuration Register is automatically accessed and the DPD bit is disabled.

No command should be applied during 150 µs, before re-entry in idle or standby mode.

25.4.4.3 Temperature Compensated Refresh (TCR) or Temperature Compensated Self-refresh (TCSR)

This feature is activated by adjusting Temperature Compensated Refresh bits (TCR) in BCRAMC Low Power Register (See "TCR_TCSR: Temperature Compensated Refresh or Temperature Compensated Self-refresh" on page 273.).

This feature allows to adjust the refresh period in function of different temperatures.

Some Cellular RAM devices include an on chip temperature sensor that automatically adjusts the refresh period according to operating temperature, in this case the value of TCR is set to 0.

On the contrary, some Cellular RAM devices do not include an internal sensor. In this case, TCR values can be changed by modifying the TCR bits in the BCRAMC Low Power Register the Refresh Configuration Register is automatically accessed and TCR value is adjusted before doing the next access in the Cellular RAM device.

25.4.4.4 Partial Array Refresh (PAR)

This feature is activated by adjusting Partial Array Refresh bits (PAR) in the BCRAMC Low Power Register (See "PAR: Partial Array Refresh" on page 273.).

PAR can restrict the refresh operation to a portion of the total memory area. Data stored in addresses not receiving refresh will become corrupted.

As soon as TCR field is modified, the Refresh Configuration Register and theTCR bit are automatically updated before doing the next access in the Cellular RAM device.

25.5 Burst Cellular RAM Controller (BCRAMC) User Interface

The User interface is connected to the APB bus. The BCRAMC is programmed using the registers listed in Table 25-1.

Table 25-1. BCRAMC Memory Map

Offset	Register	Name	Access	Reset
0x00	BCRAMC Configuration Register	BCRAMC_CR	Read-write	0x00000130
0x04	BCRAMC Timing Register	BCRAMC_TR	Read-write	0x004
0x0C	BCRAMC Low Power Register	BCRAMC_LPR	Read-write	0x0
0x10	BCRAMC Memory Device Register	BCRAMC_MDR	Read-write	0x0
0x14 - 0xE8	Reserved			
0xEC	BCRAMC Address Size Register	BCRAMC_ADDRSIZE	Read-only	0x-(2)
0xF0	BCRAMC IP Name 1	BCRAMC_IPNAME1	Read-only	“HBCR”
0xF4	BCRAMC IP Name 2	BCRAMC_IPNAME2	Read-only	“AMC1“
0xF8	BCRAMC Features Registers	BCRAMC_FEATURES	Read-only	0x

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

2. Values in the BCRAMC_ADDRSIZE register are product dependent. For more information, see “BCRAMC ADDRSIZE Register” on page 275.

25.5.1 BCRAMC Configuration Register

Name: BCRAMC_CR

Address: 0xFFFFE400

Access: Read-write

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

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

15	14	13	12	11	10	9	8
–	–	BOUNDARY_WORD		–	–	–	DBW

7	6	5	4	3	2	1	0
–	LM			–	–	–	CRAM_EN

- **CRAM_EN: BCRAMC Enabled**

The Reset Value is 0.

This field enables or disables the BCRAMC. As soon as Cellular RAM is enabled, power up sequence can be done. When cram_en bit is low then BCRAMC is in idle mode and the owait signal is masked. When this bit is disabled during functional mode, an initialization procedure will be performed to again access the Cellular RAM device.

- **LM: Latency Mode**

The Reset Value is 3 Cycles.

LM	Latency Cycles	Cellular RAM Version
000	Reserved	Reserved
001	Reserved	Reserved
010	2	1.0/1.5/2.0
011	3	1.0/1.5/2.0
100	4	1.5/2.0
101	5	1.5/2.0
110	6	1.5/2.0
111	Reserved	Reserved

- **DBW: Data Bus Width**

0: Data bus width is 32 bits.

1: Data bus width is 16 bits.

- **BOUNDARY_WORD: Number of Words in Row**

The Reset Value is 64 words (word = 16 or 32 bits) in row.

This field manages the row boundaries. Some Cellular RAM providers do not provide the number of words in row in their devices, in this case the reset value is used.

Boundary Word	Number of Words in Row
00	64
01	128
10	256
11	512

- **ADDRDATA_MUX**

Reset Value is 0.

This field is used to multiplex the address and data bus. This feature is reserved for Cellular RAM version 2.0. In the case of Cellular RAM version 1.0, 1.5 the value is 0.

ADDRDTAT_MUX	
0	address data bus not multiplexed
1	address data bus multiplexed

- **DS: Drive Strength**

Reset Value is 0.

This field is used to select the driver strength of Cellular RAM output.

The table below gives an example and can change with the Cellular RAM provider.

DS	Drive Strength
00	full
01	1/2
10	1/4
11	reserved

- **VAR_FIX_LAT: Variable Latency or Fixed Latency**

The Reset Value is 0, variable latency.

VAR_FIX_LAT	
0	variable latency
1	fixed latency

This feature is reserved for Cellular RAM version 1.5/2.0. In the case of Cellular RAM version 1.0 the value is 0.

In the variable latency mode, the latency programmed in the bus configuration register is not guaranteed, it is maintained only if there is no refresh collision. The wait signal must be monitored.

In the fixed latency mode, the first data outputs conform to the fixed timing, including refresh collision. The wait signal can be unmonitored. This mode is of benefit for applications with low clock frequency.

25.5.2 BCRAMC Timing Register

Name: BCRAMC_TR

Address: 0xFFFFE404

Access: 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
–	–	–	–	TCKA			

7	6	5	4	3	2	1	0
–	–	TCRES		TCW			

- **TCW: Chip Enable to End of Write.**

Reset value is 4.

This field defines the time between the falling edge of BCCS and the rising edge of BCWE in number of cycles. The Number of cycles is equal to TCW + 4. This time is used during initialization sequence since accesses to configuration registers are done in asynchronous mode.

- **TCRES: Control Register Enable Setup**

Reset value is 0.

This field defines the time between the rising edge of cre and the falling edge of BCWE in number of cycles. The Number of cycles is equal to TCRES + 1. This time is used during initialization sequence since accesses to configuration registers are done in asynchronous mode.

- **TCKA: BCWE High to BCCK Valid**

Reset value is 0.

This field defines the time between the BCWE rising edge and BCCK switch on, in number of cycles. The number of cycles is equal to TCKA + 2.5.

25.5.3 BCRAMC Low Power Register

Name: BCRAMC_LPR

Address: 0xFFFFE40C

Access: 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
–	–	–	–	–	–	LPCB	

7	6	5	4	3	2	1	0
–	–	TCR_TCSR		–	PAR		

- **PAR: Partial Array Refresh**

PAR restricts the refresh operation to a portion of the total memory array.

The table below gives an example of PAR but can change with the Cellular RAM provider.

PAR	Refresh Coverage
000	Full array
001	Bottom 1/2 array
010	Bottom 1/4 array
011	Bottom 1/8 array
100	None of 2 array
101	Top 1/2 array
110	Top 1/2 array
111	Top 1/2 array

- **TCR_TCSR: Temperature Compensated Refresh or Temperature Compensated Self-refresh**

TCR or TCSR refresh the device in function of difference in temperature.

The table below gives an example of TCR or TCSR but can change with the Cellular RAM provider.

TCR or TCSR	Temperature
11	+85° C
00	Internal sensor or +70° C
01	+45° C
10	+15° C

- **LPCB: Low Power Command Bit**

00: Low Power Feature is inhibited: Standby and Deep Power mode are not issued to the Cellular RAM device.

01: The Cellular RAM device Standby mode is enabled.

10: Deep Power Down mode is enabled.

11: reserved

25.5.4 BCRAMC Memory Device Register

Name: BCRAMC_MDR

Address: 0xFFFFE410

Access: 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**

Gives the type of memory used.

00: Cellular RAM Version 1.0

01: Cellular RAM Version 1.5

10: Cellular RAM Version 2.0

11: reserved

25.5.5 BCRAMC ADDRSIZE Register

Name: BCRAMC_ADDRSIZE

Address: 0xFFFFE4EC

Access: Read-only

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

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

15	14	13	12	11	10	9	8
ADDRSIZE							

7	6	5	4	3	2	1	0
ADDRSIZE							

- **ADDRSIZE**

Reserved. Value subject to change. No functionality associated. The returned value corresponds to the number of bytes mapped into the BCRAMC address space. It could be an integer in range of 16384, 8192, 4096, 2048, 1024, 512, 256.

25.5.6 BCRAMC Name1 Register

Name: BCRAMC_IPNAME1

Access: Read-only

31	30	29	28	27	26	25	24
IPNAME							

23	22	21	20	19	18	17	16
IPNAME							

15	14	13	12	11	10	9	8
IPNAME							

7	6	5	4	3	2	1	0
IPNAME							

- **IPNAME**

Reserved. Value subject to change. No functionality associated. The name in ASCII format is “HBCR”.

25.5.7 BCRAMC Name2 Register

Name: BCRAMC_IPNAME2

Access: Read-only

31	30	29	28	27	26	25	24
IPNAME							

23	22	21	20	19	18	17	16
IPNAME							

15	14	13	12	11	10	9	8
IPNAME							

7	6	5	4	3	2	1	0
IPNAME							

- **IPNAME**

Reserved. Value subject to change. No functionality associated. The name in ASCII format is “AMC1”.

25.5.8 BCRAMC Features Register

Name: BCRAMC_FEATURES

Address: 0xFFFFE4F8

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

Reserved

26. Error Corrected Code (ECC) Controller

26.1 Description

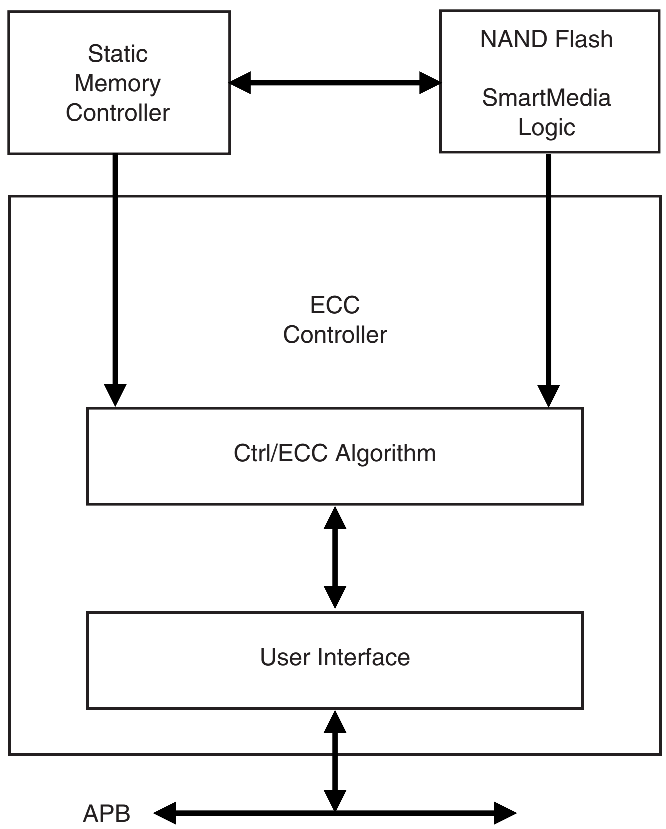
NAND Flash/SmartMedia devices contain by default invalid blocks which have one or more invalid bits. Over the NAND Flash/SmartMedia lifetime, additional invalid blocks may occur which can be detected/corrected by ECC code.

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).

26.2 Block Diagram

Figure 26-1. Block Diagram



26.3 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.

Over time, some memory locations may fail to program or erase properly. In order to ensure that data is stored properly over the life of the NAND Flash device, NAND Flash providers recommend to utilize either 1 ECC per 256 bytes of data, 1 ECC per 512 bytes of data or 1 ECC for all of the page.

The only configurations required for ECC are the NAND Flash or the SmartMedia page size (528/2112/4224) and the type of correction wanted (1 ECC for all the page/1 ECC per 256 bytes of data /1 ECC per 512 bytes of data). Page size is configured setting the PAGESIZE field in the ECC Mode Register (ECC_MR). Type of correction is configured setting the TYPECORRECT 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 Registers (ECC_PR0 to ECC_PR15) are then valid and locked until a new start condition occurs (read/write command followed by address cycles).

26.3.1 Write Access

Once the Flash memory page is written, the computed ECC codes are available in the ECC Parity (ECC_PR0 to ECC_PR15) registers. The ECC code values must be written by the software application in the extra area used for redundancy. The number of write accesses in the extra area is a function of the value of the type of correction field. For example, for 1 ECC per 256 bytes of data for a page of 512 bytes, only the values of ECC_PR0 and ECC_PR1 must be written by the software application. Other registers are meaningless.

26.3.2 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.

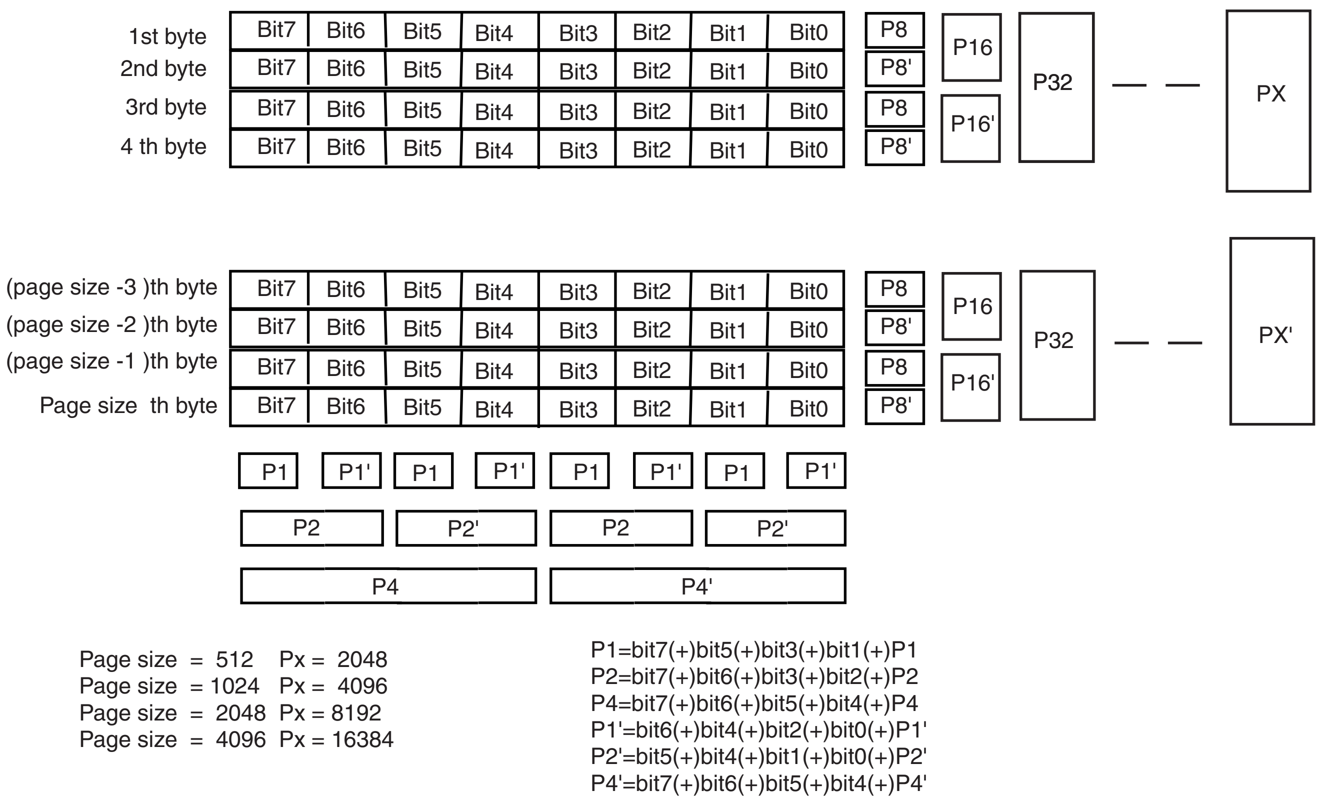
The application can check the ECC Status Registers (ECC_SR1/ECC_SR2) for any detected errors. It is up to the application to correct any detected error. ECC computation can detect four different circumstances:

- 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 Registers (ECC_SR1/ECC_SR2).
- Recoverable error: Only the RECERR flags in the ECC Status registers (ECC_SR1/ECC_SR2) are set. The corrupted word offset in the read page is defined by the WORDADDR field in the ECC Parity Registers (ECC_PR0 to ECC_PR15). The corrupted bit position in the concerned word is defined in the BITADDR field in the ECC Parity Registers (ECC_PR0 to ECC_PR15).
- ECC error: The ECCERR flag in the ECC Status Registers (ECC_SR1/ECC_SR2) are 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 Registers (ECC_SR1/ECC_SR2) are set. Several unrecoverable errors have been detected in the Flash memory page.

ECC Status Registers, ECC Parity Registers 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. 24-bit ECC is generated in order to perform one bit correction per 256 or 512 bytes for pages of 512/2048/4096 8-bit words. 32-bit ECC is generated in order to perform one bit correction per 512/1024/2048/4096 8- or 16-bit words.They are generated according to the schemes shown in Figure 26-2 and Figure 26-3.

Figure 26-2. Parity Generation for 512/1024/2048/4096 8-bit Words

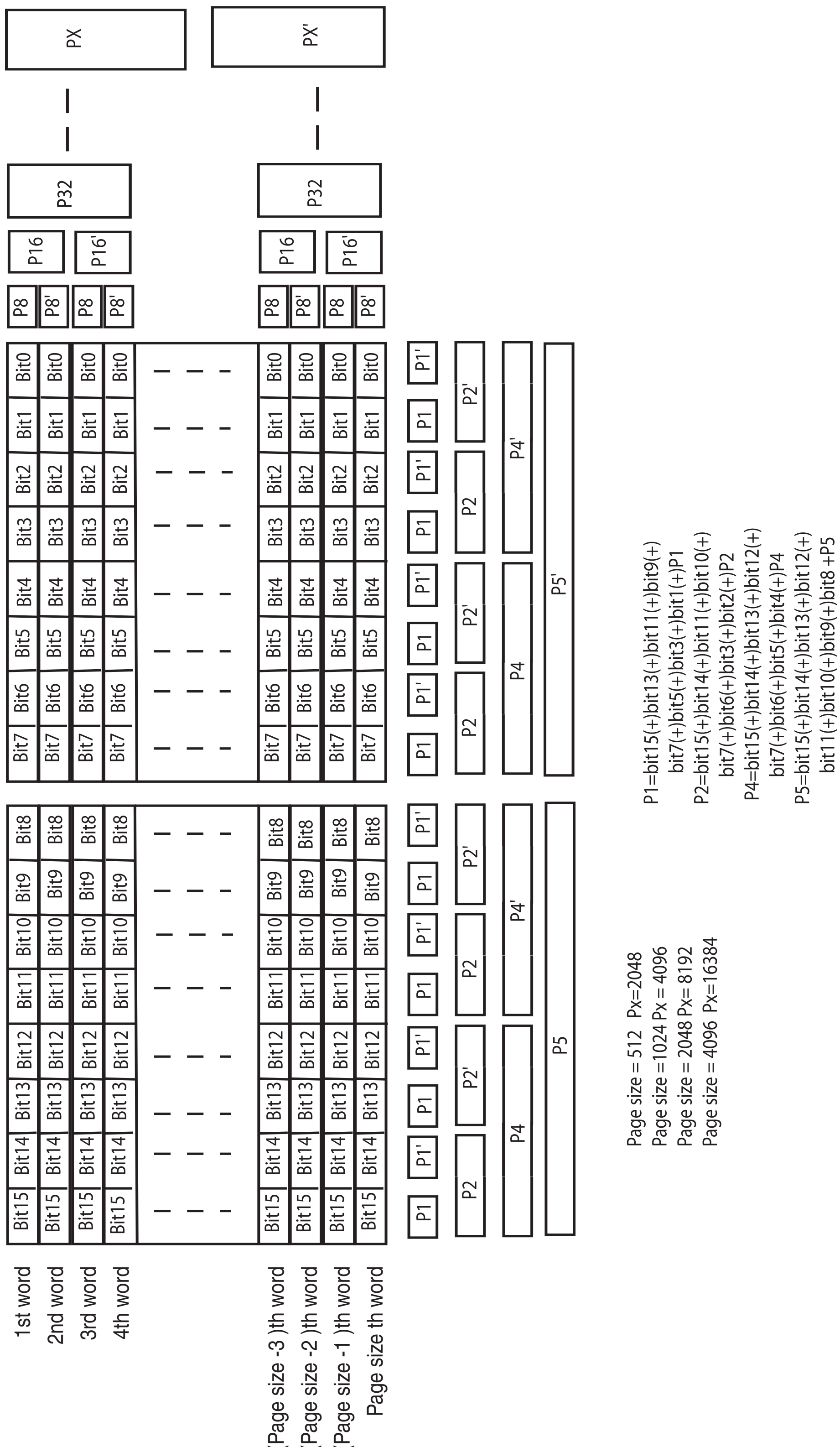


To calculate P8' to PX' and P8 to PX, apply the algorithm that follows.

```
Page size = 2^n

   for i =0 to n
   begin
    for (j = 0 to page_size_byte)
    begin
     if(j[i] ==1)
      P[2^(i+3)]=bit7(+)bit6(+)bit5(+)bit4(+)bit3(+)
                 bit2(+)bit1(+)bit0(+)P[2^(i+3)]
     else
      P[2^(i+3)]'=bit7(+)bit6(+)bit5(+)bit4(+)bit3(+)
                  bit2(+)bit1(+)bit0(+)P[2^(i+3)]'
     end
   end
```

Figure 26-3. Parity Generation for 512/1024/2048/4096 16-bit Words



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[2^(i+3)]= bit15(+)bit14(+)bit13(+)bit12(+)
                 bit11(+)bit10(+)bit9(+)bit8(+)
                 bit7(+)bit6(+)bit5(+)bit4(+)bit3(+)
                 bit2(+)bit1(+)bit0(+)P[2^(n+3)]
     else
      P[2^(i+3)]'=bit15(+)bit14(+)bit13(+)bit12(+)
                 bit11(+)bit10(+)bit9(+)bit8(+)
                 bit7(+)bit6(+)bit5(+)bit4(+)bit3(+)
                 bit2(+)bit1(+)bit0(+)P[2^(i+3)]'
     end
   end
```

26.4 Error Corrected Code Controller (ECC) User Interface

Table 26-1. Register Mapping

Offset	Register	Name	Access	Reset
0x00	ECC Control Register	ECC_CR	Write-only	0x0
0x04	ECC Mode Register	ECC_MR	Read-write	0x0
0x08	ECC Status1 Register	ECC_SR1	Read-only	0x0
0x0C	ECC Parity Register 0	ECC_PR0	Read-only	0x0
0x10	ECC Parity Register 1	ECC_PR1	Read-only	0x0
0x14	ECC Status2 Register	ECC_SR2	Read-only	0x0
0x18	ECC Parity 2	ECC_PR2	Read-only	0x0
0x1C	ECC Parity 3	ECC_PR3	Read-only	0x0
0x20	ECC Parity 4	ECC_PR4	Read-only	0x0
0x24	ECC Parity 5	ECC_PR5	Read-only	0x0
0x28	ECC Parity 6	ECC_PR6	Read-only	0x0
0x2C	ECC Parity 7	ECC_PR7	Read-only	0x0
0x30	ECC Parity 8	ECC_PR8	Read-only	0x0
0x34	ECC Parity 9	ECC_PR9	Read-only	0x0
0x38	ECC Parity 10	ECC_PR10	Read-only	0x0
0x3C	ECC Parity 11	ECC_PR11	Read-only	0x0
0x40	ECC Parity 12	ECC_PR12	Read-only	0x0
0x44	ECC Parity 13	ECC_PR13	Read-only	0x0
0x48	ECC Parity 14	ECC_PR14	Read-only	0x0
0x4C	ECC Parity 15	ECC_PR15	Read-only	0x0

26.4.1 ECC Control Register

Name: ECC_CR

Address: 0xFFFFE200

Access: Write-only

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

- **RST: RESET Parity**

Provides reset to current ECC by software.

1: Reset ECC Parity registers

0: No effect

- **SRST: Soft Reset**

Provides soft reset to ECC block

1: Resets all registers.

0: No effect.

26.4.2 ECC Mode Register

Name: ECC_MR

Address: 0xFFFFE204

Access: 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**
–	–	TYPECORRECT		–	–	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.

- **TYPECORRECT: Type of Correction**

00: 1 bit correction for a page size of 512/1024/2048/4096 bytes.

01: 1 bit correction for 256 bytes of data for a page size of 512/2048/4096 bytes.

10: 1 bit correction for 512 bytes of data for a page size of 512/2048/4096 bytes.

26.4.3 ECC Status Register 1

Name: ECC_SR1

Address: 0xFFFFE208

Access: Read-only

31	30	29	28	27	26	25	24
–	MULERR7	ECCERR7	RECERR7	–	MULERR6	ECCERR6	RECERR6
23	22	21	20	19	18	17	16
–	MULERR5	ECCERR5	RECERR5	–	MULERR4	ECCERR4	RECERR4
15	14	13	12	11	10	9	8
–	MULERR3	ECCERR3	RECERR3	–	MULERR2	ECCERR2	RECERR2
7	6	5	4	3	2	1	0
–	MULERR1	ECCERR1	RECERR1	–	MULERR0	ECCERR0	RECERR0

- **RECERR0: 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.

- **ECCERR0: ECC Error**

0 = No Errors Detected.

1 = A single bit error occurred in the ECC bytes.

If TYPECORRECT = 0, read both ECC Parity 0 and ECC Parity 1 registers, the error occurred at the location which contains a 1 in the least significant 16 bits; else read ECC Parity 0 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

- **MULERR0: Multiple Error**

0 = No Multiple Errors Detected.

1 = Multiple Errors Detected.

- **RECERR1: Recoverable Error in the page between the 256th and the 511th bytes or the 512th and the 1023rd bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected.

- **ECCERR1: ECC Error in the page between the 256th and the 511th bytes or the 512th and the 1023rd bytes**

Fixed to 0 if TYPECORREC = 0

0 = No Errors Detected.

1 = A single bit error occurred in the ECC bytes.

Read ECC Parity 1 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

- **MULERR1: Multiple Error in the page between the 256th and the 511th bytes or the 512th and the 1023rd bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Multiple Errors Detected.

1 = Multiple Errors Detected.

- **RECERR2: Recoverable Error in the page between the 512th and the 767th bytes or the 1024th and the 1535th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise, multiple uncorrected errors were detected.

- **ECCERR2: ECC Error in the page between the 512th and the 767th bytes or the 1024th and the 1535th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = A single bit error occurred in the ECC bytes.

Read ECC Parity 2 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

- **MULERR2: Multiple Error in the page between the 512th and the 767th bytes or the 1024th and the 1535th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Multiple Errors Detected.

1 = Multiple Errors Detected.

- **RECERR3: Recoverable Error in the page between the 768th and the 1023rd bytes or the 1536th and the 2047th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected.

- **ECCERR3: ECC Error in the page between the 768th and the 1023rd bytes or the 1536th and the 2047th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = A single bit error occurred in the ECC bytes.

Read ECC Parity 3 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

- **MULERR3: Multiple Error in the page between the 768th and the 1023rd bytes or the 1536th and the 2047th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Multiple Errors Detected.

1 = Multiple Errors Detected.

- **RECERR4: Recoverable Error in the page between the 1024th and the 1279th bytes or the 2048th and the 2559th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected.

- **ECCERR4: ECC Error in the page between the 1024th and the 1279th bytes or the 2048th and the 2559th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = A single bit error occurred in the ECC bytes.

Read ECC Parity 4 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

- **MULERR4: Multiple Error in the page between the 1024th and the 1279th bytes or the 2048th and the 2559th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Multiple Errors Detected.

1 = Multiple Errors Detected.

- **RECERR5: Recoverable Error in the page between the 1280th and the 1535th bytes or the 2560th and the 3071st bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected

- **ECCERR5: ECC Error in the page between the 1280th and the 1535th bytes or the 2560th and the 3071st bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = A single bit error occurred in the ECC bytes.

Read ECC Parity 5 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

- **MULERR5: Multiple Error in the page between the 1280th and the 1535th bytes or the 2560th and the 3071st bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Multiple Errors Detected.

1 = Multiple Errors Detected.

- **RECERR6: Recoverable Error in the page between the 1536th and the 1791st bytes or the 3072nd and the 3583rd bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected.

- **ECCERR6: ECC Error in the page between the 1536th and the 1791st bytes or the 3072nd and the 3583rd bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = A single bit error occurred in the ECC bytes.

Read ECC Parity 6 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

- **MULERR6: Multiple Error in the page between the 1536th and the 1791st bytes or the 3072nd and the 3583rd bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Multiple Errors Detected.

1 = Multiple Errors Detected.

- **RECERR7: Recoverable Error in the page between the 1792nd and the 2047th bytes or the 3584th and the 4095th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise, multiple uncorrected errors were detected.

- **ECCERR7: ECC Error in the page between the 1792nd and the 2047th bytes or the 3584th and the 4095th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = A single bit error occurred in the ECC bytes.

Read ECC Parity 7 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

- **MULERR7: Multiple Error in the page between the 1792nd and the 2047th bytes or the 3584th and the 4095th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Multiple Errors Detected.

1 = Multiple Errors Detected.

26.4.4 ECC Status Register 2

Name: ECC_SR2

Address: 0xFFFFE214

Access: Read-only

31	30	29	28	27	26	25	24
–	MULERR15	ECCERR15	RECERR15	–	MULERR14	ECCERR14	RECERR14
23	22	21	20	19	18	17	16
–	MULERR13	ECCERR13	RECERR13	–	MULERR12	ECCERR12	RECERR12
15	14	13	12	11	10	9	8
–	MULERR11	ECCERR11	RECERR11	–	MULERR10	ECCERR10	RECERR10
7	6	5	4	3	2	1	0
–	MULERR9	ECCERR9	RECERR9	–	MULERR8	ECCERR8	RECERR8

- **RECERR8: Recoverable Error in the page between the 2048th and the 2303rd bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected

- **ECCERR8: ECC Error in the page between the 2048th and the 2303rd bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = A single bit error occurred in the ECC bytes.

Read ECC Parity 8 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

- **MULERR8: Multiple Error in the page between the 2048th and the 2303rd bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Multiple Errors Detected.

1 = Multiple Errors Detected.

- **RECERR9: Recoverable Error in the page between the 2304th and the 2559th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected.

- **ECCERR9: ECC Error in the page between the 2304th and the 2559th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = A single bit error occurred in the ECC bytes.

Read ECC Parity 9 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

- **MULERR9: Multiple Error in the page between the 2304th and the 2559th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Multiple Errors Detected.

1 = Multiple Errors Detected.

- **RECERR10: Recoverable Error in the page between the 2560th and the 2815th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise, multiple uncorrected errors were detected.

- **ECCERR10: ECC Error in the page between the 2560th and the 2815th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = A single bit error occurred in the ECC bytes.

Read ECC Parity 10 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

- **MULERR10: Multiple Error in the page between the 2560th and the 2815th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Multiple Errors Detected.

1 = Multiple Errors Detected.

- **RECERR11: Recoverable Error in the page between the 2816th and the 3071st bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise, multiple uncorrected errors were detected

- **ECCERR11: ECC Error in the page between the 2816th and the 3071st bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = A single bit error occurred in the ECC bytes.

Read ECC Parity 11 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

- **MULERR11: Multiple Error in the page between the 2816th and the 3071st bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Multiple Errors Detected.

1 = Multiple Errors Detected.

- **RECERR12: Recoverable Error in the page between the 3072nd and the 3327th bytes**

Fixed to 0 if TYPECORREC = 0

0 = No Errors Detected

1 = Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected

- **ECCERR12: ECC Error in the page between the 3072nd and the 3327th bytes**

Fixed to 0 if TYPECORREC = 0

0 = No Errors Detected

1 = A single bit error occurred in the ECC bytes.

Read ECC Parity 12 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

- **MULERR12: Multiple Error in the page between the 3072nd and the 3327th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Multiple Errors Detected.

1 = Multiple Errors Detected.

- **RECERR13: Recoverable Error in the page between the 3328th and the 3583rd bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise multiple uncorrected errors were detected.

- **ECCERR13: ECC Error in the page between the 3328th and the 3583rd bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = A single bit error occurred in the ECC bytes.

Read ECC Parity 13 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

- **MULERR13: Multiple Error in the page between the 3328th and the 3583rd bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Multiple Errors Detected.

1 = Multiple Errors Detected.

- **RECERR14: Recoverable Error in the page between the 3584th and the 3839th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise, multiple uncorrected errors were detected.

- **ECCERR14: ECC Error in the page between the 3584th and the 3839th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = A single bit error occurred in the ECC bytes.

Read ECC Parity 14 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

- **MULERR14: Multiple Error in the page between the 3584th and the 3839th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Multiple Errors Detected.

1 = Multiple Errors Detected.

- **RECERR15: Recoverable Error in the page between the 3840th and the 4095th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Errors Detected.

1 = Errors Detected. If MUL_ERROR is 0, a single correctable error was detected. Otherwise, multiple uncorrected errors were detected

- **ECCERR15: ECC Error in the page between the 3840th and the 4095th bytes**

Fixed to 0 if TYPECORREC = 0

0 = No Errors Detected.

1 = A single bit error occurred in the ECC bytes.

Read ECC Parity 15 register, the error occurred at the location which contains a 1 in the least significant 24 bits.

- **MULERR15: Multiple Error in the page between the 3840th and the 4095th bytes**

Fixed to 0 if TYPECORREC = 0.

0 = No Multiple Errors Detected.

1 = Multiple Errors Detected.

26.5 Registers for 1 ECC for a page of 512/1024/2048/4096 bytes

26.5.1 ECC Parity Register 0

Name: ECC_PR0

Address: 0xFFFFE20C

Access: Read-only

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

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: Word Address**

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.

26.5.2 ECC Parity Register 1

Name: ECC_PR1

Address: 0xFFFFE210

Access: Read-only

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

- **NPARITY:**

Parity N

26.6 Registers for 1 ECC per 512 bytes for a page of 512/2048/4096 bytes, 8-bit word

26.6.1 ECC Parity Register 0

Name: ECC_PR0

Address: 0xFFFFE20C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
NPARITY0							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NPARITY0				WORDADD0			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WORDADDR0					BITADDR0		

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.

- **BITADDR0: corrupted Bit Address in the page between the first byte and the 511th bytes**

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.

- **WORDADDR0: corrupted Word Address in the page between the first byte and the 511th bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY0:**

Parity N

26.6.2 ECC Parity Register 1

Name: ECC_PR1

Address: 0xFFFFE210

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
NPARITY1							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NPARITY1				WORDADD1			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WORDADDR1					BITADDR1		

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.

- **BITADDR1: corrupted Bit Address in the page between the 512th and the 1023rd bytes**

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.

- **WORDADDR1: corrupted Word Address in the page between the 512th and the 1023rd bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY1:**

Parity N

26.6.3 ECC Parity Register 2

Name: ECC_PR2

Address: 0xFFFFE218

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
NPARITY2							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NPARITY2				WORDADDR2			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WORDADDR2					BITADDR2		

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.

- **BITADDR2: corrupted Bit Address in the page between the 1023rd and the 1535th bytes**

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.

- **WORDADDR2: corrupted Word Address in the page in the page between the 1023rd and the 1535th bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY2:**

Parity N

26.6.4 ECC Parity Register 3

Name: ECC_PR3

Address: 0xFFFFE21C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
NPARITY3							
15	14	13	12	11	10	9	8
NPARITY3				WORDADDR3			
7	6	5	4	3	2	1	0
WORDADDR3					BITADDR3		

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.

- **BITADDR3: corrupted Bit Address in the page between the1536th and the 2047th bytes**

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.

- **WORDADDR3 corrupted Word Address in the page between the 1536th and the 2047th bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY3**

Parity N

26.6.5 ECC Parity Register 4

Name: ECC_PR4

Address: 0xFFFFE220

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
NPARITY4							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NPARITY4				WORDADDR4			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WORDADDR4					BITADDR4		

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.

- **BITADDR4: corrupted Bit Address in the page between the 2048th and the 2559th bytes**

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.

- **WORDADDR4: corrupted Word Address in the page between the 2048th and the 2559th bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY4:**

Parity N

26.6.6 ECC Parity Register 5

Name: ECC_PR5

Address: 0xFFFFE224

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
NPARITY5							
15	14	13	12	11	10	9	8
NPARITY5				WORDADDR5			
7	6	5	4	3	2	1	0
WORDADDR5					BITADDR5		

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.

- **BITADDR5: corrupted Bit Address in the page between the 2560th and the 3071st bytes**

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.

- **WORDADDR5: corrupted Word Address in the page between the 2560th and the 3071st bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY5:**

Parity N

26.6.7 ECC Parity Register 6

Name: ECC_PR6

Address: 0xFFFFE228

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
NPARITY6							
15	14	13	12	11	10	9	8
NPARITY6				WORDADDR6			
7	6	5	4	3	2	1	0
WORDADDR6					BITADDR6		

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.

- **BITADDR6: corrupted Bit Address in the page between the 3072nd and the 3583rd bytes**

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.

- **WORDADDR6: corrupted Word Address in the page between the 3072nd and the 3583rd bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY6:**

Parity N

26.6.8 ECC Parity Register 7

Name: ECC_PR7

Address: 0xFFFFE22C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
NPARITY7							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NPARITY7				WORDADDR7			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WORDADDR7					BITADDR7		

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.

- **BITADDR7: corrupted Bit Address in the page between the 3584h and the 4095th bytes**

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.

- **WORDADDR7: corrupted Word Address in the page between the 3584th and the 4095th bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY7:**

Parity N

26.7 Registers for 1 ECC per 256 bytes for a page of 512/2048/4096 bytes, 8-bit word

26.7.1 ECC Parity Register 0

Name: ECC_PR0

Address: 0xFFFFE20C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
0	NPARITY0						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NPARITY0				0	WORDADDR0		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WORDADDR0					BITADDR0		

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.

- **BITADDR0: corrupted Bit Address in the page between the first byte and the 255th bytes**

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.

- **WORDADDR0: corrupted Word Address in the page between the first byte and the 255th bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY0:**

Parity N

26.7.2 ECC Parity Register 1

Name: ECC_PR1

Address: 0xFFFFE210

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
0	NPARITY1						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NPARITY1				0	WORDADDR1		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WORDADDR1					BITADDR1		

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

- **BITADDR1: corrupted Bit Address in the page between the 256th and the 511th bytes**

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.

- **WORDADDR1: corrupted Word Address in the page between the 256th and the 511th bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY1:**

Parity N

26.7.3 ECC Parity Register 2

Name: ECC_PR2

Address: 0xFFFFE218

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY2						
15	14	13	12	11	10	9	8
NPARITY2				0	WORDADD2		
7	6	5	4	3	2	1	0
WORDADDR2					BITADDR2		

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.

- **BITADDR2: corrupted Bit Address in the page between the 512th and the 767th bytes**

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.

- **WORDADDR2: corrupted Word Address in the page between the 512th and the 767th bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY2:**

Parity N

26.7.4 ECC Parity Register 3

Name: ECC_PR3

Address: 0xFFFFE21C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY3						
15	14	13	12	11	10	9	8
NPARITY3				0	WORDADDR3		
7	6	5	4	3	2	1	0
WORDADDR3					BITADDR3		

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.

- **BITADDR3: corrupted Bit Address in the page between the 768th and the 1023rd bytes**

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.

- **WORDADDR3: corrupted Word Address in the page between the 768th and the 1023rd bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless

- **NPARITY3:**

Parity N

26.7.5 ECC Parity Register 4

Name: ECC_PR4

Address: 0xFFFFE220

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
0	NPARITY4						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NPARITY4				0	WORDADDR4		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WORDADDR4					BITADDR4		

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

- **BITADDR4: corrupted bit address in the page between the 1024th and the 1279th bytes**

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.

- **WORDADDR4: corrupted word address in the page between the 1024th and the 1279th bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY4**

Parity N

26.7.6 ECC Parity Register 5

Name: ECC_PR5

Address: 0xFFFFE224

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
0	NPARITY5						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NPARITY5				0	WORDADDR5		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WORDADDR5					BITADDR5		

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.

- **BITADDR5: corrupted Bit Address in the page between the 1280th and the 1535th bytes**

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.

- **WORDADDR5: corrupted Word Address in the page between the 1280th and the 1535th bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY5:**

Parity N

26.7.7 ECC Parity Register 6

Name: ECC_PR6

Address: 0xFFFFE228

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
0	NPARITY6						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NPARITY6				0	WORDADDR6		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WORDADDR6					BITADDR6		

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.

- **BITADDR6: corrupted bit address in the page between the 1536th and the1791st bytes**

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.

- **WORDADDR6: corrupted word address in the page between the 1536th and the1791st bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY6:**

Parity N

26.7.8 ECC Parity Register 7

Name: ECC_PR7

Address: 0xFFFFE22C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
0	NPARITY7						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NPARITY7				0	WORDADDR7		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WORDADDR7					BITADDR7		

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.

- **BITADDR7: corrupted Bit Address in the page between the 1792nd and the 2047th bytes**

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.

- **WORDADDR7: corrupted Word Address in the page between the 1792nd and the 2047th bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY7:**

Parity N

26.7.9 ECC Parity Register 8

Name: ECC_PR8

Address: 0xFFFFE230

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY8						
15	14	13	12	11	10	9	8
NPARITY8				0	WORDADDR8		
7	6	5	4	3	2	1	0
WORDADDR8					BITADDR8		

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.

- **BITADDR8: corrupted Bit Address in the page between the 2048th and the2303rd bytes**

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.

- **WORDADDR8: corrupted Word Address in the page between the 2048th and the 2303rd bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY8:**

Parity N.

26.7.10 ECC Parity Register 9

Name: ECC_PR9

Address: 0xFFFFE234

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
0	NPARITY9						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NPARITY9				0	WORDADDR9		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WORDADDR9					BITADDR9		

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

- **BITADDR9: corrupted bit address in the page between the 2304th and the 2559th bytes**

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.

- **WORDADDR9: corrupted word address in the page between the 2304th and the 2559th bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless

- **NPARITY9**

Parity N

26.7.11 ECC Parity Register 10

Name: ECC_PR10

Address: 0xFFFFE238

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY10						
15	14	13	12	11	10	9	8
NPARITY10				0	WORDADDR10		
7	6	5	4	3	2	1	0
WORDADDR10					BITADDR10		

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.

- **BITADDR10: corrupted Bit Address in the page between the 2560th and the2815th bytes**

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.

- **WORDADDR10: corrupted Word Address in the page between the 2560th and the 2815th bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY10:**

Parity N

26.7.12 ECC Parity Register 11

Name: ECC_PR11

Address: 0xFFFFE23C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY11						
15	14	13	12	11	10	9	8
NPARITY11				0	WORDADDR11		
7	6	5	4	3	2	1	0
WORDADDR11					BITADDR11		

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.

- **BITADDR11: corrupted Bit Address in the page between the 2816th and the 3071st bytes**

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.

- **WORDADDR11: corrupted Word Address in the page between the 2816th and the 3071st bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY11:**

Parity N

26.7.13 ECC Parity Register 12

Name: ECC_PR12

Address: 0xFFFFE240

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
0	NPARITY12						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NPARITY12				0	WORDADDR12		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WORDADDR12					BITADDR12		

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.

- **BITADDR12; corrupted Bit Address in the page between the 3072nd and the 3327th bytes**

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.

- **WORDADDR12: corrupted Word Address in the page between the 3072nd and the 3327th bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY12:**

Parity N

26.7.14 ECC Parity Register 13

Name: ECC_PR13

Address: 0xFFFFE244

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
0	NPARITY13						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NPARITY13				0	WORDADDR13		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WORDADDR13					BITADDR13		

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.

- **BITADDR13: corrupted Bit Address in the page between the 3328th and the 3583rd bytes**

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.

- **WORDADDR13: corrupted Word Address in the page between the 3328th and the 3583rd bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY13:**

Parity N

26.7.15 ECC Parity Register 14

Name: ECC_PR14

Address: 0xFFFFE248

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
0	NPARITY14						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NPARITY14				0	WORDADDR14		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WORDADDR14					BITADDR14		

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.

- **BITADDR14: corrupted Bit Address in the page between the 3584th and the 3839th bytes**

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.

- **WORDADDR14: corrupted Word Address in the page between the 3584th and the 3839th bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY14:**

Parity N

26.7.16 ECC Parity Register 15

Name: ECC_PR15

Address: 0xFFFFE24C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
0	NPARITY15						
15	14	13	12	11	10	9	8
NPARITY15				0	WORDADDR15		
7	6	5	4	3	2	1	0
WORDADDR15					BITADDR15		

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

- **BITADDR15: corrupted Bit Address in the page between the 3840th and the 4095th bytes**

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.

- **WORDADDR15: corrupted Word Address in the page between the 3840th and the 4095th bytes**

During a page read, this value contains the word address (8-bit word) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **NPARITY15**

Parity N

27. DMA Controller (DMAC)

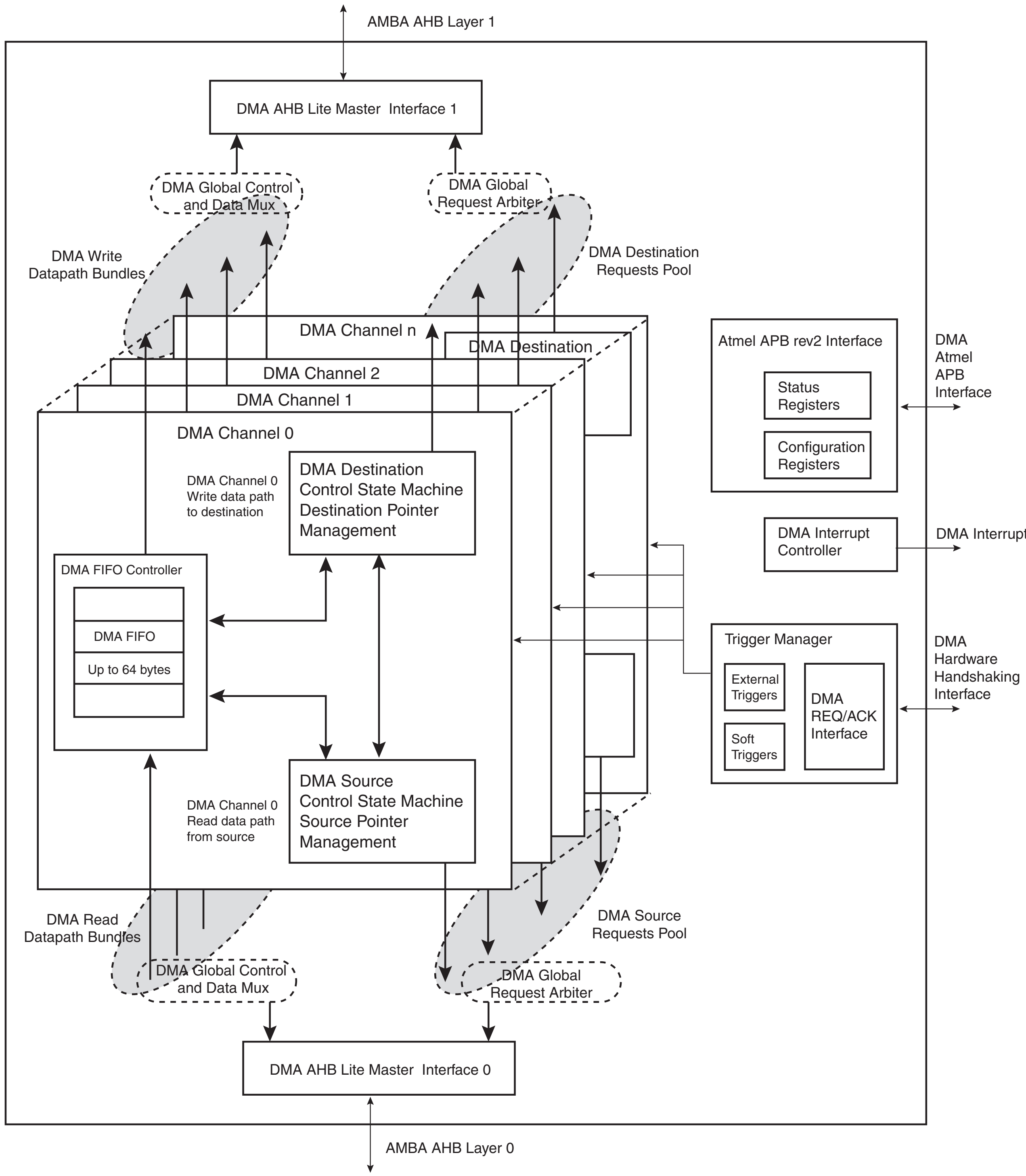
27.1 Description

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.

27.2 Block Diagram

Figure 27-1. DMA Controller (DMAC) Block Diagram



27.3 Functional Description

27.3.1 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. If the source peripheral is not memory, then a source handshaking interface is assigned to the channel. If the destination peripheral is not memory, then a destination handshaking interface is assigned to the channel. Source and destination handshaking interfaces can be assigned dynamically by programming the channel registers.

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.

Handshaking interface: A set of signal registers that conform to a protocol and *handshake* between the DMAC and source or destination peripheral to control the transfer of a single or chunk transfer between them. This interface is used to request, acknowledge, and control a DMAC transaction. A channel can receive a request through one of two types of handshaking interface: hardware or software.

Hardware handshaking interface: Uses hardware signals to control the transfer of a single or chunk transfer between the DMAC and the source or destination peripheral.

Software handshaking interface: Uses software registers to contr5ol the transfer of a single or chunk transfer between the DMAC and the source or destination peripheral. No special DMAC handshaking signals are needed on the I/O of the peripheral. This mode is useful for interfacing an existing peripheral to the DMAC without modifying it.

Flow controller: The device (either the DMAC or source/destination peripheral) that determines the length of and terminates a DMAC buffer transfer. If the length of a buffer is known before enabling the channel, then the DMAC should be programmed as the flow controller. If the length of a buffer is not known prior to enabling the channel, the source or destination peripheral needs to terminate a buffer transfer. In this mode, the peripheral is the flow controller.

Transfer hierarchy: Figure 27-2 on page 324 illustrates the hierarchy between DMAC transfers, buffer transfers, chunk or single, and AMBA transfers (single or burst) for non-memory peripherals. Figure 27-3 on page 324 shows the transfer hierarchy for memory.

Figure 27-2. DMAC Transfer Hierarchy for Non-Memory Peripheral

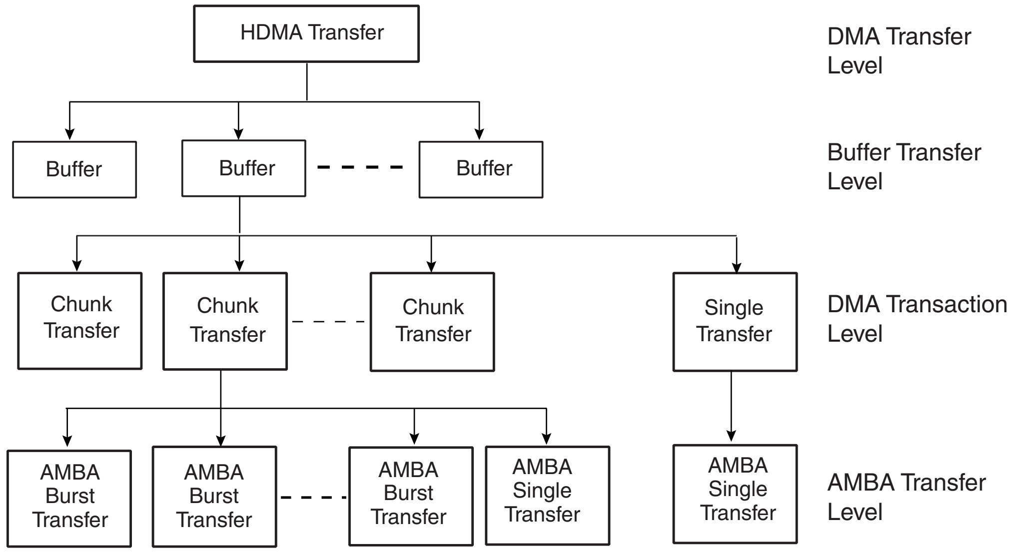
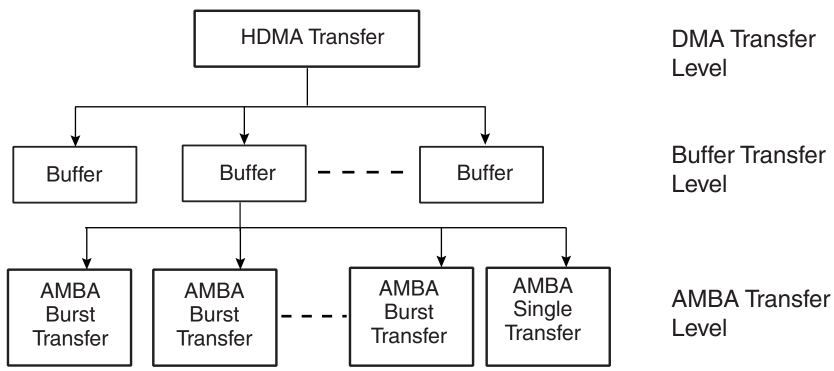


Figure 27-3. DMAC Transfer Hierarchy for Memory



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.

For transfers between the DMAC and a non-memory peripheral, a buffer is broken into a sequence of DMAC transactions (single and chunks). These are in turn broken into a sequence of AMBA transfers.

Transaction: A basic unit of a DMAC transfer as determined by either the hardware or software handshaking interface. A transaction is only relevant for transfers between the DMAC and a source or destination peripheral if the source or destination peripheral is a non-memory device. There are two types of transactions: single transfer and chunk transfer.

– **Single transfer:** The length of a single transaction is always 1 and is converted to a single AMBA access.

– **Chunk transfer:** The length of a chunk is programmed into the DMAC. The chunk is then converted into a sequence of AHB access.DMAC executes each AMBA burst transfer by performing incremental bursts that are no longer than 16 beats.

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.

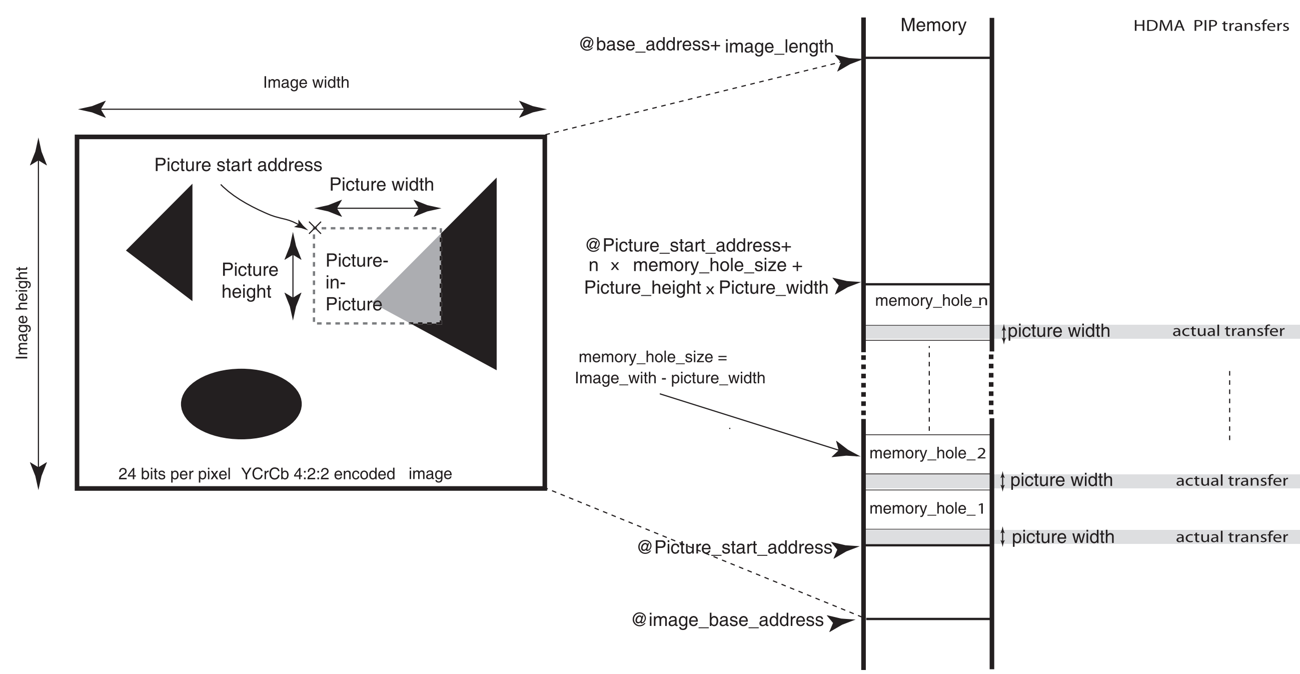
Multi-buffer DMAC transfer: A DMAC transfer may consist of multiple DMAC buffers. Multi-buffer DMAC transfers are supported through buffer chaining (linked list pointers), auto-reloading of channel registers, and contiguous buffers. The source and destination can independently select which method to 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 27-4 on page 325 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 27-4. Picture-In-Picture Mode Support



Channel locking: Software can program a channel to keep the AHB master interface by locking the arbitration for the master bus interface for the duration of a DMAC transfer, buffer, or chunk.

Bus locking: Software can program a channel to maintain control of the AMBA bus by asserting hmastlock for the duration of a DMAC transfer, buffer, or transaction (single or chunk). Channel locking is asserted for the duration of bus locking at a minimum.

27.3.2 Memory Peripherals

Figure 27-3 on page 324 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 handshaking 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. By using the handshaking interface, the peripheral can signal to the DMAC that it is ready to transmit/receive data, and then the DMAC can access the peripheral without the peripheral inserting wait states onto the bus.

27.3.3 Handshaking Interface

Handshaking interfaces are used at the transaction level to control the flow of single or chunk transfers. The operation of the handshaking interface is different and depends on whether the peripheral or the DMAC is the flow controller.

The peripheral uses the handshaking interface to indicate to the DMAC that it is ready to transfer/accept data over the AMBA bus. A non-memory peripheral can request a DMAC transfer through the DMAC using one of two handshaking interfaces:

- Hardware handshaking
- Software handshaking

Software selects between the hardware or software handshaking interface on a per-channel basis. Software handshaking is accomplished through memory-mapped registers, while hardware handshaking is accomplished using a dedicated handshaking interface.

27.3.3.1 Software Handshaking

When the slave peripheral requires the DMAC to perform a DMAC transaction, it communicates this request by sending an interrupt to the CPU or interrupt controller.

The interrupt service routine then uses the software registers to initiate and control a DMAC transaction. These software registers are used to implement the software handshaking interface.

The SRC_H2SEL/DST_H2SEL bit in the DMAC_CFGx channel configuration register must be set to zero to enable software handshaking.

When the peripheral is not the flow controller, then the last transaction register DMAC_LAST is not used, and the values in these registers are ignored.

27.3.3.2 Chunk Transactions

Writing a 1 to the DMAC_CREQ[2x] register starts a source chunk transaction request, where x is the channel number. Writing a 1 to the DMAC_CREQ[2x+1] register starts a destination chunk transfer request, where x is the channel number.

Upon completion of the chunk transaction, the hardware clears the DMAC_CREQ[*2x*] or DMAC_CREQ[2x+1].

27.3.3.3 Single Transactions

Writing a 1 to the DMAC_SREQ[2x] register starts a source single transaction request, where x is the channel number. Writing a 1 to the DMAC_SREQ[2x+1] register starts a destination single transfer request, where x is the channel number.

Upon completion of the chunk transaction, the hardware clears the DMAC_SREQ[x] or DMAC_SREQ[2x+1].

Software can poll the relevant channel bit in the DMAC_CREQ[2x]/DMAC_CREQ[2x+1] and DMAC_SREQ[x]/DMAC_SREQ[2x+1] registers. When both are 0, then either the requested chunk or single transaction has completed.

27.3.4 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 reprogrammed 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.

27.3.4.1 Multi-buffer Transfers

27.3.4.2 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.

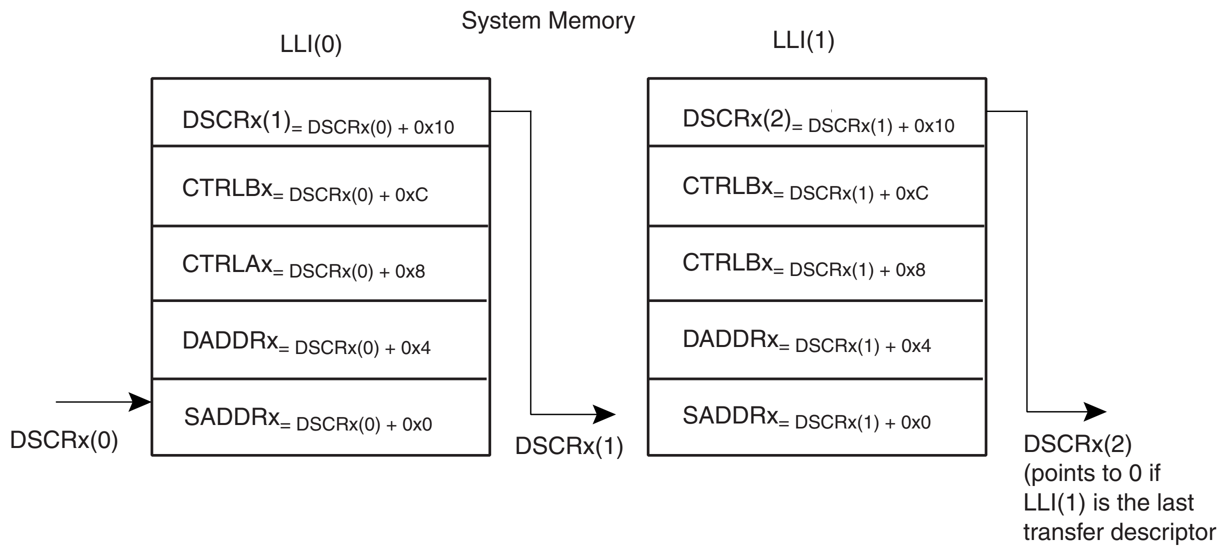
The DMAC_SADDRx, DMAC_DADDRx, DMAC_DSCRx, DMAC_CTRLAx and DMAC_CTRLBx registers are fetched from system memory on an LLI update. The updated content of the DMAC_CTRLAx register is written back to memory on buffer completion. Figure 27-5 on page

328 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 27-5. Multi Buffer Transfer Using Linked List



27.3.4.3 Programming DMAC for Multiple Buffer Transfers

Table 27-1. 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 is 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.

27.3.4.4 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 27-1 on page 329, 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.

27.3.4.5 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.

27.3.4.6 *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.

27.3.4.7 *Ending Multi-buffer Transfers*

All multi-buffer transfers must end as shown in Row 1 of Table 27-1 on page 329. 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 27-1 on page 329, (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_CRTLBx.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.

27.3.5 Programming a Channel

Four registers, the DMAC_DSCRx, the DMAC_CTRLAx, the DMAC_CTRLBx and DMAC_CFGx, 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 27-1 on page 329.

The "BTSIZE, SADDR and DADDR" columns indicate where the values of DMAC_SARx, DMAC_DARx, DMAC_CTLx, and DMAC_LLPx are obtained for the next buffer transfer when multi-buffer DMAC transfers are enabled.

27.3.5.1 Programming Examples

27.3.5.2 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_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 1 as shown in Table 27-1 on page 329. Program the DMAC_CTRLBx 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_CTRLAx and DMAC_CTRLBx registers for channel x. For example, in the register, you can program the following:
   – i. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the FC of the DMAC_CTRLBx register.
   – ii. 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_CFGx register for channel x.
   – i. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the SRC_H2SEL/DST_H2SEL bits, respectively. Writing a '1' activates the hardware handshaking interface to handle source/destination requests. Writing a '0' activates the software handshaking interface to handle source/destination requests.

– ii. If the hardware handshaking interface is activated for the source or destination peripheral, assign a handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DST_PER bits, respectively.

f. If source picture-in-picture mode is enabled (DMAC_CTRLBx.SRC_PIP is enabled), program the DMAC_SPIPx register for channel x.

g. If destination picture-in-picture mode is enabled (DMAC_CTRLBx.DST_PIP is enabled), program the DMAC_DPIPx 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. Source and destination request single and chunk DMAC transactions to transfer the buffer of data (assuming non-memory peripherals). The DMAC acknowledges at the completion of every transaction (chunk and single) in the buffer and carry out the buffer transfer.
6. 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.

27.3.5.3 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 27-6 on page 334) for channel x. For example, in the register, you can program the following:

a. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the FC of the DMAC_CTRLBx register.

b. 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.

a. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the SRC_H2SEL/DST_H2SEL bits, respectively. Writing a '1' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '0' activates the software handshaking interface to handle source/destination requests.

b. If the hardware handshaking interface is activated for the source or destination peripheral, assign the handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DST_PER bits, respectively.

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 27-1 on page 329. The

LLI.DMAC_CTRLBx register of the last Linked List Item must be set as described in Row 1 of Table 27-1. Figure 27-5 on page 328 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 27-1 on page 329.
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).

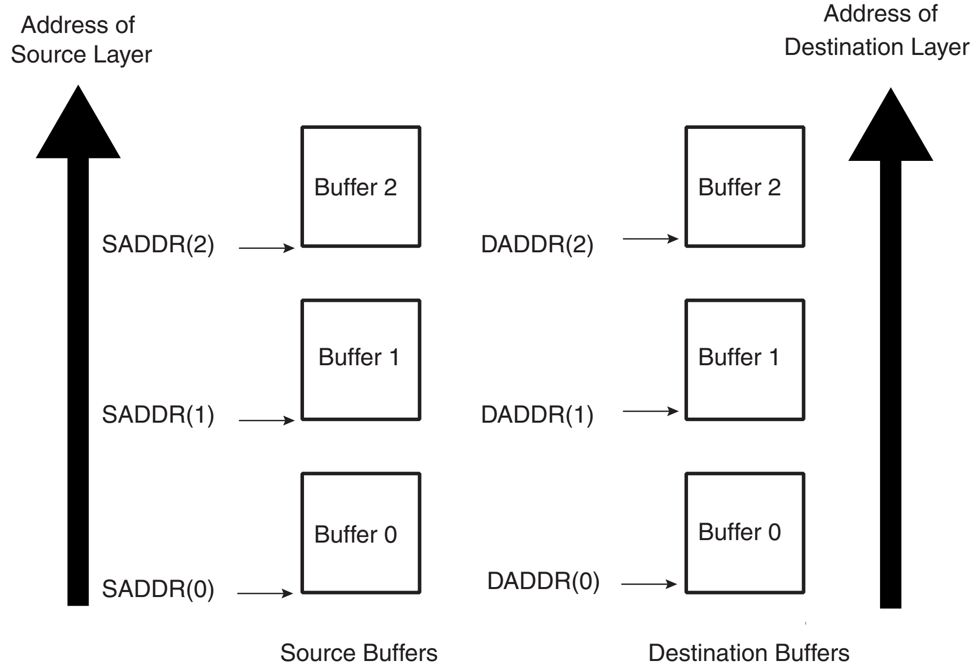
Note: The LLI.DMAC_SADDRx, LLI. DMAC_DADDRx, LLI.DMAC_DSCRx, LLI.DMAC_CTRLAx and LLI.DMAC_CTRLBx registers are fetched. The DMAC automatically reprograms the DMAC_SADDRx, DMAC_DADDRx, DMAC_DSCRx, DMAC_CTRLBx and DMAC_CTRLAx channel registers from the DMAC_DSCRx(0).

15. Source and destination request single and chunk DMAC transactions to transfer the buffer of data (assuming non-memory peripheral). The DMAC acknowledges at the completion of every transaction (chunk and single) in the buffer and carry out the buffer transfer.
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 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.

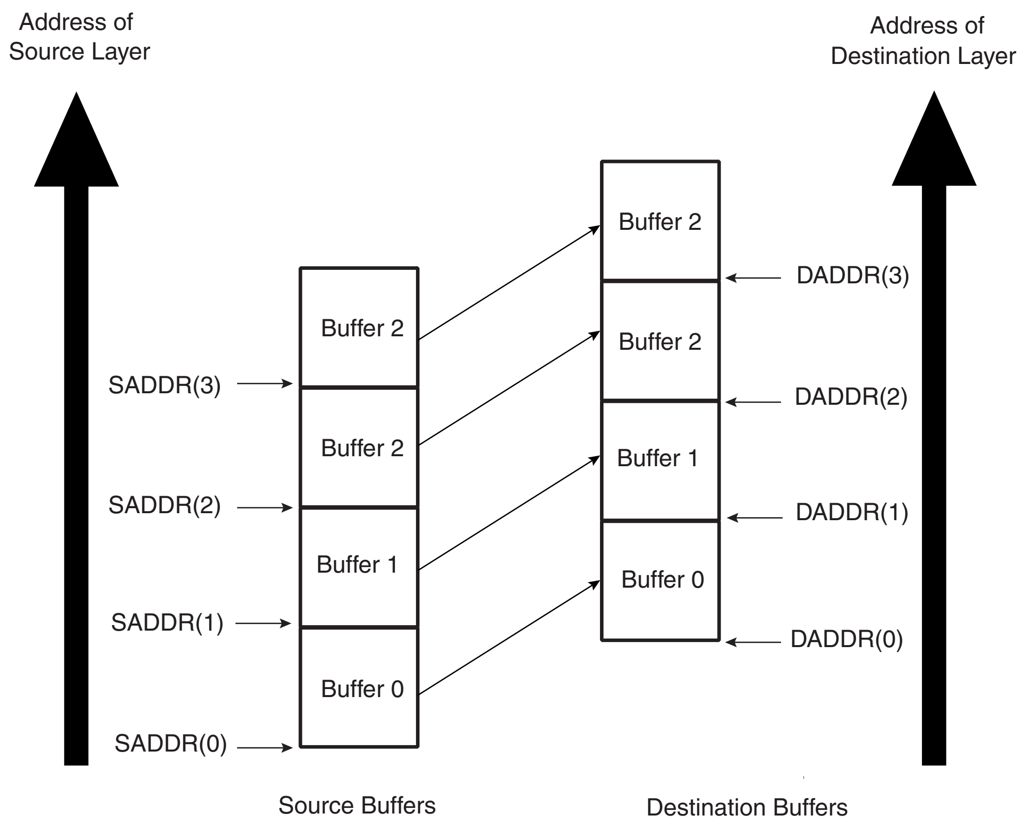
17. The DMAC does not wait for the buffer interrupt to be cleared, but continues fetching the next LLI from the memory location pointed to by current DMAC_DSCRx register and automatically reprograms the DMAC_SADDRx, DMAC_DADDRx, DMAC_DSCRx, DMAC_CTRLAx and DMAC_CTRLBx channel registers. The DMAC transfer 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 27-1 on page 329. 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 27-6 on page 334.

Figure 27-6. Multi-buffer with Linked List Address for Source and Destination



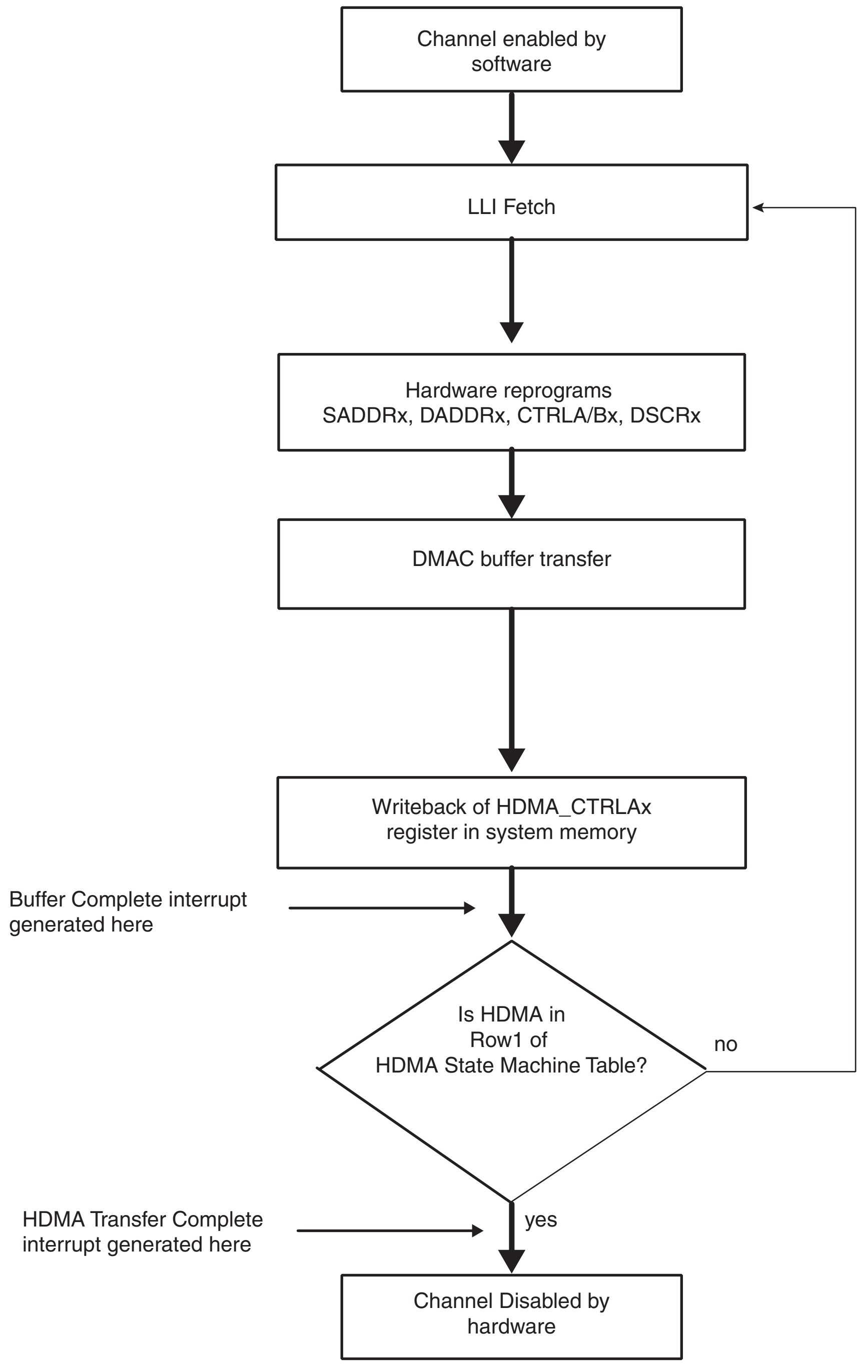
If the user needs to execute a DMAC transfer where the source and destination address are contiguous but the amount of data to be transferred is greater than the maximum buffer size DMAC_CTRLAx.BTSIZE, then this can be achieved using the type of multi-buffer transfer as shown in Figure 27-7 on page 335.

Figure 27-7. Multi-buffer with Linked Address for Source and Destination Buffers are Contiguous



The DMAC transfer flow is shown in Figure 27-8 on page 336.

Figure 27-8. DMAC Transfer Flow for Source and Destination Linked List Address



27.3.5.4 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 27-1 on page 329. 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:

– i. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the FC of the DMAC_CTRLBx register.

– ii. 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.

– i. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the SRC_H2SEL/DST_h2SEL bits, respectively. Writing a '1' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '0' activates the software handshaking interface to handle source/destination requests.

– ii. If the hardware handshaking interface is activated for the source or destination peripheral, assign handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DST_PER bits, respectively.

3. After the DMAC selected channel has been programmed, enable the channel by writing a '1' to the DMAC_CHER.ENABLE[n] bit where is the channel number. Make sure that bit 0 of the DMAC_EN register is enabled.

4. Source and destination request single and chunk DMAC transactions to transfer the buffer of data (assuming non-memory peripherals). The DMAC acknowledges on completion of each chunk/single transaction and carry out the buffer transfer.

5. 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 27-1 on page 329. 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.

6. 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 27-1 on page 329. 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 27-1 on page 329 before the last buffer of the DMAC transfer has completed. The transfer is similar to that shown in Figure 27-9 on page 338. The DMAC transfer flow is shown in Figure 27-10 on page 339.

Figure 27-9. Multi-buffer DMAC Transfer with Source and Destination Address Auto-reloaded

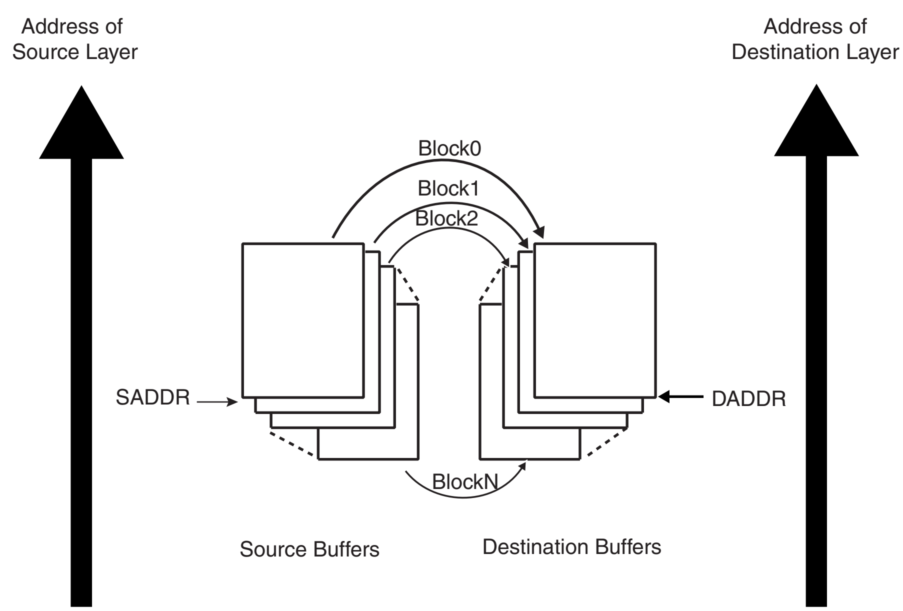
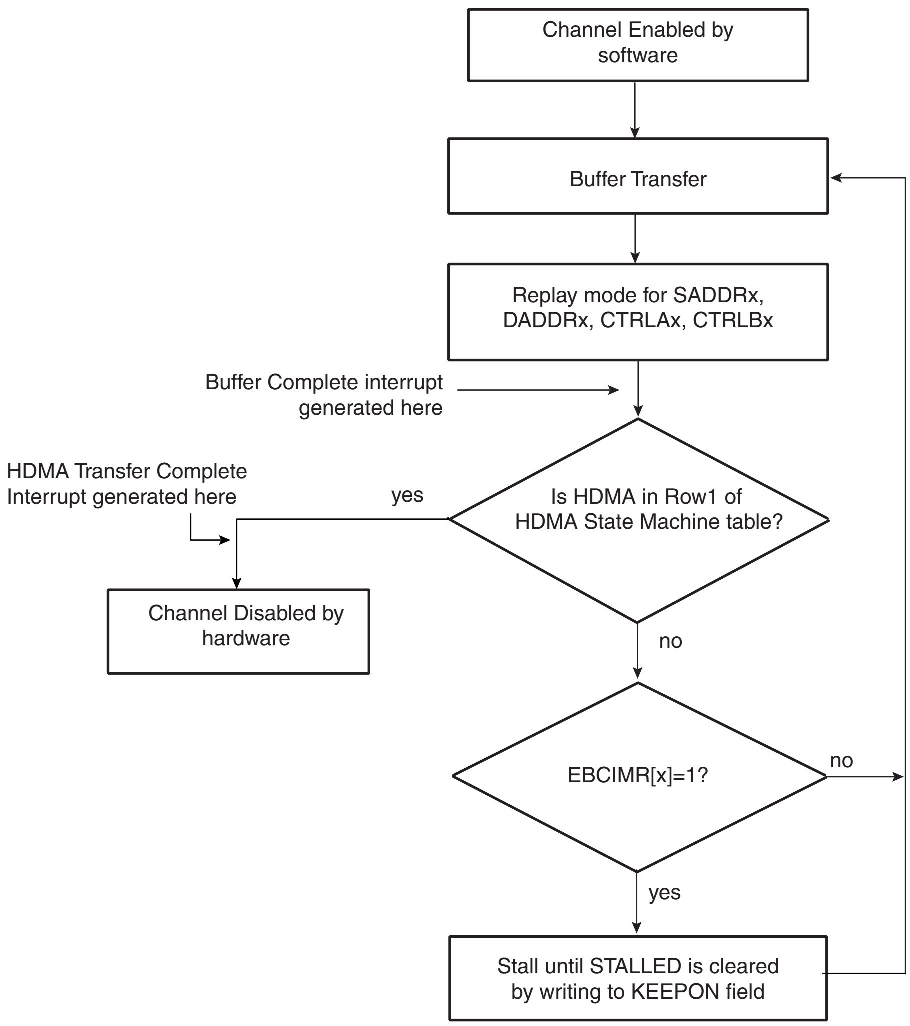


Figure 27-10. DMAC Transfer Flow for Source and Destination Address Auto-reloaded



27.3.5.5 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:
   a. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control peripheral by programming the FC of the DMAC_CTRLBx register.
   b. 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.
   a. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the SRC_H2SEL/DST_H2SEL bits, respectively. Writing a ‘1’ activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a ‘0’ activates the software handshaking interface source/destination requests.
   b. If the hardware handshaking interface is activated for the source or destination peripheral, assign handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DST_PER bits, respectively.
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 27-1 on page 329 while the LLI.DMAC_CTRLBx register of the last Linked List item must be set as described in Row 1 of Table 27-1. Figure 27-5 on page 328 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 27-1 on page 329.
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_LLPx LLI.DMAC_CTRLAx and LLI.DMAC_CTRLBx registers are fetched. The LLI.DMAC_SADDRx register although fetched is not used.

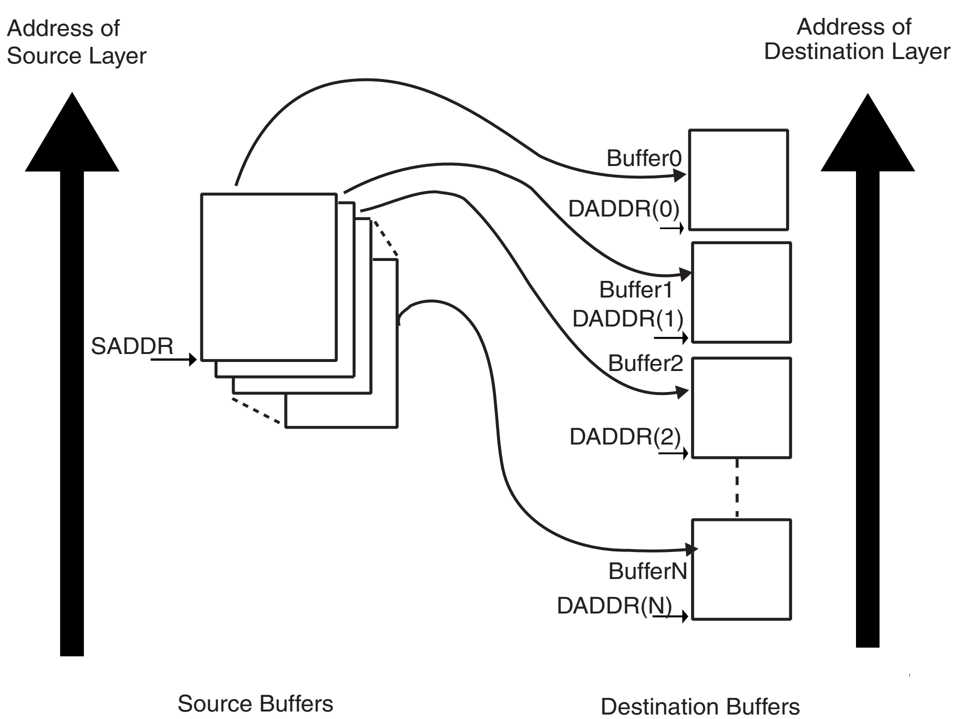
16. Source and destination request single and chunk DMAC transactions to transfer the buffer of data (assuming non-memory peripherals). DMAC acknowledges at the completion of every transaction (chunk and single) in the buffer and carry out the buffer transfer.
17. The DMAC_CTRLAx register is written 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.

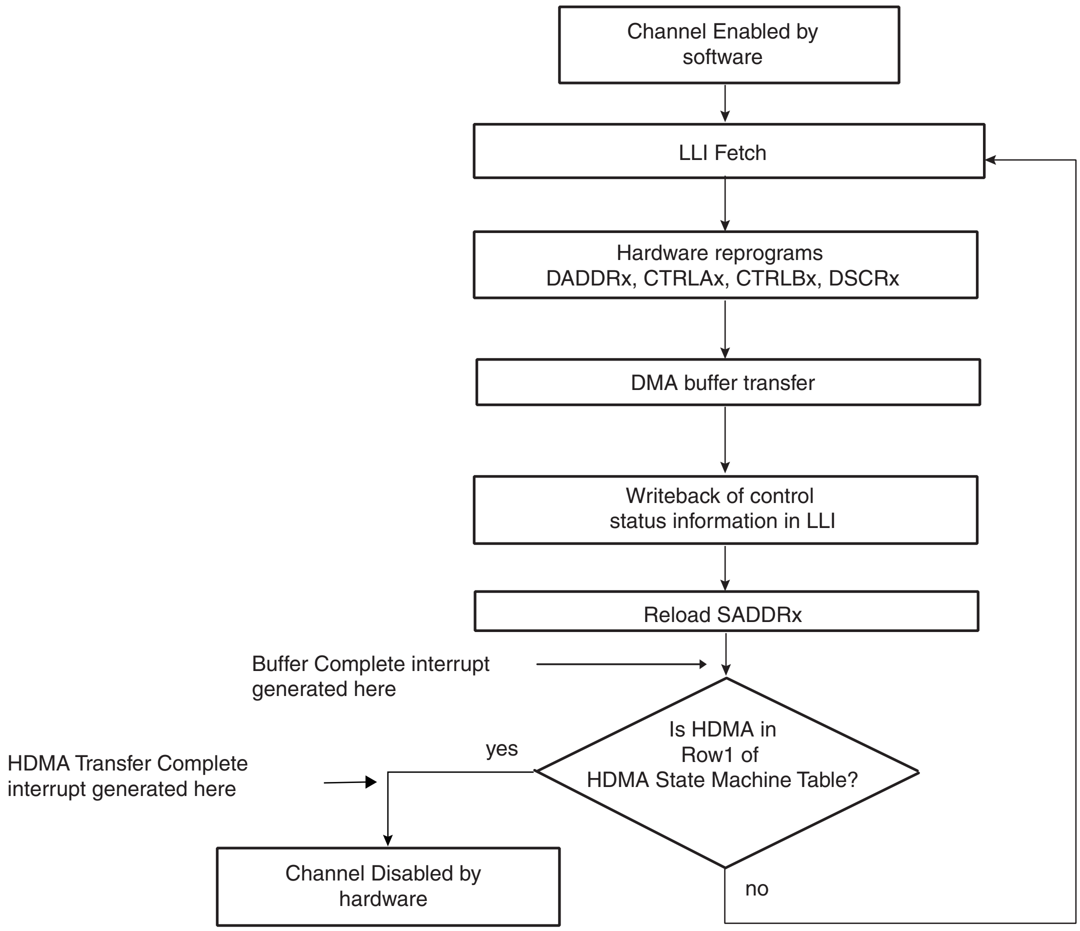
18. The DMAC reloads the DMAC_SADDRx register from the initial value. Hardware sets the buffer complete interrupt. The DMAC samples the row number as shown in Table 27-1 on page 329. If the DMAC is in Row 1, then the DMAC transfer has completed. Hardware sets the transfer complete interrupt and disables 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 27-1 on page 329, the following step is performed.
19. 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 27-1 on page 329. The DMAC transfer might look like that shown in Figure 27-11 on page 341.

Figure 27-11. Multi-buffer DMAC Transfer with Source Address Auto-reloaded and Linked List Destination Address



The DMAC Transfer flow is shown in Figure 27-12 on page 342.

Figure 27-12. DMAC Transfer Flow for Replay Mode at Source and Linked List Destination Address



27.3.5.6 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 27-1 on page 329. 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:
   – i. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the FC of the DMAC_CTRLBx register.

– ii. 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.

– i. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the SRC_H2SEL/DST_H2SEL bits, respectively. Writing a ‘1’ activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a ‘0’ activates the software handshaking interface to handle source/destination requests.

– ii. If the hardware handshaking interface is activated for the source or destination peripheral, assign handshaking interface to the source and destination peripheral. This requires programming the SRC_PER and DST_PER bits, respectively.

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. Source and destination request single and chunk DMAC transactions to transfer the buffer of data (assuming non-memory peripherals). The DMAC acknowledges at the completion of every transaction (chunk and single) in the buffer and carries out the buffer transfer.
6. 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 27-1 on page 329. 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.
7. 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 27-1 on page 329. 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 27-1 on page 329.

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 27-1 on page 329 before the last buffer of the DMAC transfer has completed.

The transfer is similar to that shown in Figure 27-13 on page 344.

The DMAC Transfer flow is shown in Figure 27-14 on page 345.

Figure 27-13. Multi-buffer Transfer with Source Address Auto-reloaded and Contiguous Destination Address

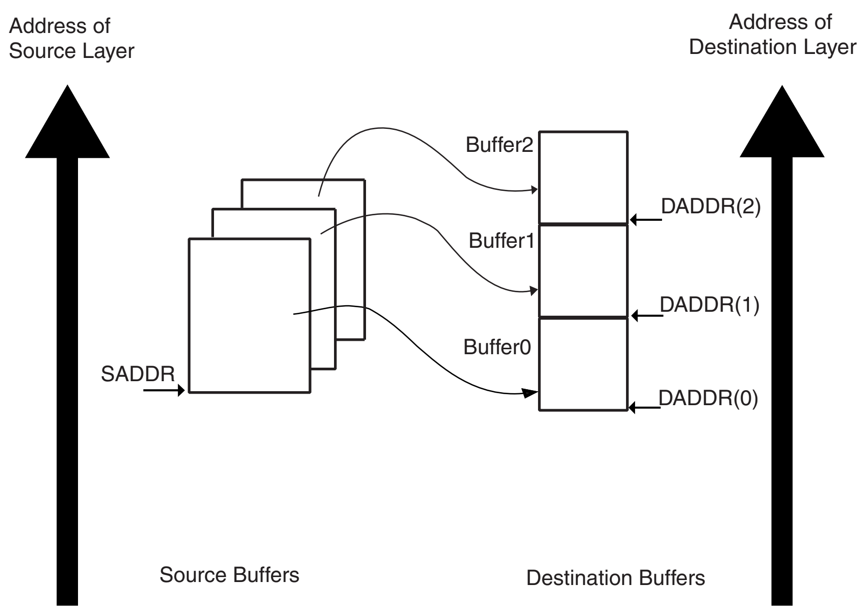
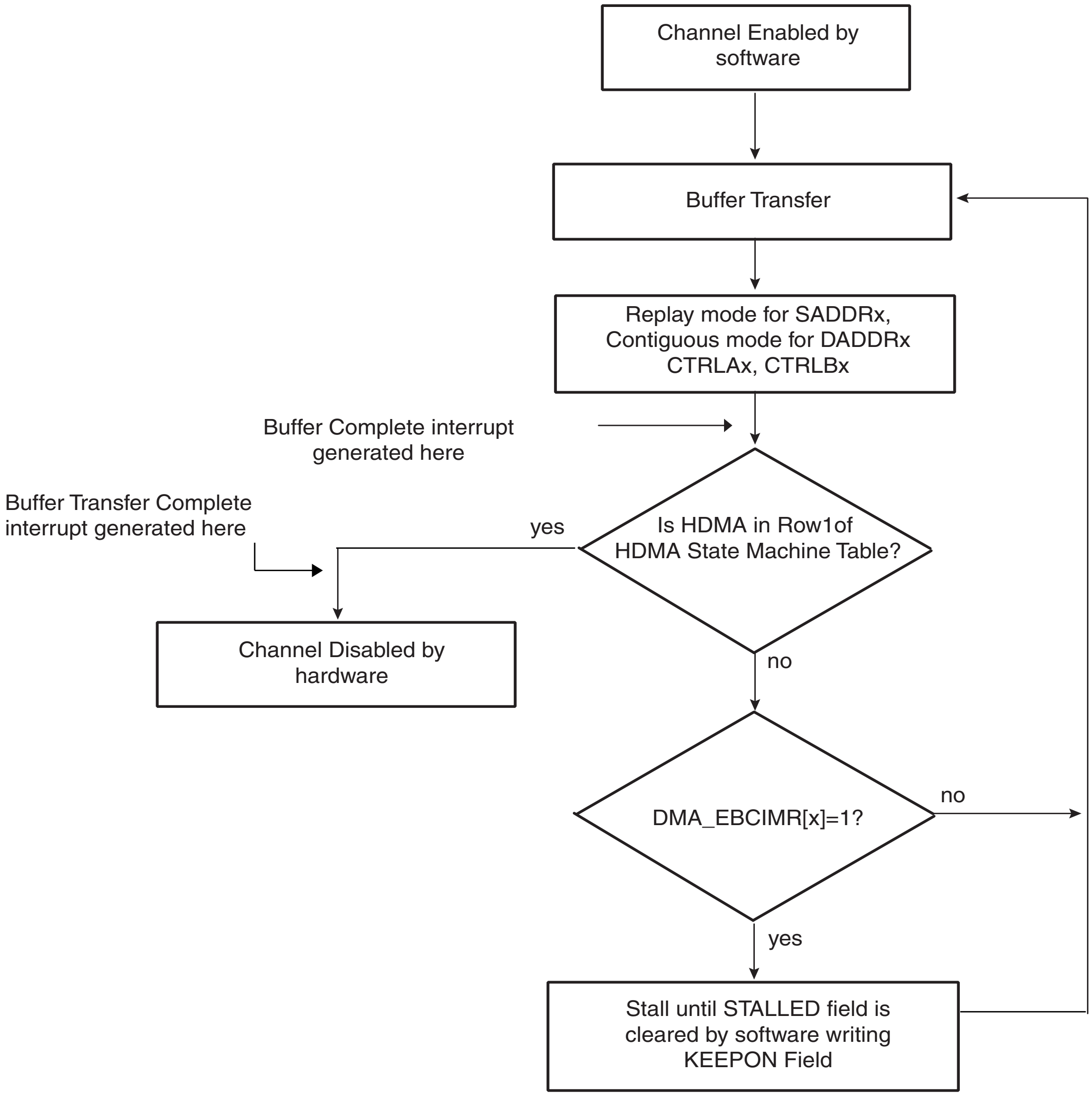


Figure 27-14. DMAC Transfer Replay Mode is Enabled for the Source and Contiguous Destination Address



27.3.5.7 *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:
   a. Set up the transfer type (memory or non-memory peripheral for source and destination) and flow control device by programming the FC of the DMAC_CTRLBx register.
   b. 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.
   a. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the SRC_H2SEL/DST_H2SEL bits, respectively. Writing a '1' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '0' activates the software handshaking interface to handle source/destination requests.
   b. If the hardware handshaking interface is activated for the source or destination peripheral, assign handshaking interface to the source and destination peripherals. This requires programming the SRC_PER and DST_PER bits, respectively.
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 27-1 on page 329, while the LLI.DMAC_CTRLBx register of the last Linked List item must be set as described in Row 1 of Table 27-1. Figure 27-5 on page 328 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 27-1 on page 329
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. Source and destination requests single and chunk DMAC transactions to transfer the buffer of data (assuming non-memory peripherals). The DMAC acknowledges at the completion of every transaction (chunk and single) in the buffer and carry out the buffer transfer
17. 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

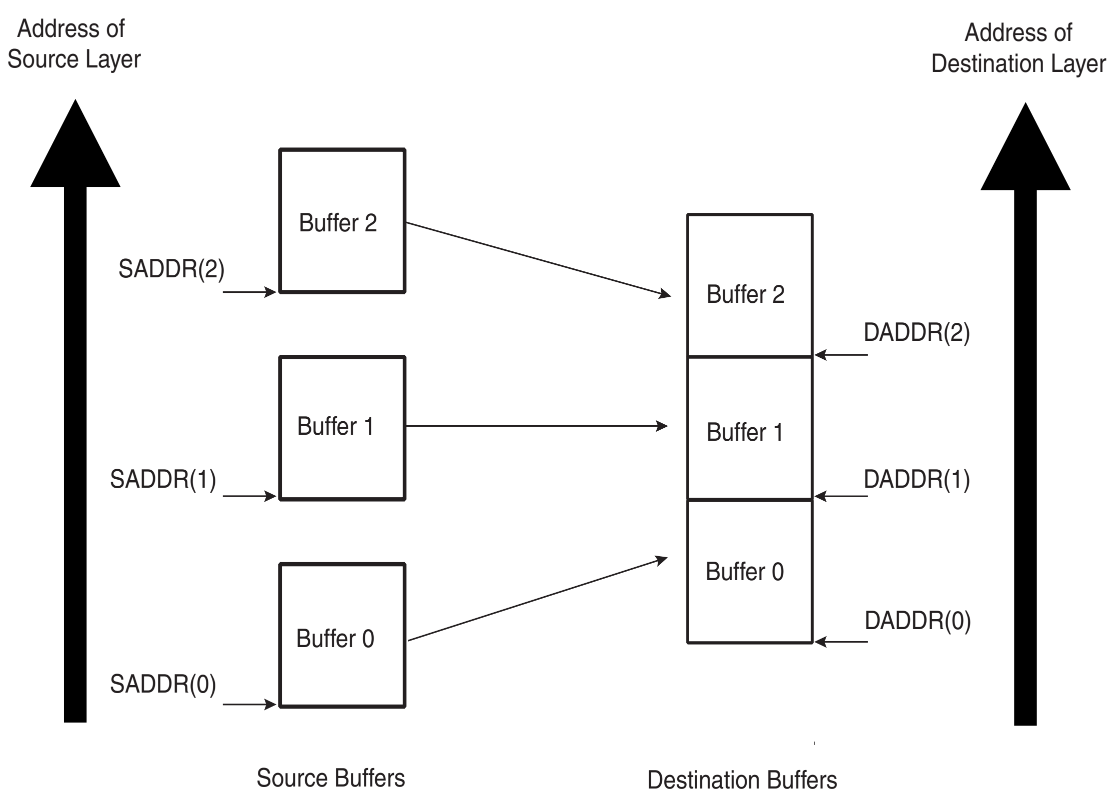
the linked list item fetched prior to the start of the buffer transfer. Only DMAC_CTRLAx register is written out because only the DMAC_CTRLAx.BTSIZE and DMAC_CTRLAX.DONE fields have been updated by DMAC hardware. Additionally, the DMAC_CTRLAx.DONE bit is asserted when the buffer transfer has completed.

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.

18. 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 27-1 on page 329. The DMAC then knows that the previous buffer transferred was the last buffer in the DMAC transfer.

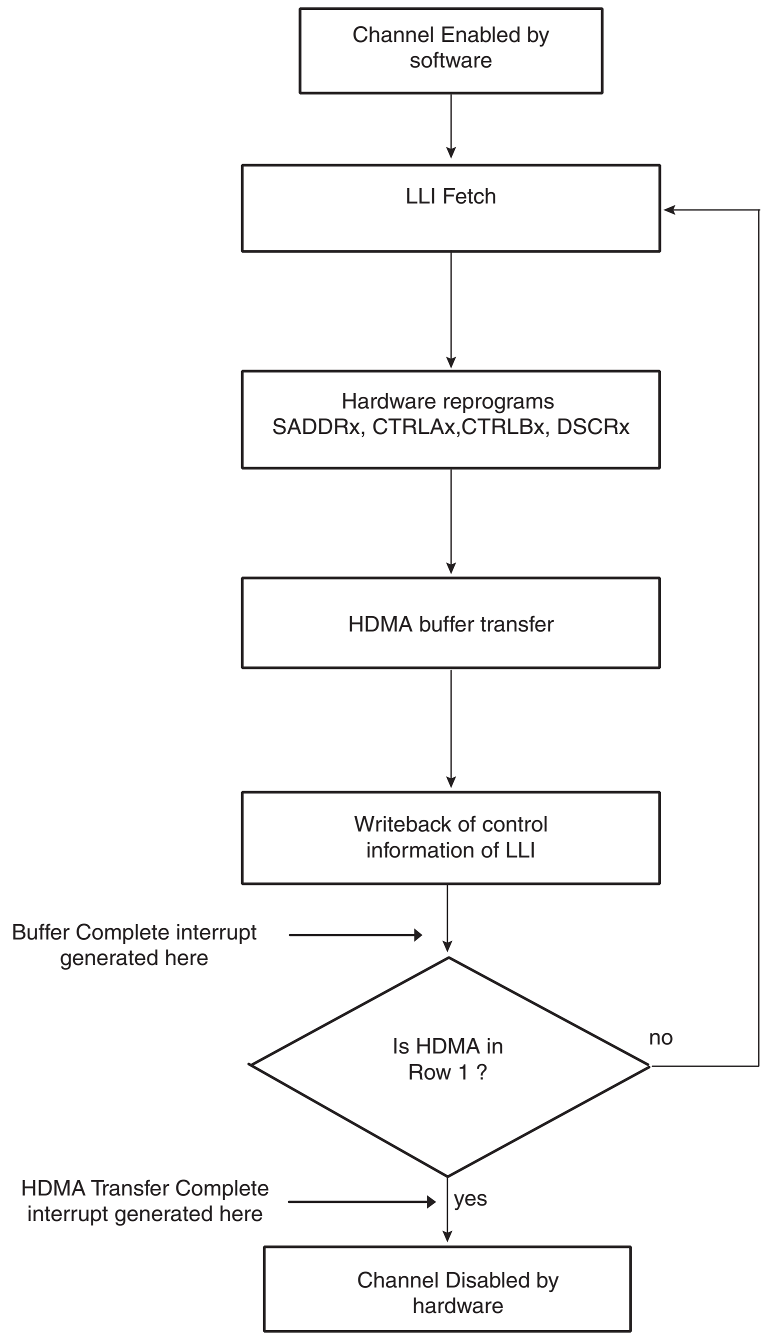
The DMAC transfer might look like that shown in Figure 27-15 on page 347 Note that the destination address is decrementing.

Figure 27-15. DMAC Transfer with Linked List Source Address and Contiguous Destination Address



The DMAC transfer flow is shown in Figure 27-16 on page 348.

Figure 27-16. DMAC Transfer Flow for Linked List Source Address and Contiguous Destination Address



27.3.6 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.

Note: If a channel is disabled by software, an active single or chunk transaction is not guaranteed to receive an acknowledgement.

27.3.6.1 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.

Note: If a channel is disabled by software, an active single or chunk transaction is not guaranteed to receive an acknowledgement.

27.4 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.
- When destination peripheral is defined as the flow controller, source single transfer request are not serviced until Destination Peripheral has asserted its Last Transfer Flag.
- When Source Peripheral is flow controller, destination single transfer request are not serviced until Source Peripheral has asserted its Last Transfer Flag.

- When destination peripheral is defined as the flow controller, if the destination width is smaller than the source width, then a data loss may occur, and the loss is equal to Source Single Transfer size in bytes- destination Single Transfer size in bytes.
- When a Memory to Peripheral transfer occurs if the destination peripheral is flow controller, then a prefetch operation is performed. It means that data are extracted from memory before any request from the peripheral is generated.
- 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 hardware handshaking interface protocol is fully implemented, a peripheral is expected to deassert any sreq or breq signals on receiving the ack signal irrespective of the request the ack was asserted in response to.
- Multiple Transfers involving the same peripheral must not be programmed and enabled on different channel, unless this peripheral integrates several hardware handshaking interface.
- When a Peripheral is flow controller, the targeted DMAC Channel must be enabled before the Peripheral. If you do not ensure this the DMAC Channel might miss a Last Transfer Flag, if the First DMAC request is also the last transfer.
- 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.

27.5 DMA Controller (DMAC) User Interface

Table 27-2. Register Mapping

Offset	Register	Name	Access	Reset
0x000	DMAC Global Configuration Register	DMAC_GCFG	Read-write	0x10
0x004	DMAC Enable Register	DMAC_EN	Read-write	0x0
0x008	DMAC Software Single Request Register	DMAC_SREQ	Read-write	0x0
0x00C	DMAC Software Chunk Transfer Request Register	DMAC_CREQ	Read-write	0x0
0x010	DMAC Software Last Transfer Flag Register	DMAC_LAST	Read-write	0x0
0x014	DMAC Request Synchronization Register	DMAC_SYNC	Read-write	0x0
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+ch_num*(0x28)+(0x0)	DMAC Channel Source Address Register	DMAC_SADDR	Read-write	0x0
0x03C+ch_num*(0x28)+(0x4)	DMAC Channel Destination Address Register	DMAC_DADDR	Read-write	0x0
0x03C+ch_num*(0x28)+(0x8)	DMAC Channel Descriptor Address Register	DMAC_DSCR	Read-write	0x0
0x03C+ch_num*(0x28)+(0xC)	DMAC Channel Control A Register	DMAC_CTRLA	Read-write	0x0
0x03C+ch_num*(0x28)+(0x10)	DMAC Channel Control B Register	DMAC_CTRLB	Read-write	0x0
0x03C+ch_num*(0x28)+(0x14)	DMAC Channel Configuration Register	DMAC_CFG	Read-write	0x01000000
0x03C+ch_num*(0x28)+(0x18)	DMAC Channel Source Picture in Picture Configuration Register	DMAC_SPIP	Read-write	0x0
0x03C+ch_num*(0x28)+(0x1C)	DMAC Channel Destination Picture in Picture Configuration Register	DMAC_DPIP	Read-write	0x0
0x03C+ch_num*(0x28)+(0x20)	Reserved	–	–	–
0x03C+ch_num*(0x28)+(0x24)	Reserved	–	–	–
0x064 - 0x100	DMAC Channel 1 to 3 Register(1)		Read-write	0x0
0x017C- 0x1FC	Reserved	–	–	–

Note: 1. The addresses for the DMAC registers shown here are for DMA Channel 0. This sequence of registers is repeated successively for each DMA channel located between 0x064 and 0x100.

27.5.1 DMAC Global Configuration Register

Name: DMAC_GCFG

Address: 0xFFFFEC00

Access: Read-write

Reset: 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.

27.5.2 DMAC Enable Register

Name: DMAC_EN

Address: 0xFFFFEC04

Access: Read-write

Reset: 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.

27.5.3 DMAC Software Single Request Register

Name: DMAC_SREQ

Address: 0xFFFFEC08

Access: Read-write

Reset: 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**
DSREQ3	SSREQ3	DSREQ2	SSREQ2	DSREQ1	SSREQ1	DSREQ0	SSREQ0

- **DSREQx**

Request a destination single transfer on channel i.

- **SSREQx**

Request a source single transfer on channel i.

27.5.4 DMAC Software Chunk Transfer Request Register

Name: DMAC_CREQ

Address: 0xFFFFEC0C

Access: Read-write

Reset: 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
DCREQ3	SCREQ3	DCREQ2	SCREQ2	DCREQ1	SCREQ1	DCREQ0	SCREQ0

- **DCREQx**

Request a destination chunk transfer on channel i.

- **SCREQx**

Request a source chunk transfer on channel i.

27.5.5 DMAC Software Last Transfer Flag Register

Name: DMAC_LAST

Address: 0xFFFFEC10

Access: Read-write

Reset: 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
DLAST3	SLAST3	DLAST2	SLAST2	DLAST1	SLAST1	DLAST0	SLAST0

- **DLASTx**

Writing one to DLASTx prior to writing one to DSREQx or DCREQx indicates that this destination request is the last transfer of the buffer.

- **SLASTx**

Writing one to SLASTx prior to writing one to SSREQx or SCREQx indicates that this source request is the last transfer of the buffer.

27.5.6 DMAC Request Synchronization Register

Name: DMAC_SYNC

Address: 0xFFFFEC14

Access: Read-write

Reset: 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
SYR7	SYR6	SYR5	SYR4	SYR3	SYR2	SYR1	SYR0

- **SYR[7:0]**

Request Synchronizer Register. Write one to DMAC_SYNCx to synchronize peripheral i request lines.

27.5.7 DMAC Error, Buffer Transfer and Chained Buffer Transfer Interrupt Enable Register

Name: DMAC_EBCIER

Address: 0xFFFFEC18

Access: Write-only

Reset: 0x00000000

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

23	22	21	20	19	18	17	16
–	–	–	–	ERR3	ERR2	ERR1	ERR0

15	14	13	12	11	10	9	8
–	–	–	–	CBTC3	CBTC2	CBTC1	CBTC0

7	6	5	4	3	2	1	0
–	–	–	–	BTC3	BTC2	BTC1	BTC0

- **BTC[3:0]**

Buffer Transfer Completed Interrupt Enable Register. Set the relevant bit in the BTC field to enable the interrupt for channel i.

- **CBTC[3:0]**

Chained Buffer Transfer Completed Interrupt Enable Register. Set the relevant bit in the CBTC field to enable the interrupt for channel i.

- **ERR[3:0]**

Access Error Interrupt Enable Register. Set the relevant bit in the ERR field to enable the interrupt for channel i.

27.5.8 DMAC Error, Buffer Transfer and Chained Buffer Transfer Interrupt Disable Register

Name: DMAC_EBCIDR

Address: 0xFFFFEC1C

Access: Write-only

Reset: 0x00000000

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

23	22	21	20	19	18	17	16
–	–	–	–	ERR3	ERR2	ERR1	ERR0

15	14	13	12	11	10	9	8
–	–	–	–	CBTC3	CBTC2	CBTC1	CBTC0

7	6	5	4	3	2	1	0
–	–	–	–	BTC3	BTC2	BTC1	BTC0

- **BTC[3:0]**

Buffer transfer completed Disable Interrupt Register. When set, a bit of the BTC field disables the interrupt from the relevant DMAC channel.

- **CBTC[3:0]**

Chained Buffer transfer completed Disable Register. When set, a bit of the CBTC field disables the interrupt from the relevant DMAC channel.

- **ERR[3:0]**

Access Error Interrupt Disable Register. When set, a bit of the ERR field disables the interrupt from the relevant DMAC channel.

27.5.9 DMAC Error, Buffer Transfer and Chained Buffer Transfer Interrupt Mask Register

Name: DMAC_EBCIMR

Address: 0xFFFFEC20

Access: Read-only

Reset: 0x00000000

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

23	22	21	20	19	18	17	16
–	–	–	–	ERR3	ERR2	ERR1	ERR0

15	14	13	12	11	10	9	8
–	–	–	–	CBTC3	CBTC2	CBTC1	CBTC0

7	6	5	4	3	2	1	0
–	–	–	–	BTC3	BTC2	BTC1	BTC0

- **BTC[3:0]**

0: Buffer Transfer completed interrupt is disabled for channel i.

1: Buffer Transfer completed interrupt is enabled for channel i.

- **CBTC[3:0]**

0: Chained Buffer Transfer interrupt is disabled for channel i.

1: Chained Buffer Transfer interrupt is enabled for channel i.

- **ERR[3:0]**

0: Transfer Error Interrupt is disabled for channel i.

1: Transfer Error Interrupt is enabled for channel i.

27.5.10 DMAC Error, Buffer Transfer and Chained Buffer Transfer Status Register

Name: DMAC_EBCISR

Address: 0xFFFFEC24

Access: Read-only

Reset: 0x00000000

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

23	22	21	20	19	18	17	16
–	–	–	–	ERR3	ERR2	ERR1	ERR0

15	14	13	12	11	10	9	8
–	–	–	–	CBTC3	CBTC2	CBTC1	CBTC0

7	6	5	4	3	2	1	0
–	–	–	–	BTC3	BTC2	BTC1	BTC0

- **BTC[3:0]**

When BTC[*i*] is set, Channel *i* buffer transfer has terminated.

- **CBTC[3:0]**

When CBTC[*i*] is set, Channel *i* Chained buffer has terminated. LLI Fetch operation is disabled.

- **ERR[3:0]**

When ERR[*i*] is set, Channel *i* has detected an AHB Read or Write Error Access.

27.5.11 DMAC Channel Handler Enable Register

Name: DMAC_CHER

Address: 0xFFFFEC28

Access: Write-only

Reset: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	KEEP3	KEEP2	KEEP1	KEEP0

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

15	14	13	12	11	10	9	8
–	–	–	–	SUSP3	SUSP2	SUSP1	SUSP0

7	6	5	4	3	2	1	0
–	–	–	–	ENA3	ENA2	ENA1	ENA0

- **ENA[3:0]**

When set, a bit of the ENA field enables the relevant channel.

- **SUSP[3:0]**

When set, a bit of the SUSPfield freezes the relevant channel and its current context.

- **KEEP[3:0]**

When set, a bit of the KEEP field resumes the current channel from an automatic stall state.

27.5.12 DMAC Channel Handler Disable Register

Name: DMAC_CHDR

Address: 0xFFFFEC2C

Access: Write-only

Reset: 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
–	–	–	–	RES3	RES2	RES1	RES0

7	6	5	4	3	2	1	0
–	–	–	–	DIS3	DIS2	DIS1	DIS0

- **DIS[3:0]**

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[3:0] field in the DMAC_CHSR register to be sure that the channel is disabled.

- **RES[3:0]**

Write one to this field to resume the channel transfer restoring its context.

27.5.13 DMAC Channel Handler Status Register

Name: DMAC_CHSR

Address: 0xFFFFEC30

Access: Read-only

Reset: 0x00FF0000

31	30	29	28	27	26	25	24
–	–	–	–	STAL3	STAL2	STAL1	STAL0

23	22	21	20	19	18	17	16
–	–	–	–	EMPT3	EMPT2	EMPT1	EMPT0

15	14	13	12	11	10	9	8
–	–	–	–	SUSP3	SUSP2	SUSP1	SUSP0

7	6	5	4	3	2	1	0
–	–	–	–	ENA3	ENA2	ENA1	ENA0

- **ENA[3:0]**

A one in any position of this field indicates that the relevant channel is enabled.

- **SUSP[3:0]**

A one in any position of this field indicates that the channel transfer is suspended.

- **EMPT[3:0]**

A one in any position of this field indicates that the relevant channel is empty.

- **STAL[3:0]**

A one in any position of this field indicates that the relevant channel is stalling.

27.5.14 DMAC Channel x [x = 0..3] Source Address Register

Name: DMAC_SADDRx [x = 0..3]

Addresses: 0xFFFFEC3C [0], 0xFFFFEC64 [1], 0xFFFFEC8C [2], 0xFFFFECB4 [3]

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
SADDRx							

23	22	21	20	19	18	17	16
SADDRx							

15	14	13	12	11	10	9	8
SADDRx							

7	6	5	4	3	2	1	0
SADDRx							

- **SADDRx**

Channel x source address. This register must be aligned with the source transfer width.

27.5.15 DMAC Channel x [x = 0..3] Destination Address Register

Name: DMAC_DADDRx [x = 0..3]

Addresses: 0xFFFFEC40 [0], 0xFFFFEC68 [1], 0xFFFFEC90 [2], 0xFFFFECB8 [3]

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
DADDRx							

23	22	21	20	19	18	17	16
DADDRx							

15	14	13	12	11	10	9	8
DADDRx							

7	6	5	4	3	2	1	0
DADDRx							

- **DADDRx**

Channel x destination address. This register must be aligned with the destination transfer width.

27.5.16 DMAC Channel x [x = 0..3] Descriptor Address Register

Name: DMAC_DSCRx [x = 0..3]

Addresses: 0xFFFFEC44 [0], 0xFFFFEC6C [1], 0xFFFFEC94 [2], 0xFFFFECBC [3]

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
DSCRx							

23	22	21	20	19	18	17	16
DSCRx							

15	14	13	12	11	10	9	8
DSCRx							

7	6	5	4	3	2	1	0
DSCRx						DSCRx_IF	

- **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.

27.5.17 DMAC Channel x [x = 0..3] Control A Register

Name: DMAC_CTRLAx [x = 0..3]

Addresses: 0xFFFFEC48 [0], 0xFFFFEC70 [1], 0xFFFFEC98 [2], 0xFFFFECC0 [3]

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
DONE	–	DST_WIDTH		–	–	SRC_WIDTH	

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

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.

- **SCSIZE**

Source Chunk Transfer Size.

SCSIZE value	Number of data transferred
000	1
001	4
010	8
011	16
100	32
101	64
110	128
111	256

- **DCSIZE**

Destination Chunk Transfer size.

DCSIZE	Number of data transferred
000	1
001	4
010	8
011	16

DCSIZE	Number of data transferred
100	32
101	64
110	128
111	256

- **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.

27.5.18 DMAC Channel x [x = 0..3] Control B Register

Name: DMAC_CTRLBx [x = 0..3]

Addresses: 0xFFFFEC4C [0], 0xFFFFEC74 [1], 0xFFFFEC9C [2], 0xFFFFECC4 [3]

Access: Read-write

Reset: 0x00000000

31	30	29	28	27	26	25	24
AUTO	IEN	DST_INCR		–	–	SRC_INCR	

23	22	21	20	19	18	17	16
FC			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.

- **FC**

This field defines which device controls the size of the buffer transfer, also referred as to the Flow Controller.

FC	Type of transfer	Flow Controller
000	Memory-to-Memory	DMA Controller
001	Memory-to-Peripheral	DMA Controller
010	Peripheral-to-Memory	DMA Controller
011	Peripheral-to-Peripheral	DMA Controller
100	Peripheral-to-Memory	Peripheral
101	Memory-to-Peripheral	Peripheral
110	Peripheral-to-Peripheral	Source Peripheral
111	Peripheral-to-Peripheral	Destination Peripheral

- **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

- **IEN**

If this bit is cleared, when the buffer transfer is completed, the BTC[x] flag is set in the EBCISR status register. This bit is active low.

- **AUTO**

Automatic multiple buffer transfer is enabled. When set, this bit enables replay mode or contiguous mode when several buffers are transferred.

27.5.19 DMAC Channel x [x = 0..3] Configuration Register

Name: DMAC_CFGx [x = 0..3]

Addresses: 0xFFFFEC50 [0], 0xFFFFEC78 [1], 0xFFFFECA0 [2], 0xFFFFECC8 [3]

Access: Read-write

Reset: 0x0100000000

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

23	22	21	20	19	18	17	16
–	LOCK_IF_L	LOCK_B	LOCK_IF	–	–	–	SOD

15	14	13	12	11	10	9	8
–	–	DST_H2SEL	DST_REP	–	–	SRC_H2SEL	SRC_REP

7	6	5	4	3	2	1	0
DST_PER				SRC_PER			

- **SRC_PER**

Channel x Source Request is associated with peripheral identifier coded SRC_PER handshaking interface.

- **DST_PER**

Channel x Destination Request is associated with peripheral identifier coded DST_PER handshaking interface.

- **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.

- **SRC_H2SEL**

0: Software handshaking interface is used to trigger a transfer request.

1: Hardware handshaking interface is used to trigger a transfer request.

- **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.

- **DST_H2SEL**

0: Software handshaking interface is used to trigger a transfer request.

1: Hardware handshaking interface is used to trigger a transfer request.

- **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.

- **LOCK_IF**

0: Interface Lock capability is disabled

1: Interface Lock capability is enabled

- **LOCK_B**

0: AHB Bus Locking capability is disabled.

1: AHB Bus Locking capability is enabled.

- **LOCK_IF_L**

0: The Master Interface Arbiter is locked by the channel x for a chunk transfer.

1: The Master Interface Arbiter is locked by the channel x for a buffer transfer.

- **AHB_PROT**

AHB_PROT field provides additional information about a bus access and is primarily used to implement some level of protection.

HPROT[3]	HPROT[2]	HPROT[1]	HPROT[0]	Description
			1	Data access
		AHB_PROT[0]		0: User Access 1: Privileged Access
	AHB_PROT[1]			0: Not Bufferable 1: Bufferable
AHB_PROT[2]				0: Not cacheable 1: Cacheable

- **FIFOCFG**

FIFOCFG	FIFO request
00	The largest defined length AHB burst is performed on the destination AHB interface.
01	When half FIFO size is available/filled, a source/destination request is serviced.
10	When there is enough space/data available to perform a single AHB access, then the request is serviced.

27.5.20 DMAC Channel x [x = 0..3] Source Picture in Picture Configuration Register

Name: DMAC_SPIPx [x = 0..3]

Addresses: 0xFFFFEC54 [0], 0xFFFFEC7C [1], 0xFFFFECA4 [2], 0xFFFFECCC [3]

Access: Read-write

Reset: 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.

27.5.21 DMAC Channel x [x = 0..3] Destination Picture in Picture Configuration Register

Name: DMAC_DPIPx [x = 0..3]

Addresses: 0xFFFFEC58 [0], 0xFFFFEC80 [1], 0xFFFFECA8 [2], 0xFFFFECD0 [3]

Access: Read-write

Reset: 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
DPIP_HOLE							

7	6	5	4	3	2	1	0
DPIP_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.

28. Peripheral DMA Controller (PDC)

28.1 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 24 channels. The full-duplex peripherals feature 22 mono directional channels used in pairs (transmit only or receive only). The half-duplex peripherals feature 2 bi-directional channels.

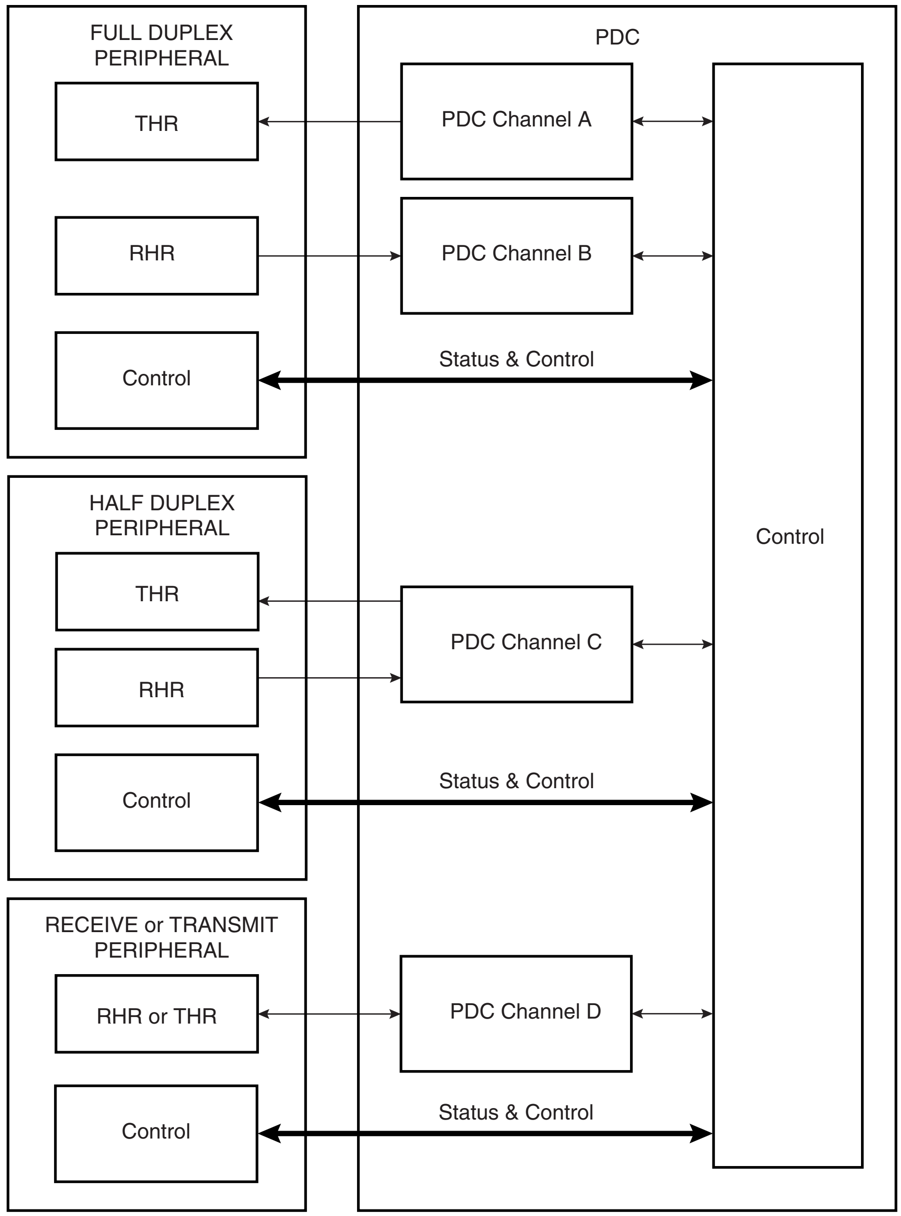
The user interface of each PDC channel is integrated into the user interface of the peripheral it serves. The user interface of mono directional channels (receive only or transmit only), contains two 32-bit memory pointers and two 16-bit counters, one set (pointer, counter) for current transfer and one set (pointer, counter) for next transfer. The bi-directional channel user interface contains four 32-bit memory pointers and four 16-bit counters. Each set (pointer, counter) is used by current transmit, next transmit, current receive and next receive.

Using the PDC removes processor overhead by reducing its intervention during the transfer. This significantly reduces the number of clock cycles required for a data transfer, which improves microcontroller performance.

To launch a transfer, the peripheral triggers its associated PDC channels by using transmit and receive signals. When the programmed data is transferred, an end of transfer interrupt is generated by the peripheral itself.

28.2 Block Diagram

Figure 28-1. Block Diagram



28.3 Functional Description

28.3.1 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 1.3.3 and to the associated peripheral user interface.

28.3.2 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.

28.3.3 Transfer Counters Access

Each channel has two 16-bit counters, one for current transfer and the other one for next transfer. These counters define the size of data to be transferred by the channel. The current transfer counter is decremented first as the data addressed by current memory pointer starts to be transferred. When the current transfer counter reaches zero, the channel checks its next transfer counter. If the value of next counter is zero, the channel stops transferring data and sets the appropriate flag. But if the next counter value is greater then zero, the values of the next pointer/next counter are copied into the current pointer/current counter and the channel resumes the transfer whereas next pointer/next counter get zero/zero as values. At the end of this transfer the PDC channel sets the appropriate flags in the Peripheral Status Register.

The following list gives an overview of how status register flags behave depending on the counters' values:

- ENDRX flag is set when the PERIPH_RCR register reaches zero.
- RXBUFF flag is set when both PERIPH_RCR and PERIPH_RNCR reach zero.
- ENDTX flag is set when the PERIPH_TCR register reaches zero.
- TXBUFE flag is set when both PERIPH_TCR and PERIPH_TNCR reach zero.

These status flags are described in the Peripheral Status Register.

28.3.4 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.

28.3.5 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.

28.3.5.1 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.

28.3.5.2 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.

28.3.5.3 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.

28.3.5.4 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.

28.4 Peripheral DMA Controller (PDC) User Interface

Table 28-1. Register Mapping

Offset	Register	Name	Access	Reset
0x100	Receive Pointer Register	PERIPH[(1)]_RPR	Read-write	0
0x104	Receive Counter Register	PERIPH_RCR	Read-write	0
0x108	Transmit Pointer Register	PERIPH_TPR	Read-write	0
0x10C	Transmit Counter Register	PERIPH_TCR	Read-write	0
0x110	Receive Next Pointer Register	PERIPH_RNPR	Read-write	0
0x114	Receive Next Counter Register	PERIPH_RNCR	Read-write	0
0x118	Transmit Next Pointer Register	PERIPH_TNPR	Read-write	0
0x11C	Transmit Next Counter Register	PERIPH_TNCR	Read-write	0
0x120	Transfer Control Register	PERIPH_PTCR	Write-only	0
0x124	Transfer Status Register	PERIPH_PTSR	Read-only	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.)

28.4.1 Receive Pointer Register

Name: PERIPH_RPR

Access: 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.

28.4.2 Receive Counter Register

Name: PERIPH_RCR

Access: 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

28.4.3 Transmit Pointer Register

Name: PERIPH_TPR

Access: 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.

28.4.4 Transmit Counter Register

Name: PERIPH_TCR

Access: 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

28.4.5 Receive Next Pointer Register

Name: PERIPH_RNPR

Access: 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.

28.4.6 Receive Next Counter Register

Name: PERIPH_RNCR

Access: 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.

28.4.7 Transmit Next Pointer Register

Name: PERIPH_TNPR

Access: 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.

28.4.8 Transmit Next Counter Register

Name: PERIPH_TNCR

Access: 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.

28.4.9 Transfer Control Register

Name: PERIPH_PTCR

Access: Write-only

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

28.4.10 Transfer Status Register

Name: PERIPH_PTSR

Access: Read-only

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

29. Clock Generator

29.1 Overview

The Clock Generator is made up of 2 programmable PLLs, a UTMI PLL, a Main Oscillator, as well as an RC Oscillator and a 32,768 Hz low-power Oscillator.

It provides the following clocks:

- SLCK, the Slow Clock, which is the only permanent clock within the system
- MAINCK is the output of the Main Oscillator

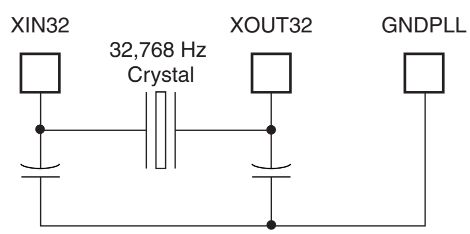
The Clock Generator User Interface is embedded within the Power Management Controller one and is described in Section 30.9. However, the Clock Generator registers are named CKGR_.

- PLLACK is the output of the Divider and PLL A block
- PLLBCK is the output of the Divider and PLL B block

29.2 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 29-1.

Figure 29-1. Typical Slow Clock Crystal Oscillator Connection



29.3 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.

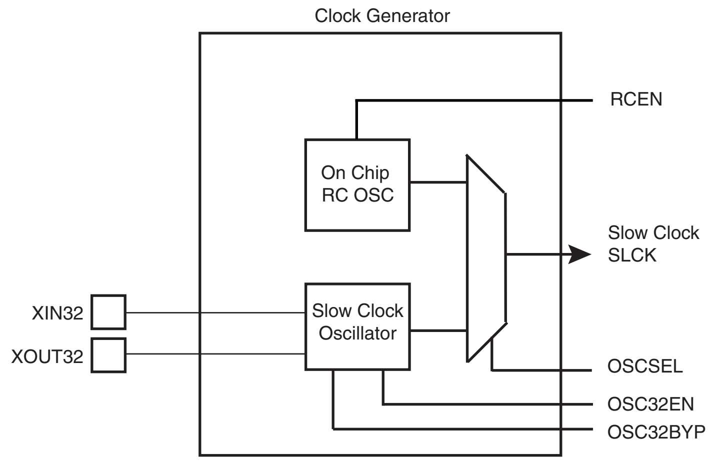
29.4 Slow Clock Selection

The AT91CAP9 slow clock can be generated either by an external 32768Hz crystal or the on-chip RC oscillator. The 32768Hz crystal oscillator can be bypassed, by setting the bit OSC32BYP, to accept an external slow clock on XIN32.

The internal RC oscillator and the 32768Hz oscillator can be enabled by setting to 1 respectively RCEN bit and OSC32EN bit in the system controller user interface. OSCSEL command selects the slow clock source.

By default the AT91CAP9S500A/AT91CAP9S250A slow clock source is the internal RC oscillator. System startup time is 4 slow clock periods, typically 125 µs.

Figure 29-2. 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 by software the slow clock switching and so must take precautions during the switching phase.

29.4.1 Switching from Internal RC Oscillator to the 32768 Hz Crystal

To switch from internal RC oscillator to the 32768 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 32768 Hz oscillator by setting the bit OSC32EN to 1.
- Wait 32768 Hz Startup Time for clock stabilization (software loop)
- Switch from internal RC to 32768 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.

29.4.2 Bypassing the 32768 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.

29.4.3 Switching from 32768 Hz Crystal to the Internal RC Oscillator

The same procedure must be followed to switch from 32768 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

29.4.4 Slow Clock Configuration Register

Name: SCKCR

Address: 0xFFFFFD50

Access: Read-write

Reset: 0x0000_0001

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: 32768Hz 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.

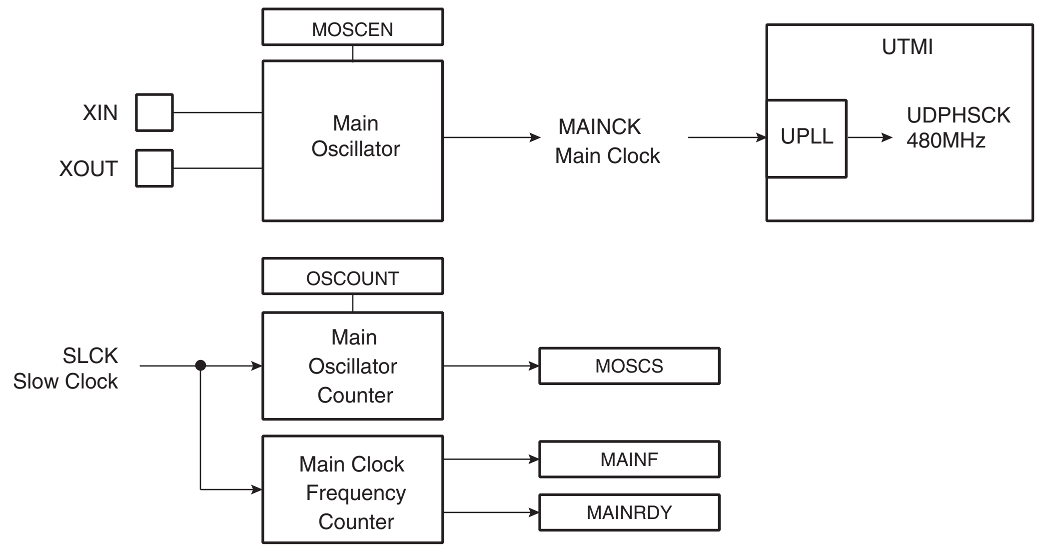
1: Slow clock is 32768Hz oscillator.

29.5 Main Oscillator

The Main Oscillator is designed for a 12 MHz fundamental crystal but supports 8 to 16 MHz crystals. The 12 MHz is also used to generate the 480 MHz USB High Speed Clock (UDPHSCK) thanks to the UTMI PLL (UPLL).

Figure 29-3 shows the Main Oscillator block diagram.

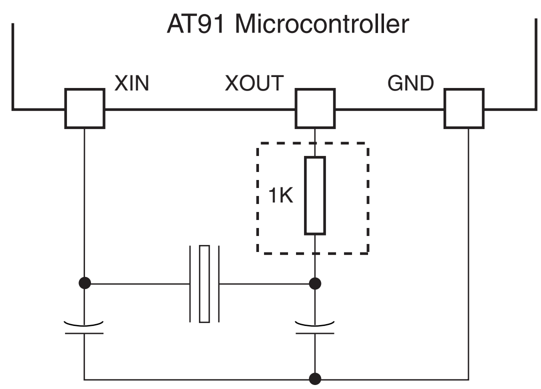
Figure 29-3. Main Oscillator Block Diagram



29.5.1 Main Oscillator Connections

The Clock Generator integrates a Main Oscillator that is designed for a 8 to 16 MHz fundamental crystal. The typical crystal connection is illustrated in Figure 29-4. For further details on the electrical characteristics of the Main Oscillator, see the section "DC Characteristics" of the product datasheet.

Figure 29-4. Typical Crystal Connection



29.5.2 Main Oscillator Startup Time

The startup time of the Main Oscillator is given in the DC Characteristics section of the product datasheet. The startup time depends on the crystal frequency and decreases when the frequency rises.

29.5.3 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.

29.5.4 Main Clock Frequency Counter

The Main Oscillator features a Main Clock frequency counter that provides the quartz frequency connected to the Main Oscillator. Generally, this value is known by the system designer; however, it is useful for the boot program to configure the device with the correct clock speed, independently of the application.

The Main Clock frequency counter starts incrementing at the Main Clock speed after the next rising edge of the Slow Clock as soon as the Main Oscillator is stable, i.e., as soon as the MOSCS bit is set. Then, at the 16th falling edge of Slow Clock, the MAINRDY bit in CKGR_MCFR (Main Clock Frequency Register) is set and the counter stops counting. Its value can be read in the MAINF field of CKGR_MCFR and gives the number of Main Clock cycles during 16 periods of Slow Clock, so that the frequency of the crystal connected on the Main Oscillator can be determined.

29.5.5 Main Oscillator Bypass

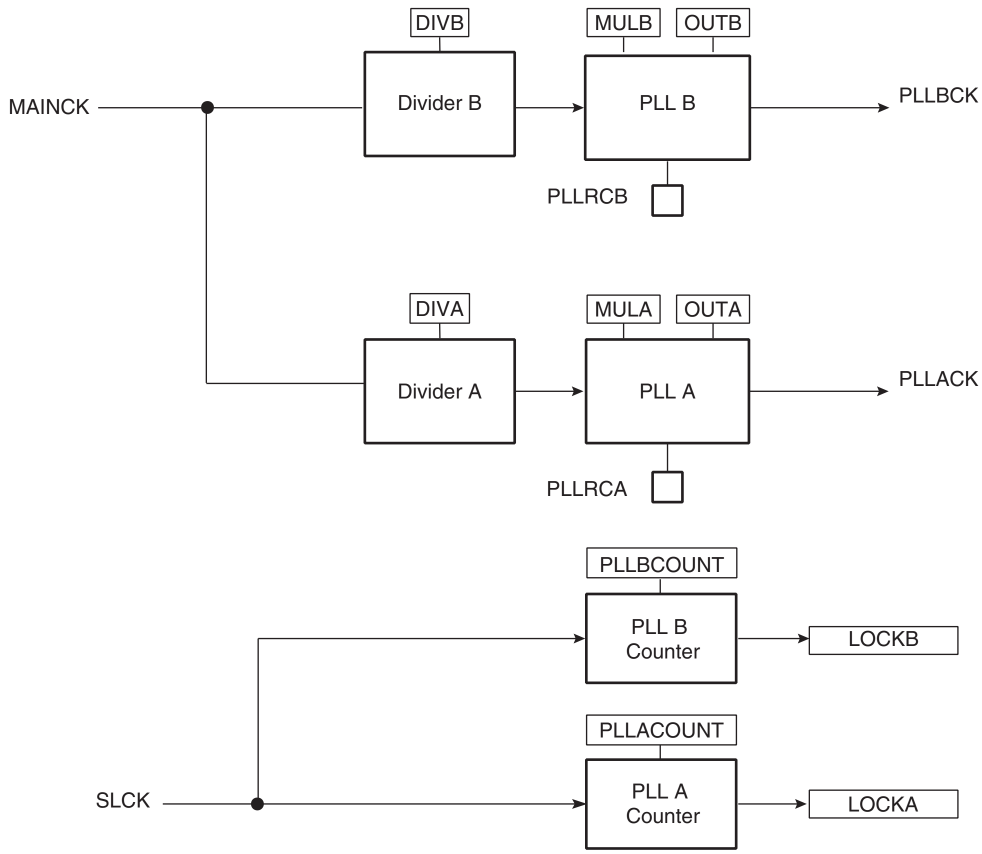
The user can input a clock on the device instead of connecting a crystal. In this case, the user has to provide the external clock signal on the XIN pin. The input characteristics of the XIN pin under these conditions are given in the product electrical characteristics section. The programmer has to be sure to set the OSCBYPASS bit to 1 and the MOSCEN bit to 0 in the Main OSC register (CKGR_MOR) for the external clock to operate properly.

29.6 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 29-5 shows the block diagram of the divider and PLL blocks.

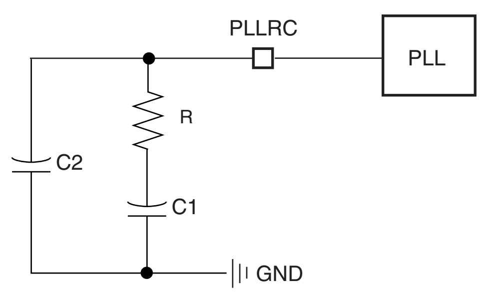
Figure 29-5. Divider and PLL Block Diagram



29.6.1 PLL Filter

The PLL requires connection to an external second-order filter through the PLLRCA and/or PLLRCB pin. Figure 29-6 shows a schematic of these filters.

Figure 29-6. 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.

29.6.2 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.

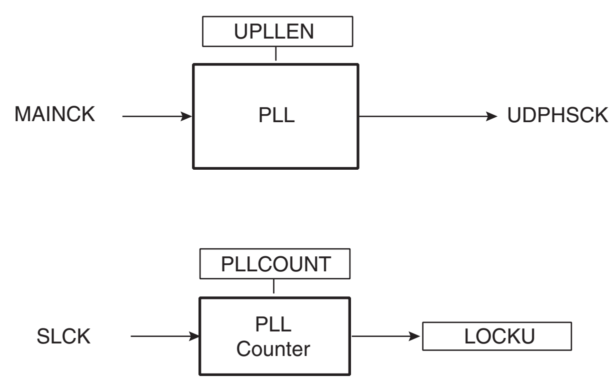
The PLL allows multiplication of the divider's outputs. The PLL clock signal has a frequency that depends on the respective source signal frequency and on the parameters DIV and MUL. The factor applied to the source signal frequency is (MUL + 1)/DIV. When MUL is written to 0, the corresponding PLL is 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 (LOCKA or LOCKB) in PMC_SR is automatically cleared. The values written in the PLLCOUNT field (PLLA-COUNT or PLLBCOUNT) in CKGR_PLLR (CKGR_PLLAR or CKGR_PLLBR), are loaded in the PLL counter. The PLL counter then decrements at the speed of the Slow Clock until it reaches 0. At this time, the LOCK bit is set in PMC_SR and can trigger an interrupt to the processor. The user has to load the number of Slow Clock cycles required to cover the PLL transient time into the PLLCOUNT field. The transient time depends on the PLL filter. The initial state of the PLL and its target frequency can be calculated using a specific tool provided by Atmel.

29.6.3 UTMI Phase Lock Loop Programming

The multiplier is hard-wired to 40 to obtain the USB High Speed 480 MHz.

Figure 29-7. UTMI PLL



Whenever the PLL is enabled by writing UPLLEN 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.

30. Power Management Controller (PMC)

30.1 Overview

The Power Management Controller (PMC) optimizes power consumption by controlling all system and user peripheral clocks. The PMC enables/disables the clock inputs to many of the peripherals and the ARM Processor.

The Power Management Controller provides the following clocks:

- MCK, the Master Clock, programmable from a few hundred Hz to the maximum operating frequency of the device. It is available to the modules running permanently, such as the AIC and the Memory Controller.
- Processor Clock (PCK), switched off when entering processor in idle mode.
- Application Clock (APCK), programmable from a few hundred Hz to the maximum operating frequency of the device, switched off when entering processor idle mode.
- DDRCK, the Double Data Rate Clock, provided to peripherals using double data rate transfer (e.g., DDR SDR Controller)
- 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.
- UHP Clock (UHPCK), required by USB Host Port operations.
- Programmable Clock Outputs can be selected from the clocks provided by the clock generator and driven on the PCKx pins.

30.2 Master Clock Controller

The Master Clock Controller provides selection and division of the Master Clock (MCK). MCK is the clock provided to all the peripherals and the memory controller.

The Master Clock is selected from one of the clocks provided by the Clock Generator. Selecting the Slow Clock provides a Slow Clock signal to the whole device. Selecting the Main Clock saves power consumption of the PLLs.

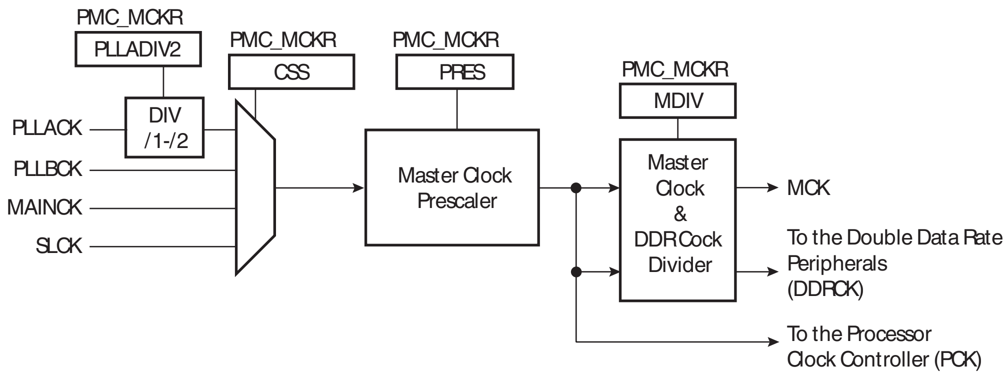
The Master Clock Controller is made up of a clock selector and a prescaler. It also contains a Master Clock divider which allows the processor clock to be faster than the Master Clock.

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.

A Double Data Rate Clock (DDRCK) is created after the clock selector and before the clock prescaler. The software must ensure that the DDRCK clock rate is twice the MCK clock rate.

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 30-1. Master Clock Controller



30.3 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 processor specific “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.

Note: The ARM “Wait for Interrupt” mode is enterred with CP15 coprocessor operation.

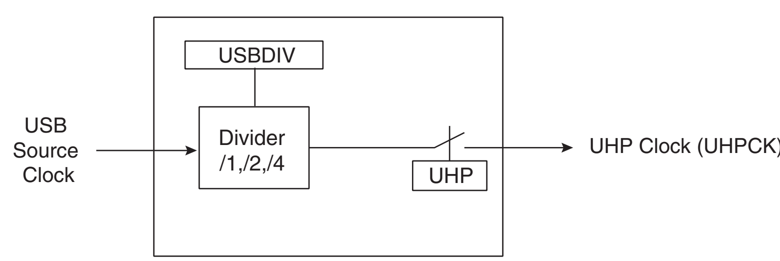
30.4 USB Clock Controller

The USB Source Clock is always generated from the PLL B output. If using the USB, the user must program the PLL to generate a 48 MHz, a 96 MHz or a 192 MHz signal with an accuracy of ± 0.25% depending on the USBDIV bit in CKGR_PLLBR (see Figure 29-5).

When the PLL B output is stable, i.e., the LOCKB is set:

- The USB host clock can be enabled by setting the UHP bit in PMC_SCER. To save power on this peripheral when it is not used, the user can set the UHP bit in PMC_SCDR. The UHP bit in PMC_SCSR gives the activity of this clock. The USB host port require both the 12/48 MHz signal and the Master Clock. The Master Clock may be controlled via the Master Clock Controller.

Figure 30-2. USB Clock Controller



30.5 Peripheral Clock Controller

The Power Management Controller controls the clocks of each embedded peripheral by the way of the Peripheral Clock Controller. The user can individually enable and disable the Master Clock on the peripherals by writing into the Peripheral Clock Enable (PMC_PCER) and Peripheral Clock Disable (PMC_PCDR) registers. The status of the peripheral clock activity can be read in the Peripheral Clock Status Register (PMC_PCSR).

When a peripheral clock is disabled, the clock is immediately stopped. The peripheral clocks are automatically disabled after a reset.

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

The bit number within the Peripheral Clock Control registers (PMC_PCER, PMC_PCDR, and PMC_PCSR) is the Peripheral Identifier defined at the product level. Generally, the bit number corresponds to the interrupt source number assigned to the peripheral.

30.6 Programmable Clock Output Controller

The PMC controls 4 signals to be output on external pins PCKx. Each signal can be independently programmed via the PMC_PCKx registers.

PCKx can be independently selected between the Slow clock, the PLL A output, the PLL B output and the main clock by writing the CSS field in PMC_PCKx. Each output signal can also be divided by a power of 2 between 1 and 64 by writing the PRES (Prescaler) field in PMC_PCKx.

Each output signal can be enabled and disabled by writing 1 in the corresponding bit, PCKx of PMC_SCER and PMC_SCDR, respectively. Status of the active programmable output clocks are given in the PCKx bits of PMC_SCSR (System Clock Status Register).

Moreover, like the PCK, a status 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.

30.7 Programming Sequence

1. Enabling the Main Oscillator:

   The main oscillator is enabled by setting the MOSCEN field in the CKGR_MOR register. In some cases it may be advantageous to define a start-up time. This can be achieved by writing a value in the OSCOUNT field in the CKGR_MOR register.

   Once this register has been correctly configured, the user must wait for MOSCS field in the PMC_SR register to be set. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to MOSCS has been enabled in the PMC_IER register.

   Code Example:

   ```
   write_register(CKGR_MOR,0x00000701)
   ```

   Start Up Time = 8 * OSCOUNT / SLCK = 56 Slow Clock Cycles.

   So, the main oscillator will be enabled (MOSCS bit set) after 56 Slow Clock Cycles.

2. Checking the Main Oscillator Frequency (Optional):

   In some situations the user may need an accurate measure of the main oscillator frequency. This measure can be accomplished via the CKGR_MCFR register.

   Once the MAINRDY field is set in CKGR_MCFR register, the user may read the MAINF field in CKGR_MCFR register. This provides the number of main clock cycles within sixteen slow clock cycles.

3. Setting PLL A and divider A:

   All parameters necessary to configure PLL A and divider A are located in the CKGR_PLLAR register.

   It is important to note that Bit 29 must always be set to 1 when programming the CKGR_PLLAR register.

   The DIVA field is used to control the divider A itself. The user can program a value between 0 and 255. Divider A output is divider A input divided by DIVA. By default, DIVA parameter is set to 0 which means that divider A is turned off.

   The OUTA field is used to select the PLL A output frequency range.

   The MULA field is the PLL A multiplier factor. This parameter can be programmed between 0 and 2047. If MULA is set to 0, PLL A will be turned off. Otherwise PLL A output frequency is PLL A input frequency multiplied by (MULA + 1).

   The PLLACOUNT field specifies the number of slow clock cycles before LOCKA bit is set in the PMC_SR register after CKGR_PLLAR register has been written.

   Once CKGR_PLLAR register has been written, the user is obliged to wait for the LOCKA bit to be set in the PMC_SR register. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to LOCKA has been enabled in the PMC_IER register.

   All parameters in CKGR_PLLAR can be programmed in a single write operation. If at some stage one of the following parameters, SRCA, MULA, DIVA is modified, LOCKA bit will go low to indicate that PLL A is not ready yet. When PLL A is locked, LOCKA will be set again. User has to wait for LOCKA bit to be set before using the PLL A output clock.

   Code Example:

   ```
   write_register(CKGR_PLLAR,0x20030605)
   ```

   PLL A and divider A are enabled. PLL A input clock is main clock divided by 5. PLL An output clock is PLL A input clock multiplied by 4. Once CKGR_PLLAR has been written, LOCKA bit will be set after six slow clock cycles.

4. Setting PLL B and divider B:

   All parameters needed to configure PLL B and divider B are located in the CKGR_PLLBR register.

   The DIVB field is used to control divider B itself. A value between 0 and 255 can be programmed. Divider B output is divider B input divided by DIVB parameter. By default DIVB parameter is set to 0 which means that divider B is turned off.

   The OUTB field is used to select the PLL B output frequency range.

The MULB field is the PLL B multiplier factor. This parameter can be programmed between 0 and 2047. If MULB is set to 0, PLL B will be turned off, otherwise the PLL B output frequency is PLL B input frequency multiplied by (MULB + 1).

The PLLBCOUNT field specifies the number of slow clock cycles before LOCKB bit is set in the PMC_SR register after CKGR_PLLBR register has been written.

Once the PMC_PLLB register has been written, the user must wait for the LOCKB bit to be set in the PMC_SR register. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to LOCKB has been enabled in the PMC_IER register. All parameters in CKGR_PLLBR can be programmed in a single write operation. If at some stage one of the following parameters, MULB, DIVB is modified, LOCKB bit will go low to indicate that PLL B is not ready yet. When PLL B is locked, LOCKB will be set again. The user is constrained to wait for LOCKB bit to be set before using the PLL A output clock.

The USBDIV field is used to control the additional divider by 1, 2 or 4, which generates the USB clock(s).

Code Example:

```
write_register(CKGR_PLLBR,0x00040805)
```

If PLL B and divider B are enabled, the PLL B input clock is the main clock. PLL B output clock is PLL B input clock multiplied by 5. Once CKGR_PLLBR has been written, LOCKB bit will be set after eight slow clock cycles.

5. Selection of Master Clock and Processor Clock

The Master Clock and the Processor Clock are configurable via the PMC_MCKR register.

The CSS field is used to select the Master Clock divider source. By default, the selected clock source is slow clock.

The PRES field is used to control the Master Clock prescaler. The user can choose between different values (1, 2, 4, 8, 16, 32, 64). Master Clock output is prescaler input divided by PRES parameter. By default, PRES parameter is set to 1 which means that master clock is equal to slow clock.

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 (CKGR_PLLAR or CKGR_PLLBR), the MCKRDY flag will go low while PLL is unlocked. Once PLL is locked again, LOCK (LOCKA or LOCKB) goes high and MCKRDY is set. While PLLA is unlocked, the Master Clock selection is automatically changed to Slow Clock. While PLLB is unlocked, the Master Clock selection is automatically changed to Main Clock. For further information, see Section 30.8.2. “Clock Switching Waveforms” on page 402.

Code Example:

```
write_register(PMC_MCKR,0x00000001)
wait (MCKRDY=1)
write_register(PMC_MCKR,0x00000011)
wait (MCKRDY=1)
```

The Master Clock is main clock divided by 16.

The Processor Clock is the Master Clock.

6. Selection of programmable clocks

Programmable clocks are controlled via registers; PMC_SCER, PMC_SCDR and PMC_SCSR.

Programmable clocks can be enabled and/or disabled via the PMC_SCER and PMC_SCDR registers. Depending on the system used, 4 programmable clocks can be enabled or disabled. The PMC_SCSR provides a clear indication as to which programmable clock is enabled. By default all programmable clocks are disabled.

PMC_PCKx registers are used to configure programmable clocks.

The CSS field is used to select the programmable clock divider source. Four clock options are available: main clock, slow clock, PLLACK, PLLBCK. By default, the clock source selected is slow clock.

The PRES field is used to control the programmable clock prescaler. It is possible to choose between different values (1, 2, 4, 8, 16, 32, 64). Programmable clock output is prescaler input divided by PRES parameter. By default, the PRES parameter is set to 1 which means that master clock is equal to slow clock.

Once the PMC_PCKx register has been programmed, The corresponding programmable clock must be enabled and the user is constrained to wait for the PCKRDYx bit to be set in the PMC_SR register. This can be done either by polling the status register or by waiting the interrupt line to be raised if the associated interrupt to PCKRDYx has been enabled in the PMC_IER register. All parameters in PMC_PCKx can be programmed in a single write operation.

If the CSS and PRES parameters are to be modified, the corresponding programmable clock must be disabled first. The parameters can then be modified. Once this has been done, the user must re-enable the programmable clock and wait for the PCKRDYx bit to be set.

Code Example:

```
write_register(PMC_PCK0,0x00000015)
```

Programmable clock 0 is main clock divided by 32.

7. Enabling Peripheral Clocks

Once all of the previous steps have been completed, the peripheral clocks can be enabled and/or disabled via registers PMC_PCER and PMC_PCDR.

Depending on the system used, 23 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.

30.8 Clock Switching Details

30.8.1 Master Clock Switching Timings

Table 30-1 and Table 30-2 give the worst case timings required for the Master Clock to switch from one selected clock to another one. This is in the event that the prescaler is de-activated. When the prescaler is activated, an additional time of 64 clock cycles of the new selected clock has to be added.

Table 30-1. Clock Switching Timings (Worst Case)

From / To	Main Clock	SLCK	PLL Clock
Main Clock	–	4 x SLCK + 2.5 x Main Clock	3 x PLL Clock + 4 x SLCK + 1 x Main Clock
SLCK	0.5 x Main Clock + 4.5 x SLCK	–	3 x PLL Clock + 5 x SLCK
PLL Clock	0.5 x Main Clock + 4 x SLCK + PLLCOUNT x SLCK + 2.5 x PLLx Clock	2.5 x PLL Clock + 5 x SLCK + PLLCOUNT x SLCK	2.5 x PLL Clock + 4 x SLCK + PLLCOUNT x SLCK

Notes: 1. PLL designates either the PLL A or the PLL B Clock.
2. PLLCOUNT designates either PLLACOUNT or PLLBCOUNT.

Table 30-2. Clock Switching Timings Between Two PLLs (Worst Case)

From / To	PLLA Clock	PLLB Clock
PLLA Clock	2.5 x PLLA Clock + 4 x SLCK + PLLACOUNT x SLCK	3 x PLLA Clock + 4 x SLCK + 1.5 x PLLA Clock
PLLB Clock	3 x PLLB Clock + 4 x SLCK + 1.5 x PLLB Clock	2.5 x PLLB Clock + 4 x SLCK + PLLBCOUNT x SLCK

30.8.2 Clock Switching Waveforms

Figure 30-3. Switch Master Clock from Slow Clock to PLL Clock

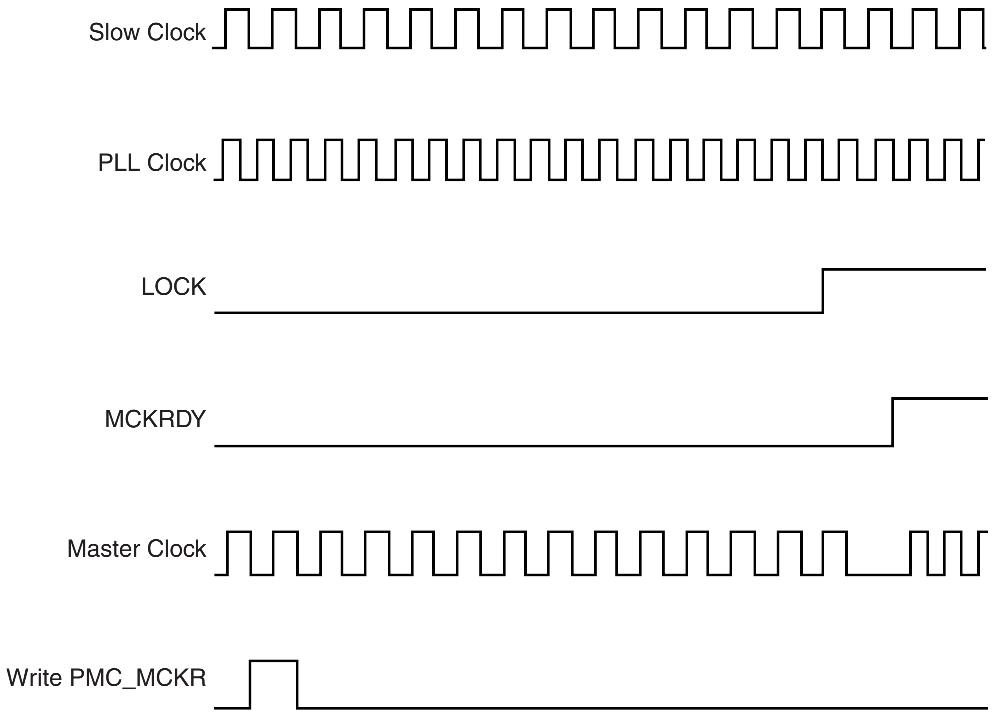


Figure 30-4. Switch Master Clock from Main Clock to Slow Clock

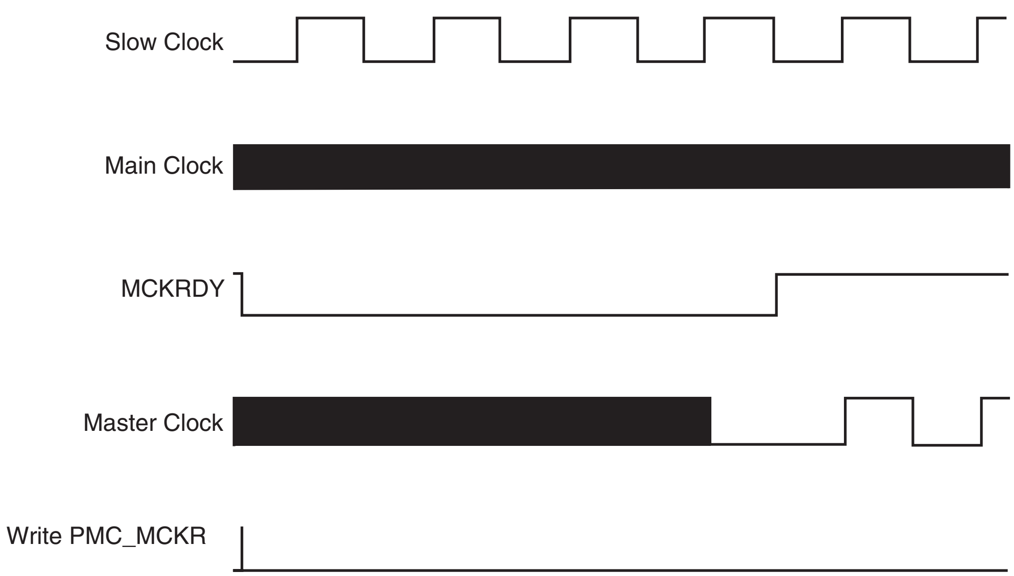


Figure 30-5. Change PLLA Programming

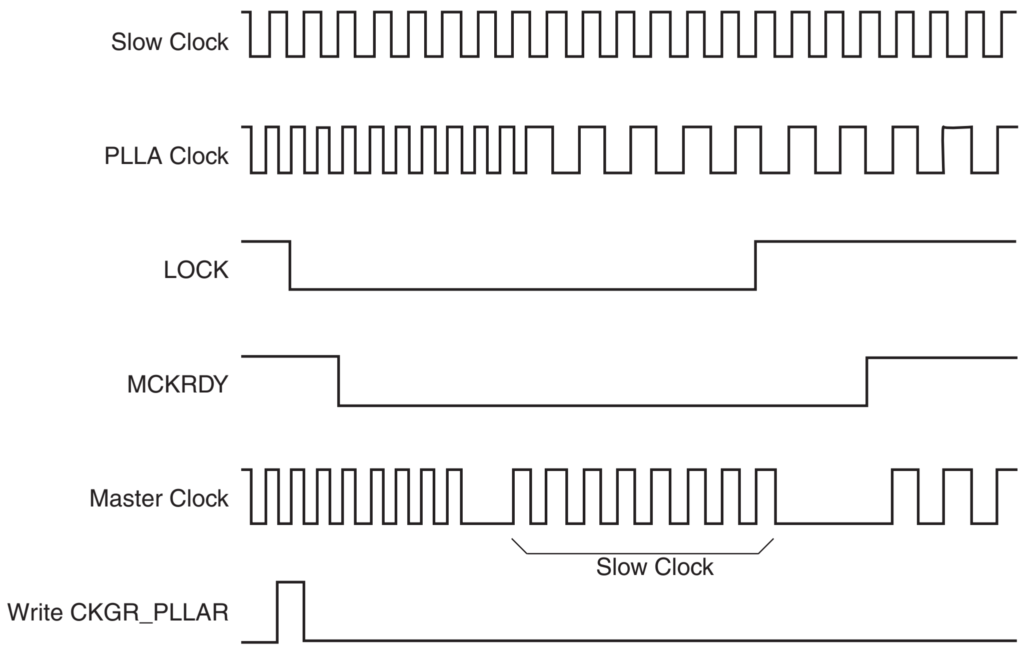


Figure 30-6. Change PLLB Programming

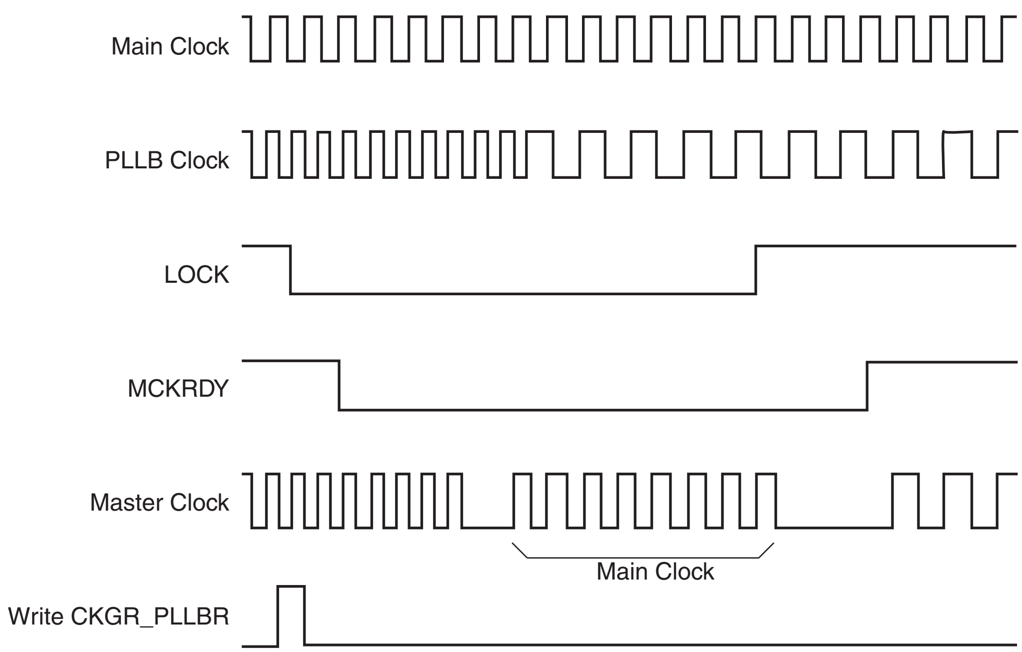
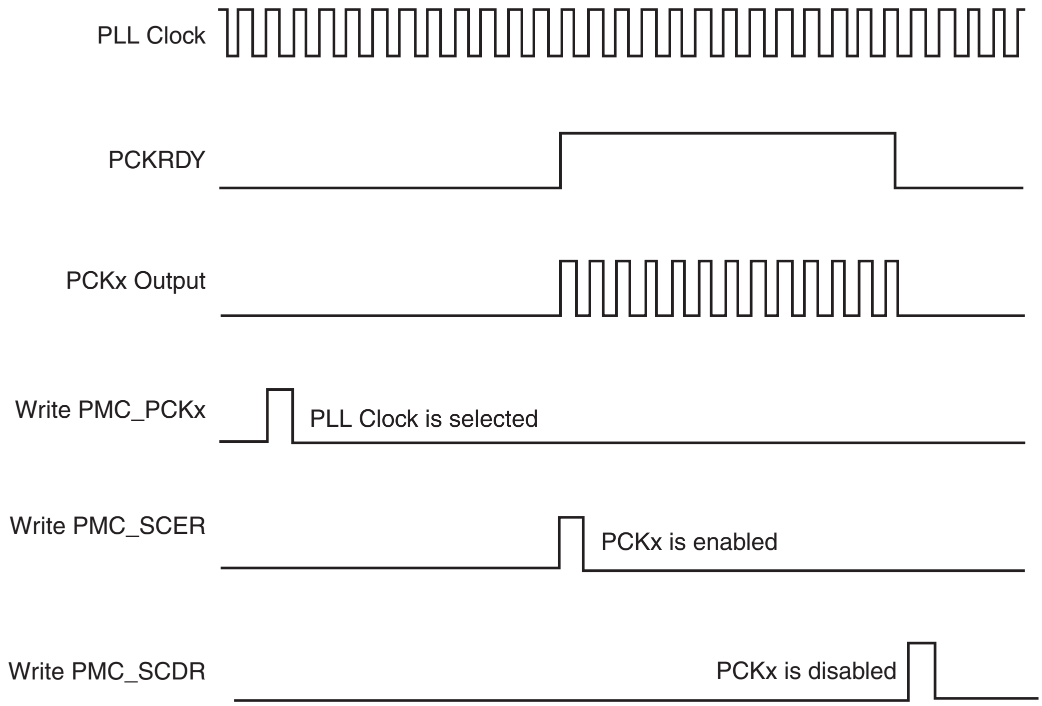


Figure 30-7. Programmable Clock Output Programming



30.9 Power Management Controller (PMC) User Interface

Table 30-3. Register Mapping

Offset	Register	Name	Access	Reset
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	0x03
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	0x1020_0800
0x0020	Main Oscillator Register	CKGR_MOR	Read-write	0x0
0x0024	Main Clock Frequency Register	CKGR_MCFR	Read-only	0x0
0x0028	PLL A Register	CKGR_PLLAR	Read-write	0x3F00
0x002C	PLL B Register	CKGR_PLLBR	Read-write	0x3F00
0x0030	Master Clock Register	PMC_MCKR	Read-write	0x0
0x0038	Reserved	–	–	–
0x003C	Reserved	–	–	–
0x0040	Programmable Clock 0 Register	PMC_PCK0	Read-write	0x0
0x0044	Programmable Clock 1 Register	PMC_PCK1	Read-write	0x0
...	...	...	...	...
0x0060	Interrupt Enable Register	PMC_IER	Write-only	--
0x0064	Interrupt Disable Register	PMC_IDR	Write-only	--
0x0068	Status Register	PMC_SR	Read-only	0x08
0x006C	Interrupt Mask Register	PMC_IMR	Read-only	0x0
0x0070 - 0x007C	Reserved	–	–	–
0x00E4	Write Protect Mode Register	PMC_WPMR	Read-write	0x0
0x00E8	Write Protect Status Register	PMC_WPSR	Read-only	0x0

30.9.1 PMC System Clock Enable Register

Name: PMC_SCER

Address: 0xFFFFFC00

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	PCK3	PCK2	PCK1	PCK0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	UHP	–	–	–	DCKE	–	PCK

- **PCK: Processor Clock Enable**

0 = No effect.

1 = Enables the Processor clock.

- **DCKE: DDR Clock Enable**

0 = No effect.

1 = Enables the DDR clock.

- **UHP: USB Host Port Clock Enable**

0 = No effect.

1 = Enables the 12 and 48 MHz clock of the USB Host Port.

- **PCKx: Programmable Clock x Output Enable**

0 = No effect.

1 = Enables the corresponding programmable Clock output.

30.9.2 PMC System Clock Disable Register

Name: PMC_SCDR

Address: 0xFFFFFC04

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	PCK3	PCK2	PCK1	PCK0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	UHP	–	–	–	DCKD	–	PCK

- **PCK: Processor Clock Disable**

0 = No effect.

1 = Disables the Processor clock. This is used to enter the processor in Idle Mode.

Note: PCK disable action will only take effect if processor enters “Wat for Interrupt” mode.

- **DCKD: DDR Clock Disable**

0 = No effect.

1 = Disables the DDR Clock.

- **UHP: USB Host Port Clock Disable**

0 = No effect.

1 = Disables the 12 and 48 MHz clock of the USB Host Port.

- **PCKx: Programmable Clock x Output Disable**

0 = No effect.

1 = Disables the corresponding programmable clock output.

30.9.3 PMC System Clock Status Register

Name: PMC_SCSR

Address: 0xFFFFFC08

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	PCK3	PCK2	PCK1	PCK0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	UHP	–	–	–	DCKS	–	PCK

- **PCK: Processor Clock Status**

0 = The Processor clock is disabled.

1 = The Processor clock is enabled.

- **DCKS: DDR Clock Status**

0 = The DDR Clock is disabled.

1 = The DDR Clock is enabled.

- **UHP: USB Host Port Clock Status**

0 = The 12 and 48 MHz clock (UHPCK) of the USB Host Port is disabled.

1 = The 12 and 48 MHz clock (UHPCK) of the USB Host Port is enabled.

- **PCKx: Programmable Clock x Output Status**

0 = The corresponding programmable clock output is disabled.

1 = The corresponding programmable clock output is enabled

30.9.4 PMC Peripheral Clock Enable Register

Name: PMC_PCER

Address: 0xFFFFFC10

Access: 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.

30.9.5 PMC Peripheral Clock Disable Register

Name: PMC_PCDR

Address: 0xFFFFFC14

Access: 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.

30.9.6 PMC Peripheral Clock Status Register

Name: PMC_PCSR

Address: 0xFFFFFC18

Access: 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.

30.9.7 PMC UTMI Clock Configuration Register

Name: CKGR_UCKR

Address: 0xFFFFFC1C

Access: 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. Should be set to 2.

- **BIASEN: UTMI BIAS Enable**

0 = The UTMI BIAS is disabled.

1 = The UTMI BIAS is enabled.

- **BIASCOUNT: UTMI BIAS Start-up Time**

Specifies the number of Slow Clock cycles for the UTMI BIAS start-up time. Should be set to 2.

30.9.8 PMC Clock Generator Main Oscillator Register

Name: CKGR_MOR

Address: 0xFFFFFC20

Access: 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.

30.9.9 PMC Clock Generator Main Clock Frequency Register

Name: CKGR_MCFR

Address: 0xFFFFFC24

Access: 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.

30.9.10 PMC Clock Generator PLL A Register

Name: CKGR_PLLAR

Address: 0xFFFFFC28

Access: Read-write

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

23	22	21	20	19	18	17	16
MULA							

15	14	13	12	11	10	9	8
OUTA		PLLACOUNT					

7	6	5	4	3	2	1	0
DIVA							

Possible limitations on PLL A input frequencies and multiplier factors should be checked before using the PMC.

Warning: Bit 29 must always be set to 1 when programming the CKGR_PLLAR register.

- **DIVA: Divider A**

DIVA	Divider Selected
0	Divider output is 0
1	Divider is bypassed
2 - 255	Divider output is the Main Clock divided by DIVA.

- **PLLACOUNT: PLL A Counter**

Specifies the number of Slow Clock cycles before the LOCKA bit is set in PMC_SR after CKGR_PLLAR is written.

- **OUTA: PLL A Clock Frequency Range**

To optimize clock performance, this field must be programmed as specified in “PLL Characteristics” in the Electrical Characteristics section of the product datasheet.

- **MULA: PLL A Multiplier**

0 = The PLL A is deactivated.

1 up to 2047 = The PLL A Clock frequency is the PLL A input frequency multiplied by MULA + 1.

30.9.11 PMC Clock Generator PLL B Register

Name: CKGR_PLLBR

Address: 0xFFFFFC2C

Access: Read-write

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

Possible limitations on PLL B input frequencies and multiplier factors should be checked before using the PMC.

- **DIVB: Divider B**

DIVB	Divider Selected
0	Divider output is 0
1	Divider is bypassed
2 - 255	Divider output is the selected clock divided by DIVB.

- **PLLBCOUNT: PLL B Counter**

Specifies the number of slow clock cycles before the LOCKB bit is set in PMC_SR after CKGR_PLLBR is written.

- **OUTB: PLLB Clock Frequency Range**

To optimize clock performance, this field must be programmed as specified in “PLL Characteristics” in the Electrical Characteristics section of the product datasheet.

- **MULB: PLL B Multiplier**

0 = The PLL B is deactivated.

1 up to 2047 = The PLL B Clock frequency is the PLL B input frequency multiplied by MULB + 1.

- **USBDIV: Divider for USB Clock**

The division ratio is: USBDIV +1

USBDIV				Divider for USB Clock(s)
0	0	0	0	Divider output is PLL B clock output.
0	0	0	1	Divider output is PLL B clock output divided by 2.
0	0	1	0	Divider output is PLL B clock output divided by 3.
0	0	1	1	Divider output is PLL B clock output divided by 4.
0	1	0	0	Divider output is PLL B clock output divided by 5
0	1	0	1	Divider output is PLL B clock output divided by 6
0	1	1	0	Divider output is PLL B clock output divided by 7

USBDIV				Divider for USB Clock(s)
0	1	1	1	Divider output is PLL B clock output divided by 8
1	0	0	0	Divider output is PLL B clock output divided by 9
1	0	0	1	Divider output is PLL B clock output divided by 10
1	0	1	0	Divider output is PLL B clock output divided by 11
1	0	1	1	Divider output is PLL B clock output divided by 12
1	1	0	0	Divider output is PLL B clock output divided by 13
1	1	0	1	Divider output is PLL B clock output divided by 14
1	1	1	0	Divider output is PLL B clock output divided by 15
1	1	1	1	Divider output is PLL B clock output divided by 16

30.9.12 PMC Master Clock Register

Name: PMC_MCKR

Address: 0xFFFFFC30

Access: 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**
–	–	–	PLLADIV2	–	–	MDIV	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	PRES			CSS	

- **CSS: Master Clock Selection**

CSS		Clock Source Selection
0	0	Slow Clock is selected
0	1	Main Clock is selected
1	0	PLL A Clock is selected
1	1	PLL B Clock is selected

- **PRES: Processor Clock Prescaler**

PRES			Processor Clock
0	0	0	Selected clock
0	0	1	Selected clock divided by 2
0	1	0	Selected clock divided by 4
0	1	1	Selected clock divided by 8
1	0	0	Selected clock divided by 16
1	0	1	Selected clock divided by 32
1	1	0	Selected clock divided by 64
1	1	1	Reserved

- **PLLADIV2: PLL A Divided by 2**

PLLADIV2	PLL A Division
0	PLL A Frequency is divided by 1.
1	PLL A Frequency is divided by 2.

- **PDIV Processor Clock Division**

0 = Processor Clock is Prescaler Output Clock divided by 1.

1 = Processor Clock is Prescaler Output Clock divided by 2.

- **MDIV: Master Clock Division**

MDIV		Master Clock Division
0	0	Master Clock is Processor Clock. DDRCK is unusable
0	1	Master Clock is Processor Clock divided by 2. DDRCK is 2xMCK
1	0	Master Clock is Processor Clock divided by 4. DDRCK is 2xMCK
1	1	Reserved.

30.9.13 PMC Programmable Clock Register

Name: PMC_PCKx

Address: 0xFFFFFC40

Access: Read-write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	PRES			CSS	

- **CSS: Master Clock Selection**

CSS		Clock Source Selection
0	0	Slow Clock is selected
0	1	Main Clock is selected
1	0	PLL A Clock is selected
1	1	PLL B Clock is selected

- **PRES: Programmable Clock Prescaler**

PRES			Programmable Clock
0	0	0	Selected clock
0	0	1	Selected clock divided by 2
0	1	0	Selected clock divided by 4
0	1	1	Selected clock divided by 8
1	0	0	Selected clock divided by 16
1	0	1	Selected clock divided by 32
1	1	0	Selected clock divided by 64
1	1	1	Reserved

30.9.14 PMC Interrupt Enable Register

Name: PMC_IER

Address: 0xFFFFFC60

Access: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	PCKRDY3	PCKRDY2	PCKRDY1	PCKRDY0

7	6	5	4	3	2	1	0
–	LOCKU	–	ACKRDY	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: Main Oscillator Status Interrupt Enable**

- **LOCKA: PLL A Lock Interrupt Enable**

- **LOCKB: PLL B Lock Interrupt Enable**

- **MCKRDY: Master Clock Ready Interrupt Enable**

- **ACKRDY: Application Clock Ready Interrupt Enable**

- **PCKRDYx: Programmable Clock Ready x Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

- **LOCKU: UTMI PLL Lock Interrupt Enable**

30.9.15 PMC Interrupt Disable Register

Name: PMC_IDR

Address: 0xFFFFFC64

Access: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	PCKRDY3	PCKRDY2	PCKRDY1	PCKRDY0

7	6	5	4	3	2	1	0
–	LOCKU	–	ACKRDY	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: Main Oscillator Status Interrupt Disable**

- **LOCKA: PLL A Lock Interrupt Disable**

- **LOCKB: PLL B Lock Interrupt Disable**

- **MCKRDY: Master Clock Ready Interrupt Disable**

- **ACKRDY: Application Clock Ready Interrupt Disable**

- **PCKRDYx: Programmable Clock Ready x Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

- **LOCKU: UTMI PLL Lock Interrupt Disable**

30.9.16 PMC Status Register

Name: PMC_SR

Address: 0xFFFFFC68

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	PCKRDY3	PCKRDY2	PCKRDY1	PCKRDY0

7	6	5	4	3	2	1	0
–	LOCKU	–	ACKRDY	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: MOSCS Flag Status**

0 = Main oscillator is not stabilized.

1 = Main oscillator is stabilized.

- **LOCKA: PLL A Lock Status**

0 = PLL A is not locked

1 = PLL A is locked.

- **LOCKB: PLL B Lock Status**

0 = PLL B is not locked.

1 = PLL B is locked.

- **MCKRDY: Master Clock Status**

0 = Master Clock is not ready.

1 = Master Clock is ready.

- **ACKRDY: Application Clock Status**

0 = Application Clock is not ready.

1 = Application Clock is ready.

- **PCKRDYx: Programmable Clock Ready Status**

0 = Programmable Clock x is not ready.

1 = Programmable Clock x is ready.

- **LOCKU: UTMI PLL Lock Interrupt Disable**

30.9.17 PMC Interrupt Mask Register

Name: PMC_IMR

Address: 0xFFFFFC6C

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	PCKRDY3	PCKRDY2	PCKRDY1	PCKRDY0

7	6	5	4	3	2	1	0
–	LOCKU	–	ACKRDY	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: Main Oscillator Status Interrupt Mask**

- **LOCKA: PLL A Lock Interrupt Mask**

- **LOCKB: PLL B Lock Interrupt Mask**

- **MCKRDY: Master Clock Ready Interrupt Mask**

- **ACKRDY: Application Clock Ready Interrupt Mask**

- **PCKRDYx: Programmable Clock Ready x Interrupt Mask**

0 = The corresponding interrupt is enabled.

1 = The corresponding interrupt is disabled.

- **LOCKU: UTMI PLL Lock Interrupt Mask**

30.9.18 Write Protect Mode Register

Name: PMC_WPMR

Address: 0xFFFFFCE4

Access: Read-write

31	30	29	28	27	26	25	24
WPKEY							

23	22	21	20	19	18	17	16
WPKEY							

15	14	13	12	11	10	9	8
WPKEY							

7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	WPEN

- **WPEN: Write Protect Enable**

0: Disables the Write Protect if WPKEY corresponds to 0x504D43 (“PMC” in ASCII).

1: Disables the Write Protect if WPKEY corresponds to 0x504D43 (“PMC” in ASCII).

Protects the registers:

- “PMC System Clock Enable Register” on page 406
- “PMC System Clock Disable Register” on page 407
- “PMC Peripheral Clock Enable Register” on page 409
- “PMC Peripheral Clock Disable Register” on page 410
- “PMC UTMI Clock Configuration Register” on page 412
- “PMC Clock Generator Main Oscillator Register” on page 413
- “PMC Clock Generator PLL A Register” on page 415
- “PMC Clock Generator PLL B Register” on page 416
- “PMC Master Clock Register” on page 418
- “PMC Programmable Clock Register” on page 420

- **WPKEY: Write Protect KEY**

Should be written at value 0x504D43 (“PMC” in ASCII). Writting any other value in this field aborts the write operation of the WPEN bit. Always reads at 0.

30.9.19 Write Protect Status Register

Name: PMC_WPSR

Address: 0xFFFFFCE8

Access: Read-only

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

- **WPVS: Write Protect Violation Status**

0 = No Write Protect Violation has occurred since the last read of the PMC_WPSR register.

1 = A Write Protect Violation has occurred since the last read of the PMC_WPSR register. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protect Violation Source**

When WPVS is active, this field indicates the write-protected register (through address offset or code) in which a write access has been attempted.

Note: Reading PMC_WPSR automatically clears all fields.

31. Advanced Interrupt Controller (AIC)

31.1 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.

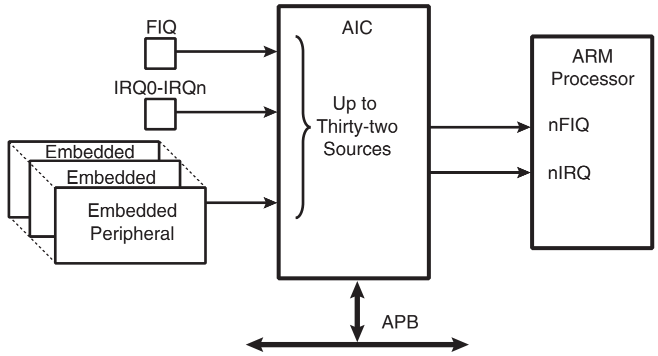
The 8-level Priority Controller allows the user to define the priority for each interrupt source, thus permitting higher priority interrupts to be serviced even if a lower priority interrupt is being treated.

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.

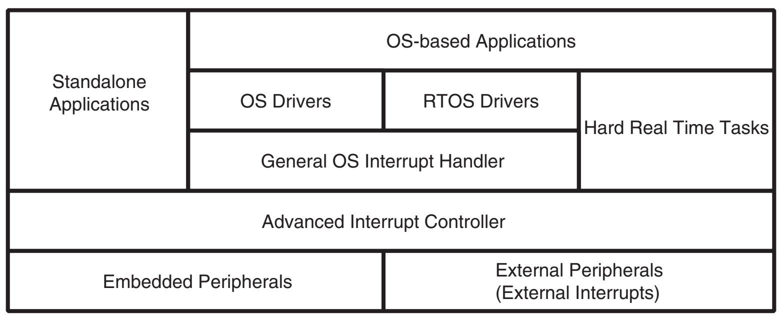
31.2 Block Diagram

Figure 31-1. Block Diagram



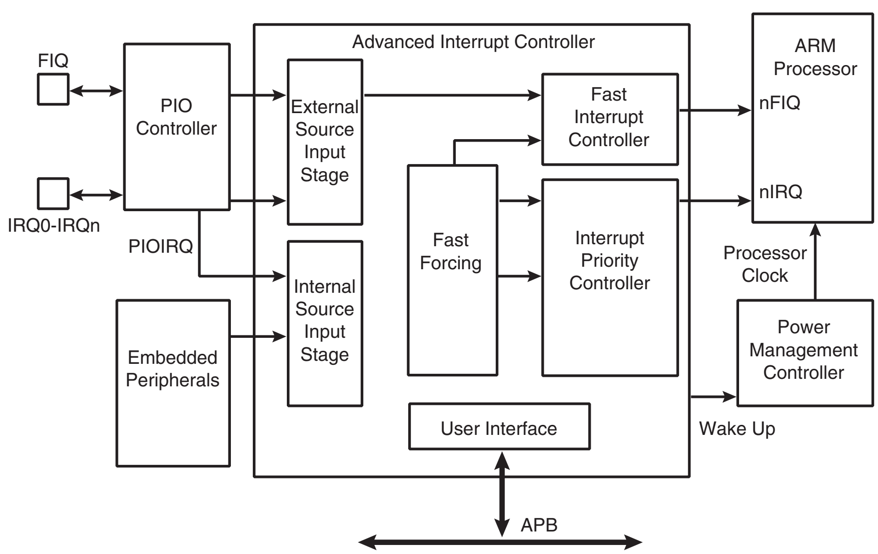
31.3 Application Block Diagram

Figure 31-2. Description of the Application Block



31.4 AIC Detailed Block Diagram

Figure 31-3. AIC Detailed Block Diagram



31.5 I/O Line Description

Table 31-1. I/O Line Description

Pin Name	Pin Description	Type
FIQ	Fast Interrupt	Input
IRQ0 - IRQn	Interrupt 0 - Interrupt n	Input

31.6 Product Dependencies

31.6.1 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.

Table 31-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
AIC	FIQ	PD4	A
AIC	IRQ0	PA10	A
AIC	IRQ1	PA14	B

31.6.2 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.

31.6.3 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. 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.

31.7 Functional Description

31.7.1 Interrupt Source Control

31.7.1.1 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.

31.7.1.2 Interrupt Source Enabling

Each interrupt source, including the FIQ in source 0, can be enabled or disabled by using the command registers; AIC_IECR (Interrupt Enable Command Register) and AIC_IDCR (Interrupt Disable Command Register). This set of registers conducts enabling or disabling in one instruction. The interrupt mask can be read in the AIC_IMR register. A disabled interrupt does not affect servicing of other interrupts.

31.7.1.3 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 433.) 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 437.)

The automatic clear of the interrupt source 0 is performed when AIC_FVR is read.

31.7.1.4 Interrupt Status

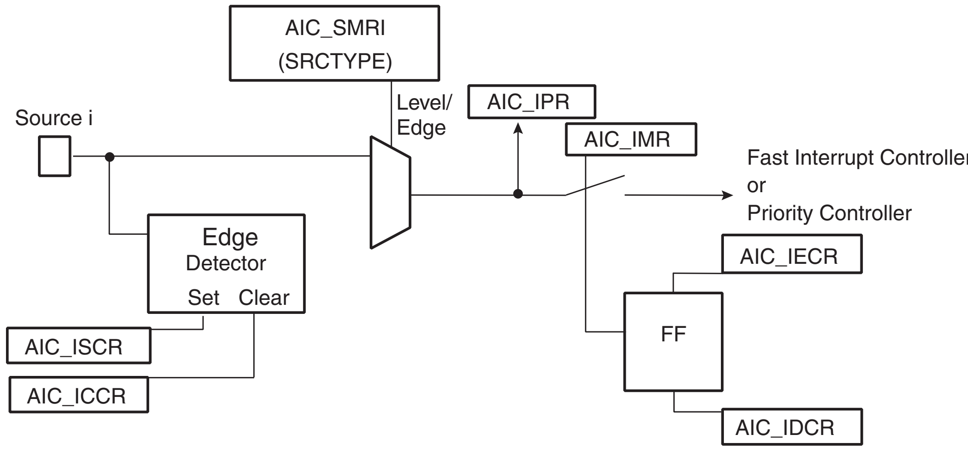
For each interrupt, the AIC operation originates in AIC_IPR (Interrupt Pending Register) and its mask in AIC_IMR (Interrupt Mask Register). AIC_IPR enables the actual activity of the sources, whether masked or not.

The AIC_ISR register reads the number of the current interrupt (see "Priority Controller" on page 433) 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.

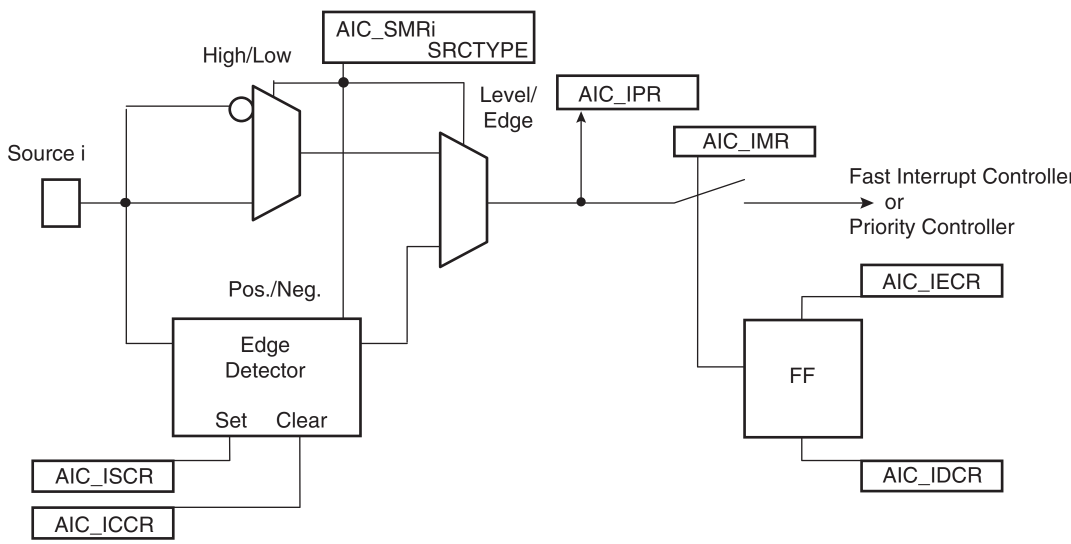
31.7.1.5 Internal Interrupt Source Input Stage

Figure 31-4. Internal Interrupt Source Input Stage



31.7.1.6 External Interrupt Source Input Stage

Figure 31-5. External Interrupt Source Input Stage



31.7.2 Interrupt Latencies

Global interrupt latencies depend on several parameters, including:

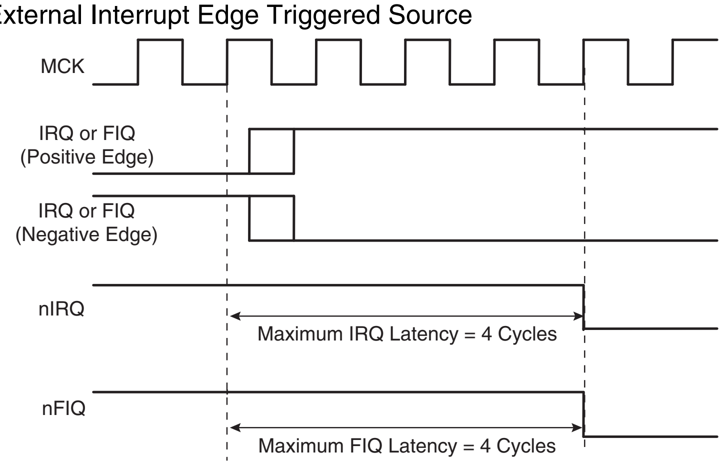
- The time the software masks the interrupts.
- Occurrence, either at the processor level or at the AIC level.
- The execution time of the instruction in progress when the interrupt occurs.
- The treatment of higher priority interrupts and the resynchronization of the hardware signals.

This section addresses only the hardware resynchronization. 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.

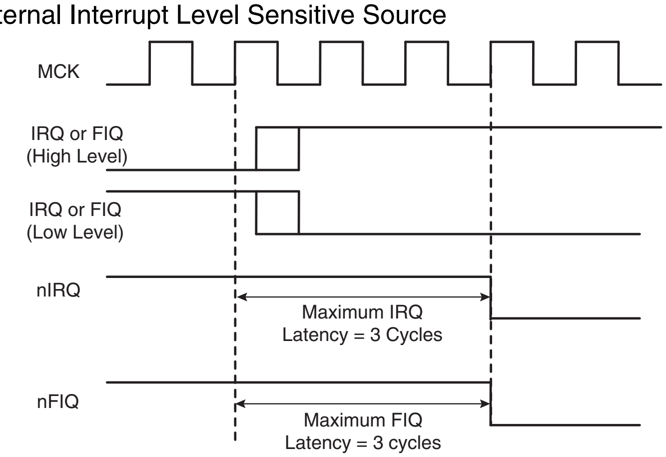
31.7.2.1 External Interrupt Edge Triggered Source

Figure 31-6. External Interrupt Edge Triggered Source



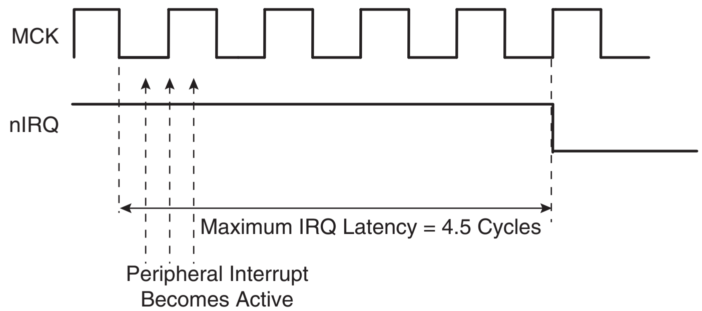
31.7.2.2 External Interrupt Level Sensitive Source

Figure 31-7. External Interrupt Level Sensitive Source



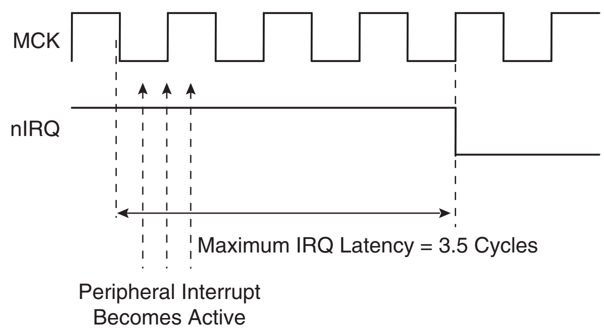
31.7.2.3 Internal Interrupt Edge Triggered Source

Figure 31-8. Internal Interrupt Edge Triggered Source



31.7.2.4 Internal Interrupt Level Sensitive Source

Figure 31-9. Internal Interrupt Level Sensitive Source



31.7.3 Normal Interrupt

31.7.3.1 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**.

31.7.3.2 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.

31.7.3.3 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.

31.7.3.4 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 will be used and restoring them at the end. During this phase, an interrupt of higher priority than the current level will restart the sequence from step 1.

Note: If the interrupt is programmed to be level sensitive, the source of the interrupt must be cleared during this phase.

7. The "I" bit in CPSR must be set in order to mask interrupts before exiting to ensure that the interrupt is completed in an orderly manner.
8. The End of Interrupt Command Register (AIC_EOICR) must be written in order to indicate to the AIC that the current interrupt is finished. This causes the current level to be popped from the stack, restoring the previous current level if one exists on the stack. If another interrupt is pending, with lower or equal priority than the old current level but with higher priority than the new current level, the nIRQ line is re-asserted, but the interrupt sequence does not immediately start because the "I" bit is set in the core. SPSR_irq is restored. Finally, the saved value of the link register is restored directly into the PC. This has the effect of returning from the interrupt to whatever was being executed before, and of loading the CPSR with the stored SPSR, masking or unmasking the interrupts depending on the state saved in SPSR_irq.

Note: The "I" bit in SPSR is significant. If it is set, it indicates that the ARM core was on the verge of masking an interrupt when the mask instruction was interrupted. Hence, when SPSR is restored, the mask instruction is completed (interrupt is masked).

31.7.4 Fast Interrupt

31.7.4.1 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.

31.7.4.2 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.

31.7.4.3 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.

31.7.4.4 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.

31.7.4.5 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).

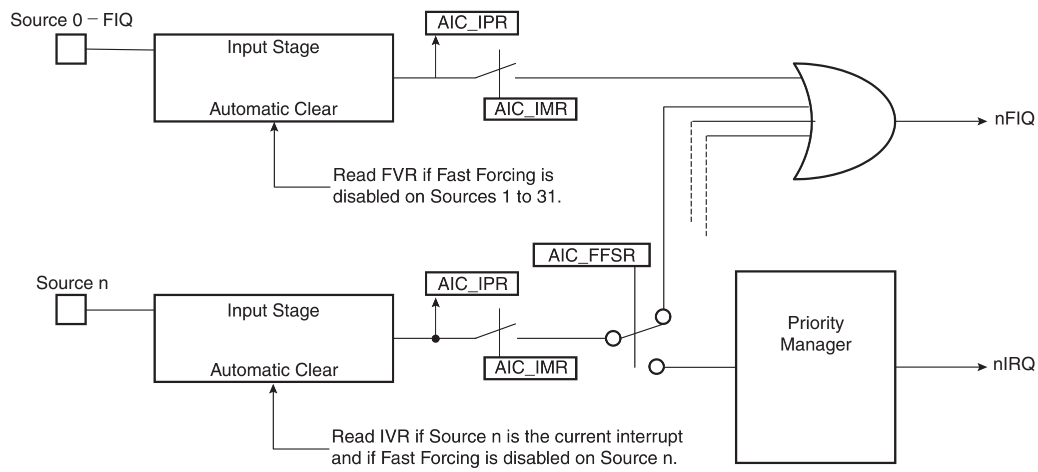
The FIQ Vector Register (AIC_FVR) reads the contents of the Source Vector Register 0 (AIC_SVR0), whatever the source of the fast interrupt may be. The read of the FVR does not clear the Source 0 when the fast forcing feature is used and the interrupt source should be cleared by writing to the Interrupt 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 31-10. Fast Forcing



31.7.5 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 PROT in AIC_DCR (Debug Control Register) at 0x1 enables the Protect Mode.

When the Protect Mode is enabled, the AIC performs interrupt stacking only when a write access is performed on the AIC_IVR. Therefore, the Interrupt Service Routines must write (arbitrary data) to the AIC_IVR just after reading it. The new context of the AIC, including the value of the Interrupt Status Register (AIC_ISR), is updated with the current interrupt only when AIC_IVR is written.

An AIC_IVR read on its own (e.g., by a debugger), modifies neither the AIC context nor the AIC_ISR. Extra AIC_IVR reads perform the same operations. However, it is recommended to not stop the processor between the read and the write of AIC_IVR of the interrupt service routine to make sure the debugger does not modify the AIC context.

To summarize, in normal operating mode, the read of AIC_IVR performs the following operations within the AIC:

1. Calculates active interrupt (higher than current or spurious).
2. Determines and returns the vector of the active interrupt.
3. Memorizes the interrupt.
4. Pushes the current priority level onto the internal stack.
5. Acknowledges the interrupt.

However, while the Protect Mode is activated, only operations 1 to 3 are performed when AIC_IVR is read. Operations 4 and 5 are only performed by the AIC when AIC_IVR is written.

Software that has been written and debugged using the Protect Mode runs correctly in Normal Mode without modification. However, in Normal Mode the AIC_IVR write has no effect and can be removed to optimize the code.

31.7.6 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.

31.7.7 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.

31.8 Advanced Interrupt Controller (AIC) User Interface

31.8.1 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 ± 4-Kbyte offset.

Table 31-3. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Source Mode Register 0	AIC_SMR0	Read-write	0x0
0x04	Source Mode Register 1	AIC_SMR1	Read-write	0x0
---	---	---	---	---
0x7C	Source Mode Register 31	AIC_SMR31	Read-write	0x0
0x80	Source Vector Register 0	AIC_SVR0	Read-write	0x0
0x84	Source Vector Register 1	AIC_SVR1	Read-write	0x0
---	---	---	---	---
0xFC	Source Vector Register 31	AIC_SVR31	Read-write	0x0
0x100	Interrupt Vector Register	AIC_IVR	Read-only	0x0
0x104	FIQ Interrupt Vector Register	AIC_FVR	Read-only	0x0
0x108	Interrupt Status Register	AIC_ISR	Read-only	0x0
0x10C	Interrupt Pending Register(2)	AIC_IPR	Read-only	0x0(1)
0x110	Interrupt Mask Register(2)	AIC_IMR	Read-only	0x0
0x114	Core Interrupt Status Register	AIC_CISR	Read-only	0x0
0x118 - 0x11C	Reserved	---	---	---
0x120	Interrupt Enable Command Register(2)	AIC_IECR	Write-only	---
0x124	Interrupt Disable Command Register(2)	AIC_IDCR	Write-only	---
0x128	Interrupt Clear Command Register(2)	AIC_ICCR	Write-only	---
0x12C	Interrupt Set Command Register(2)	AIC_ISCR	Write-only	---
0x130	End of Interrupt Command Register	AIC_EOICR	Write-only	---
0x134	Spurious Interrupt Vector Register	AIC_SPU	Read-write	0x0
0x138	Debug Control Register	AIC_DCR	Read-write	0x0
0x13C	Reserved	---	---	---
0x140	Fast Forcing Enable Register(2)	AIC_FFER	Write-only	---
0x144	Fast Forcing Disable Register(2)	AIC_FFDR	Write-only	---
0x148	Fast Forcing Status Register(2)	AIC_FFSR	Read-only	0x0
0x14C - 0x1E0	Reserved	---	---	---
0x1E4	Write Protect Mode Register	AIC_WPMR	Read-write	0x0
0x1E8	Write Protect Status Register	AIC_WPSR	Read-only	0x0
0x1EC - 0x1FC	Reserved			

Notes:
1. The reset value of this register depends on the level of the external interrupt source. All other sources are cleared at reset, thus not pending.
2. PID2...PID31 bit fields refer to the identifiers as defined in the Peripheral Identifiers Section of the product datasheet.
3. Values in the Version Register vary with the version of the IP block implementation.

31.8.2 AIC Source Mode Register

Name: AIC_SMR0..AIC_SMR31

Address: 0xFFFFF000

Access: Read-write

Reset: 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

31.8.3 AIC Source Vector Register

Name: AIC_SVR0..AIC_SVR31

Address: 0xFFFFF080

Access: Read-write

Reset: 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.

31.8.4 AIC Interrupt Vector Register

Name: AIC_IVR

Address: 0xFFFFF100

Access: Read-only

Reset: 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.

31.8.5 AIC FIQ Vector Register

Name: AIC_FVR

Address: 0xFFFFF104

Access: Read-only

Reset: 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.

31.8.6 AIC Interrupt Status Register

Name: AIC_ISR

Address: 0xFFFFF108

Access: Read-only

Reset: 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.

31.8.7 AIC Interrupt Pending Register

Name: AIC_IPR

Address: 0xFFFFF10C

Access: Read-only

Reset: 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.

31.8.8 AIC Interrupt Mask Register

Name: AIC_IMR

Address: 0xFFFFF110

Access: Read-only

Reset: 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.

31.8.9 AIC Core Interrupt Status Register

Name: AIC_CISR

Address: 0xFFFFF114

Access: Read-only

Reset: 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.

31.8.10 AIC Interrupt Enable Command Register

Name: AIC_IECR

Address: 0xFFFFF120

Access: 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 Enable**

0 = No effect.

1 = Enables corresponding interrupt.

31.8.11 AIC Interrupt Disable Command Register

Name: AIC_IDCR

Address: 0xFFFFF124

Access: 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.

31.8.12 AIC Interrupt Clear Command Register

Name: AIC_ICCR

Address: 0xFFFFF128

Access: 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.

31.8.13 AIC Interrupt Set Command Register

Name: AIC_ISCR

Address: 0xFFFFF12C

Access: 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.

31.8.14 AIC End of Interrupt Command Register

Name: AIC_EOICR

Address: 0xFFFFF130

Access: Write-only

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

31.8.15 AIC Spurious Interrupt Vector Register

Name: AIC_SPU

Address: 0xFFFFF134

Access: Read-write

Reset: 0x0

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

- **SIVR: 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.

31.8.16 AIC Debug Control Register

Name: AIC_DCR

Address: 0xFFFFF138

Access: Read-write

Reset: 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.

31.8.17 AIC Fast Forcing Enable Register

Name: AIC_FFER

Address: 0xFFFFF140

Access: 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.

31.8.18 AIC Fast Forcing Disable Register

Name: AIC_FFDR

Address: 0xFFFFF144

Access: 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.

31.8.19 AIC Fast Forcing Status Register

Name: AIC_FFSR

Address: 0xFFFFF148

Access: 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.

31.8.20 AIC Write Protect Mode Register

Name: AIC_WPMR

Address: 0xFFFFF1E4

Access: Read-write

Reset: See Table 31-3

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

- **WPEN: Write Protect Enable**

0 = Disables the Write Protect if WPKEY corresponds to 0x414943 (“AIC” in ASCII).

1 = Enables the Write Protect if WPKEY corresponds to 0x414943 (“AIC” in ASCII).

Protects the registers:

- “AIC Source Mode Register” on page 441
- “AIC Source Vector Register” on page 442
- “AIC Spurious Interrupt Vector Register” on page 448
- “AIC Debug Control Register” on page 448

- **WPKEY: Write Protect KEY**

Should be written at value 0x414943 (“AIC” in ASCII). Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

31.8.21 AIC Write Protect Status Register

Name: AIC_WPSR

Address: 0xFFFFF1E8

Access: Read-only

Reset: See Table 31-3

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

23	22	21	20	19	18	17	16
WPVSRC							

15	14	13	12	11	10	9	8
WPVSRC							

7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	WPVS

- **WPVS: Write Protect Violation Status**

0 = No Write Protect Violation has occurred since the last read of the AIC_WPSR register.

1 = A Write Protect Violation has occurred since the last read of the AIC_WPSR register. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protect Violation Source**

When WPVS is active, this field indicates the write-protected register (through address offset or code) in which a write access has been attempted.

Note: Reading AIC_WPSR automatically clears all fields.

32. Debug Unit (DBGU)

32.1 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. The Debug Unit two-pin UART can be used stand-alone for general purpose serial communication. 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.

32.2 Block Diagram

Figure 32-1. Debug Unit Functional Block Diagram

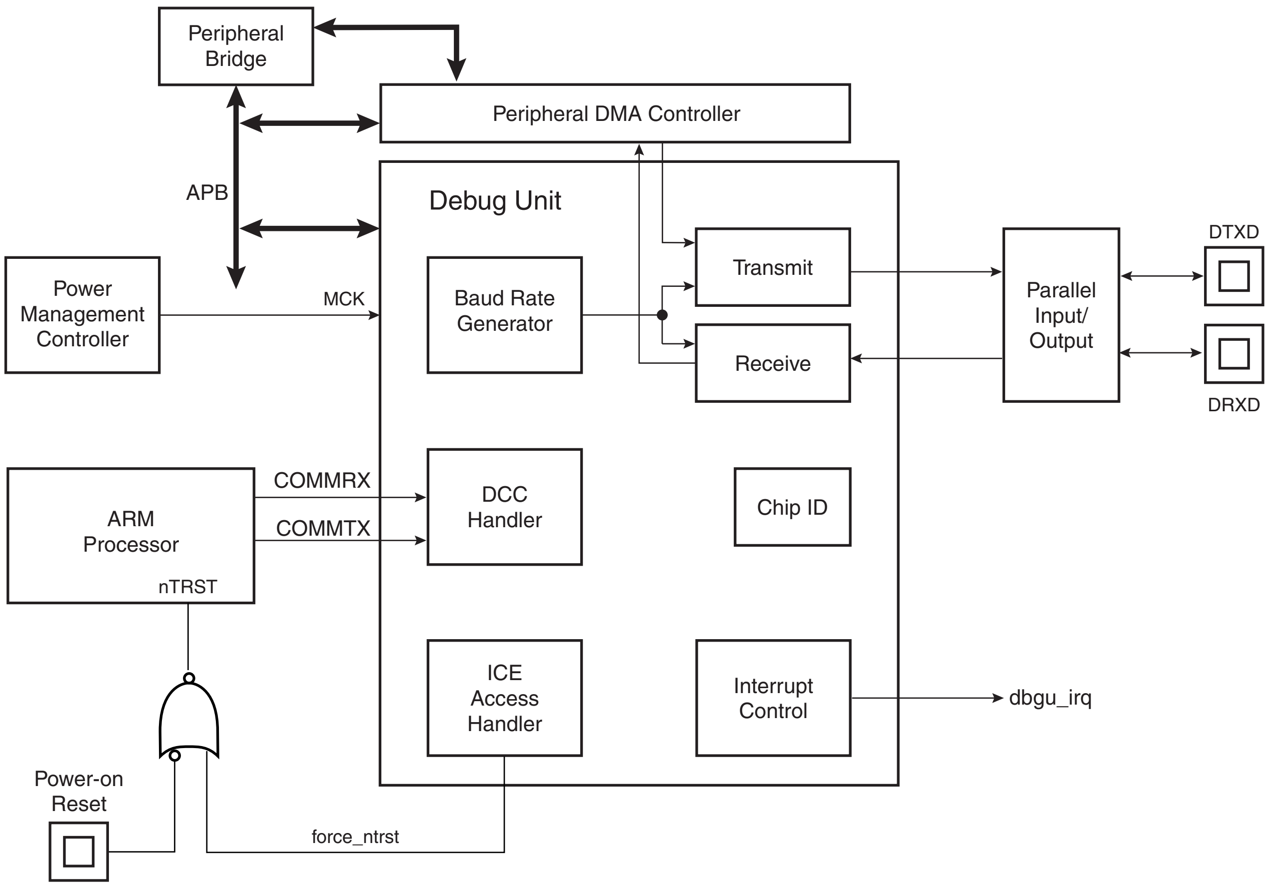
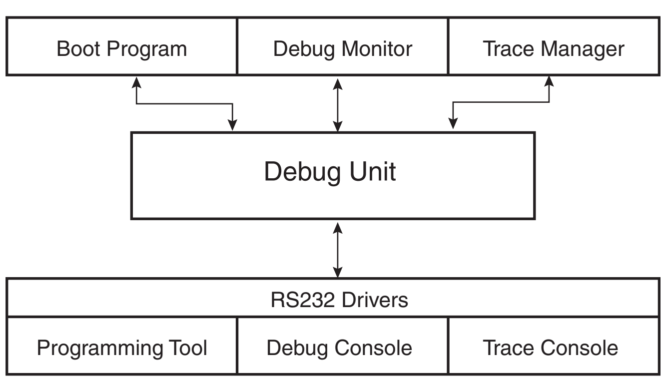


Table 32-1. Debug Unit Pin Description

Pin Name	Description	Type
DRXD	Debug Receive Data	Input
DTXD	Debug Transmit Data	Output

Figure 32-2. Debug Unit Application Example



32.3 Product Dependencies

32.3.1 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.

Table 32-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
DBGU	DRXD	PC30	A
DBGU	DTXD	PC31	A

32.3.2 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.

32.3.3 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 32-1. This sharing requires the programmer to determine the source of the interrupt when the source 1 is triggered.

32.4 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.

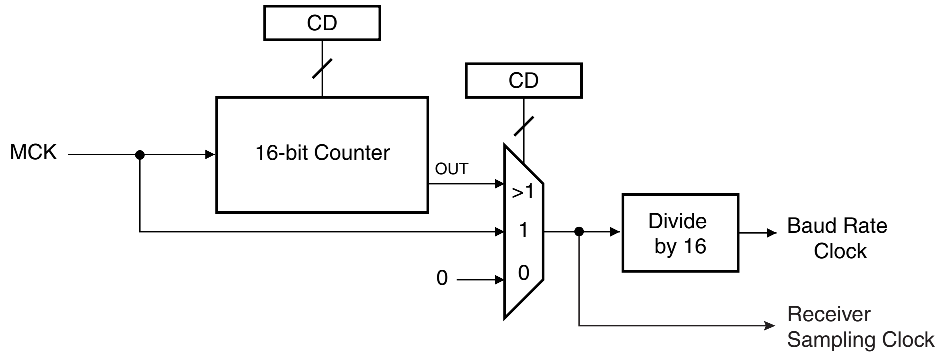
32.4.1 Baud Rate Generator

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 32-3. Baud Rate Generator



32.4.2 Receiver

32.4.2.1 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.

32.4.2.2 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 mid-point 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 32-4. Start Bit Detection

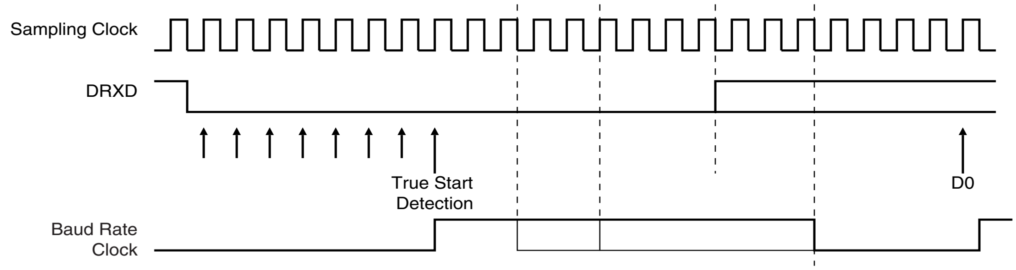
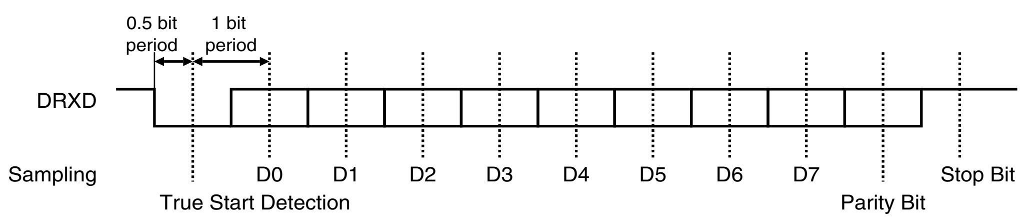


Figure 32-5. Character Reception

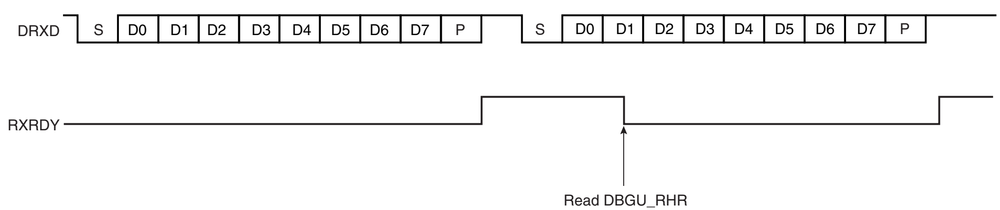
Example: 8-bit, parity enabled 1 stop



32.4.2.3 *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.

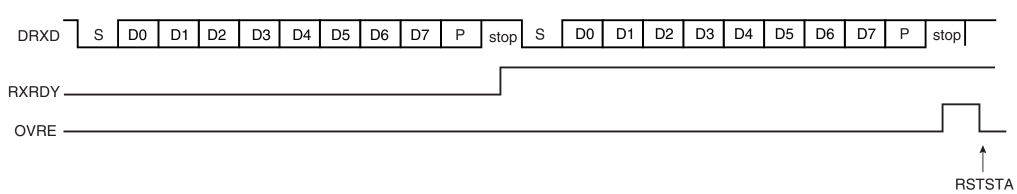
Figure 32-6. Receiver Ready



32.4.2.4 *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.

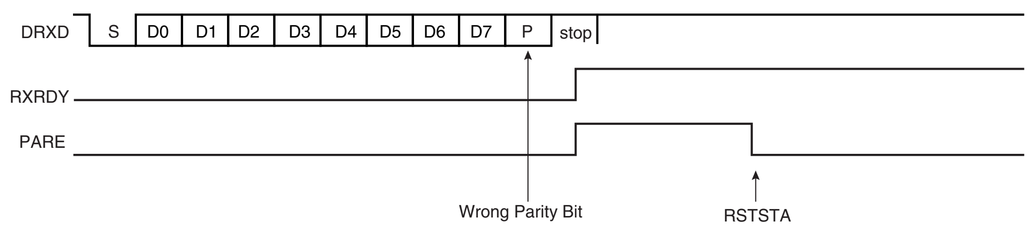
Figure 32-7. Receiver Overrun



32.4.2.5 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.

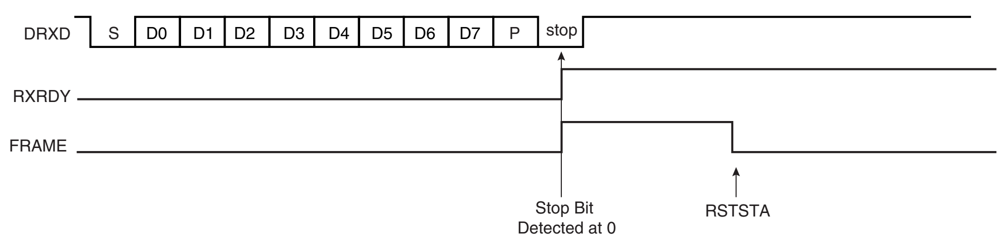
Figure 32-8. Parity Error



32.4.2.6 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 32-9. Receiver Framing Error



32.4.3 Transmitter

32.4.3.1 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.

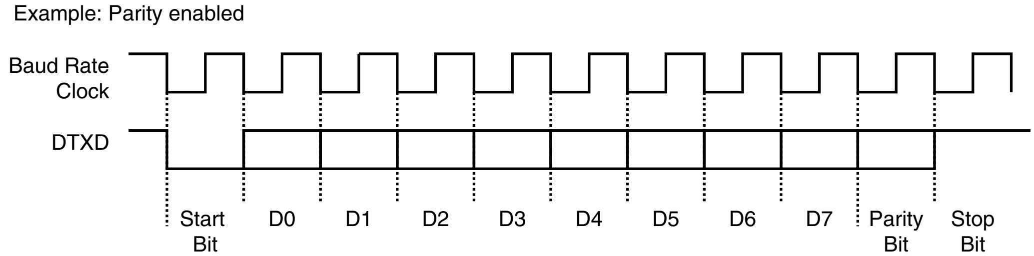
The programmer can disable the transmitter by writing DBGU_CR with the bit TXDIS at 1. If the transmitter is not operating, it is immediately stopped. However, if a character is being processed into the Shift Register and/or a character has been written in the Transmit Holding Register, the characters 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.

32.4.3.2 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 32-10. Character Transmission

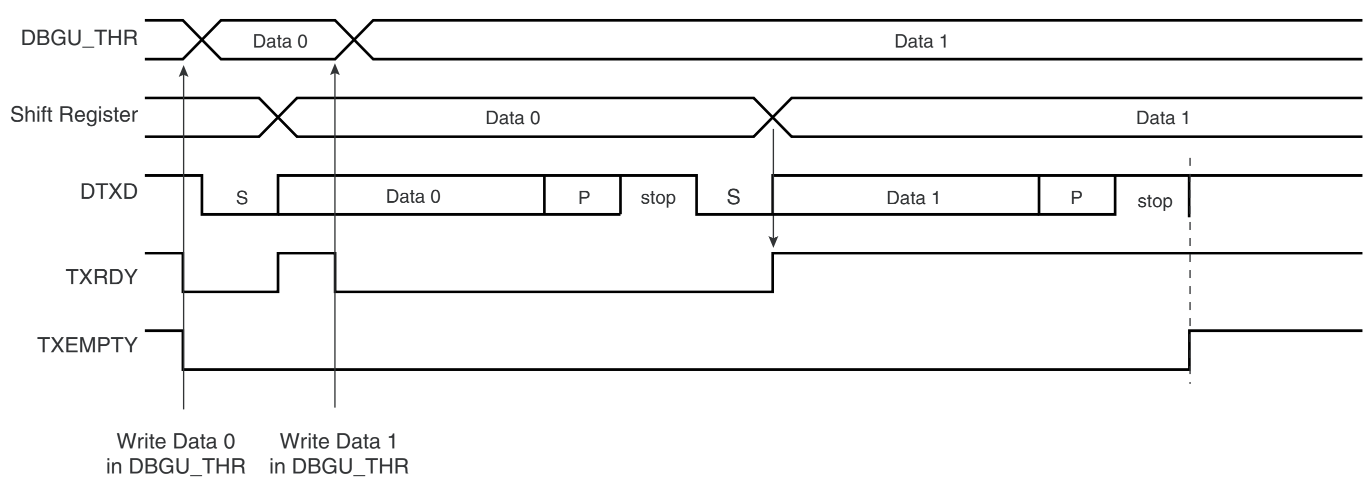


32.4.3.3 Transmitter Control

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 32-11. Transmitter Control



32.4.4 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.

32.4.5 Test Modes

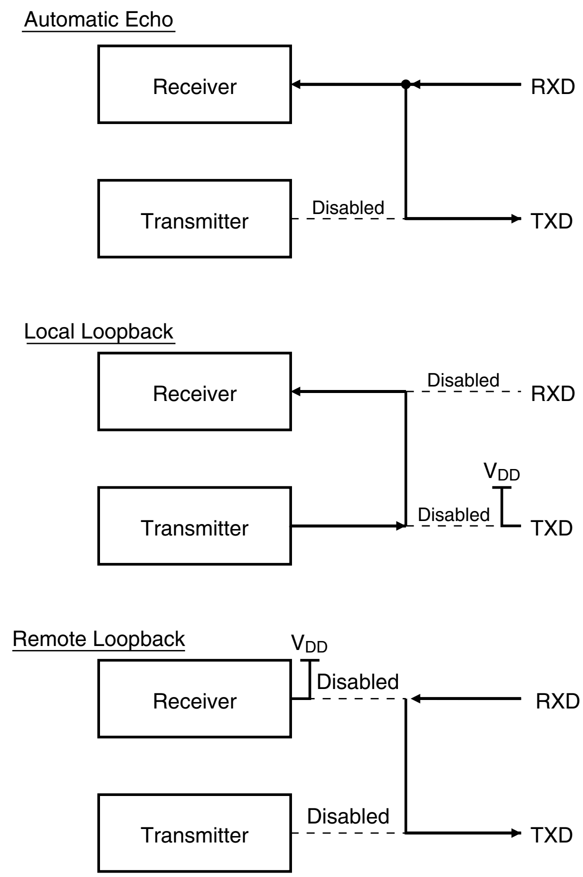
The Debug Unit supports three tests modes. These modes of operation are programmed by using the field CHMODE (Channel Mode) in the mode register DBGU_MR.

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 32-12. Test Modes



32.4.6 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.

32.4.7 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.

32.4.8 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.

32.5 Debug Unit (DBGU) User Interface

Table 32-3. Register Mapping

Offset	Register	Name	Access	Reset
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
0x0100 - 0x0124	PDC Area	–	–	–

32.5.1 Debug Unit Control Register

Name: DBGU_CR

Address: 0xFFFFEE00

Access: Write-only

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

32.5.2 Debug Unit Mode Register

Name: DBGU_MR

Address: 0xFFFFEE04

Access: 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

32.5.3 Debug Unit Interrupt Enable Register

Name: DBGU_IER

Address: 0xFFFFEE08

Access: 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.

32.5.4 Debug Unit Interrupt Disable Register

Name: DBGU_IDR

Address: 0xFFFFEE0C

Access: 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.

32.5.5 Debug Unit Interrupt Mask Register

Name: DBGU_IMR

Address: 0xFFFFEE10

Access: 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.

32.5.6 Debug Unit Status Register

Name: DBGU_SR

Address: 0xFFFFEE14

Access: 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.

32.5.7 Debug Unit Receiver Holding Register

Name: DBGU_RHR

Address: 0xFFFFEE18

Access: Read-only

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

- **RXCHR: Received Character**

Last received character if RXRDY is set.

32.5.8 Debug Unit Transmit Holding Register

Name: DBGU_THR

Address: 0xFFFFEE1C

Access: Write-only

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

32.5.9 Debug Unit Baud Rate Generator Register

Name: DBGU_BRGR

Address: 0xFFFFEE20

Access: Read-write

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

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

15	14	13	12	11	10	9	8
CD							

7	6	5	4	3	2	1	0
CD							

- **CD: Clock Divisor**

CD	Baud Rate Clock
0	Disabled
1	MCK
2 to 65535	MCK / (CD x 16)

32.5.10 Debug Unit Chip ID Register

Name: DBGU_CIDR

Address: 0xFFFFEE40

Access: 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**

Values depend upon the version of the device.

- **EPROC: Embedded Processor**

EPROC			Processor
0	0	1	ARM946ES
0	1	0	ARM7TDMI
1	0	0	ARM920T
1	0	1	ARM926EJS

- **NVPSIZ: Nonvolatile Program Memory Size**

NVPSIZ				Size
0	0	0	0	None
0	0	0	1	8K bytes
0	0	1	0	16K bytes
0	0	1	1	32K bytes
0	1	0	0	Reserved
0	1	0	1	64K bytes
0	1	1	0	Reserved
0	1	1	1	128K bytes
1	0	0	0	Reserved
1	0	0	1	256K bytes
1	0	1	0	512K bytes
1	0	1	1	Reserved
1	1	0	0	1024K bytes
1	1	0	1	Reserved
1	1	1	0	2048K bytes
1	1	1	1	Reserved

- **NVPSIZ2 Second Nonvolatile Program Memory Size**

NVPSIZ2				Size
0	0	0	0	None
0	0	0	1	8K bytes
0	0	1	0	16K bytes
0	0	1	1	32K bytes
0	1	0	0	Reserved
0	1	0	1	64K bytes
0	1	1	0	Reserved
0	1	1	1	128K bytes
1	0	0	0	Reserved
1	0	0	1	256K bytes
1	0	1	0	512K bytes
1	0	1	1	Reserved
1	1	0	0	1024K bytes
1	1	0	1	Reserved
1	1	1	0	2048K bytes
1	1	1	1	Reserved

- **SRAMSIZ: Internal SRAM Size**

SRAMSIZ				Size
0	0	0	0	Reserved
0	0	0	1	1K bytes
0	0	1	0	2K bytes
0	0	1	1	6K bytes
0	1	0	0	112K bytes
0	1	0	1	4K bytes
0	1	1	0	80K bytes
0	1	1	1	160K bytes
1	0	0	0	8K bytes
1	0	0	1	16K bytes
1	0	1	0	32K bytes
1	0	1	1	64K bytes
1	1	0	0	128K bytes
1	1	0	1	256K bytes
1	1	1	0	96K bytes
1	1	1	1	512K bytes

- **ARCH: Architecture Identifier**

ARCH		
Hex	Bin	Architecture
0x19	0001 1001	AT91SAM9xx Series
0x29	0010 1001	AT91SAM9XExx Series
0x34	0011 0100	AT91x34 Series
0x37	0011 0111	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.

32.5.11 Debug Unit Chip ID Extension Register

Name: DBGU_EXID

Address: 0xFFFFEE44

Access: 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.

32.5.12 Debug Unit Force NTRST Register

Name: DBGU_FNR

Address: 0xFFFFEE48

Access: 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.

33. Parallel Input/Output Controller (PIO)

33.1 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.

33.2 Block Diagram

Figure 33-1. Block Diagram

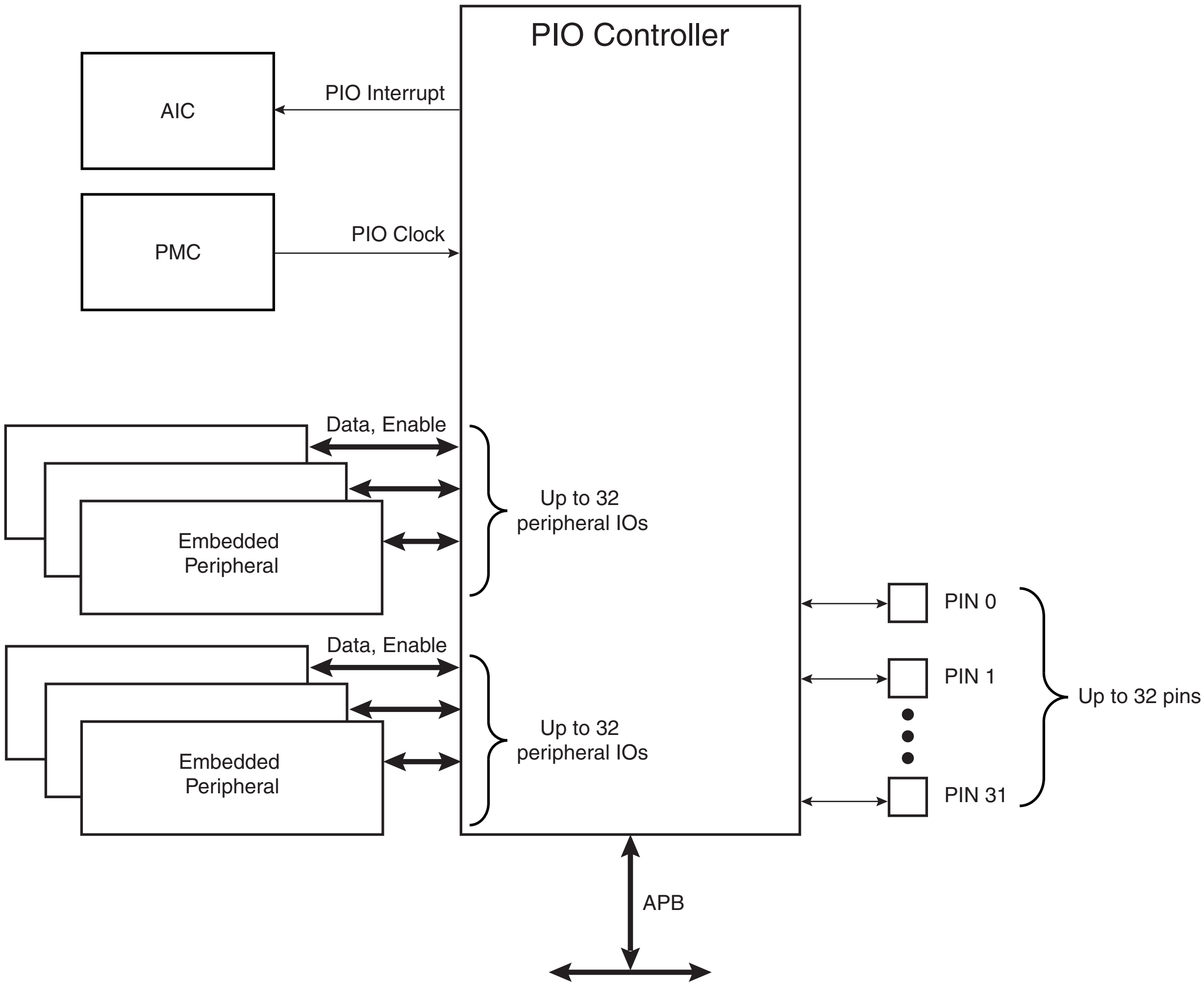
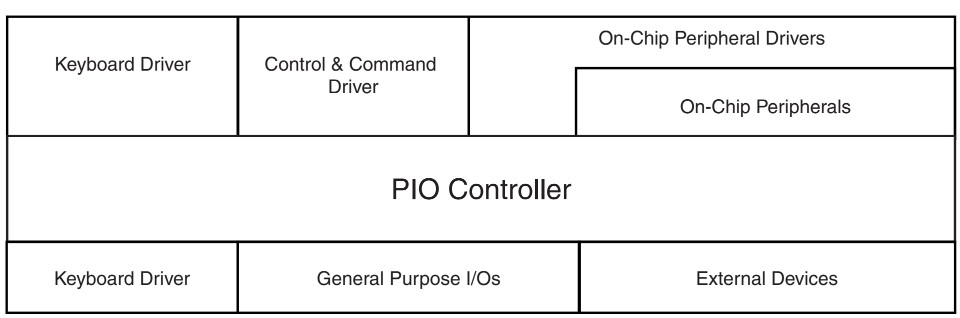


Figure 33-2. Application Block Diagram



33.3 Product Dependencies

33.3.1 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.

33.3.2 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.

33.3.3 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.

33.3.4 Interrupt Generation

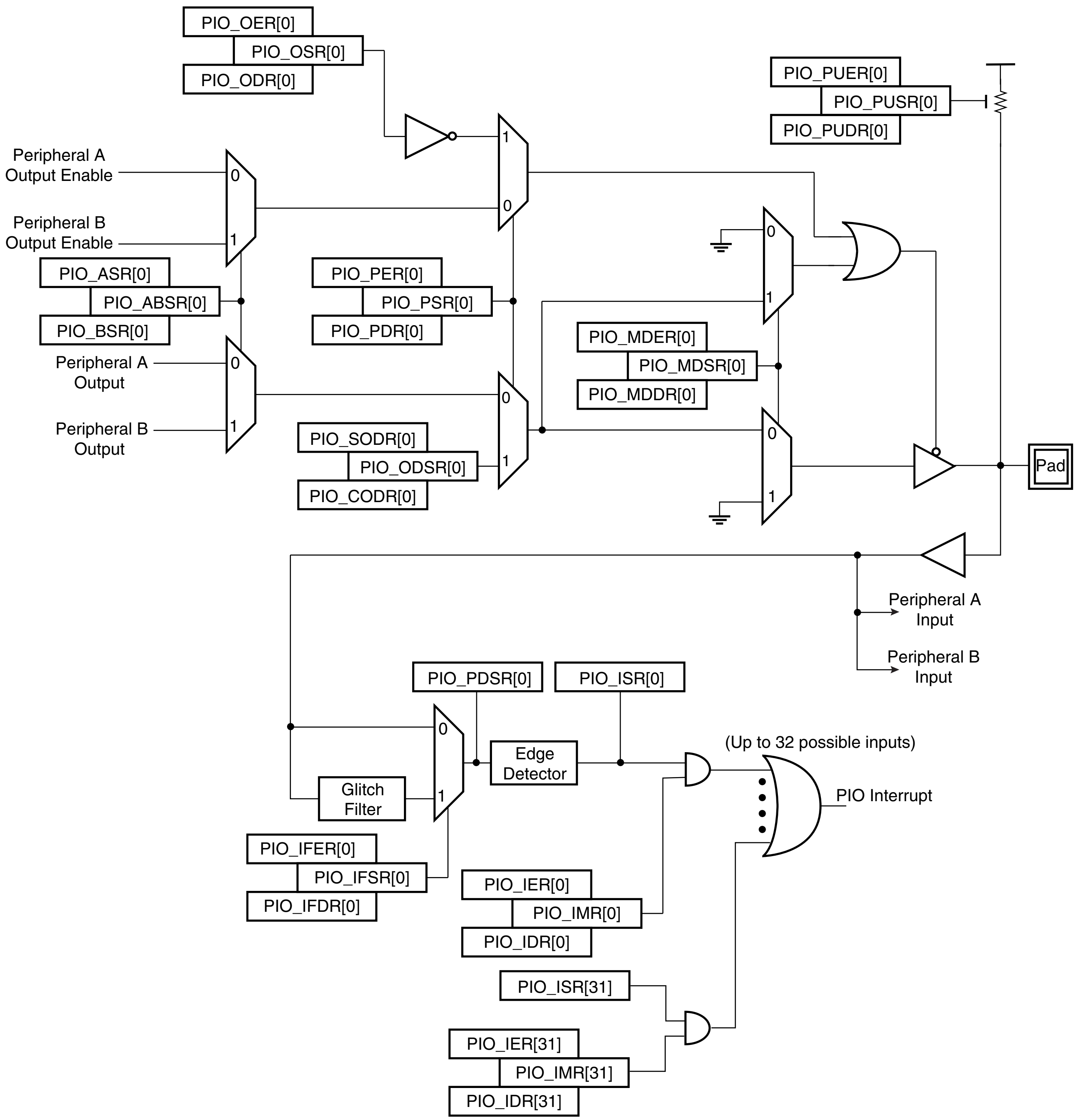
For interrupt handling, the PIO Controllers are considered as user peripherals. This means that the PIO Controller interrupt lines are connected among the interrupt sources 2 to 31. Refer to the PIO Controller peripheral identifier in the product description to identify the interrupt sources dedicated to the PIO Controllers.

The PIO Controller interrupt can be generated only if the PIO Controller clock is enabled.

33.4 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 33-3. In this description each signal shown represents but one of up to 32 possible indexes.

Figure 33-3. I/O Line Control Logic



33.4.1 Pull-up Resistor Control

Each I/O line is designed with an embedded pull-up resistor. The pull-up resistor can be enabled or disabled by writing respectively PIO_PUER (Pull-up Enable Register) and PIO_PUDR (Pull-up Disable Resistor). Writing in these registers results in setting or clearing the corresponding bit in PIO_PUSR (Pull-up Status Register). Reading a 1 in PIO_PUSR means the pull-up is disabled and reading a 0 means the pull-up is enabled.

Control of the pull-up resistor is possible regardless of the configuration of the I/O line.

After reset, all of the pull-ups are enabled, i.e. PIO_PUSR resets at the value 0x0.

33.4.2 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.

33.4.3 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.

33.4.4 Output Control

When the I/0 line is assigned to a peripheral function, i.e. the corresponding bit in PIO_PSR is at 0, the drive of the I/O line is controlled by the peripheral. Peripheral A or B, depending on the value in PIO_ABSR, determines whether the pin is driven or not.

When the I/O line is controlled by the PIO controller, the pin can be configured to be driven. This is done by writing PIO_OER (Output Enable Register) and PIO_ODR (Output Disable Register).

The results of these write operations are detected in PIO_OSR (Output Status Register). When a bit in this register is at 0, the corresponding I/O line is used as an input only. When the bit is at 1, the corresponding I/O line is driven by the PIO controller.

The level driven on an I/O line can be determined by writing in PIO_SODR (Set Output Data Register) and PIO_CODR (Clear Output Data Register). These write operations respectively set and clear PIO_ODSR (Output Data Status Register), which represents the data driven on the I/O lines. Writing in PIO_OER and PIO_ODR manages PIO_OSR whether the pin is configured to be controlled by the PIO controller or assigned to a peripheral function. This enables configuration of the I/O line prior to setting it to be managed by the PIO Controller.

Similarly, writing in PIO_SODR and PIO_CODR effects PIO_ODSR. This is important as it defines the first level driven on the I/O line.

33.4.5 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.

33.4.6 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.

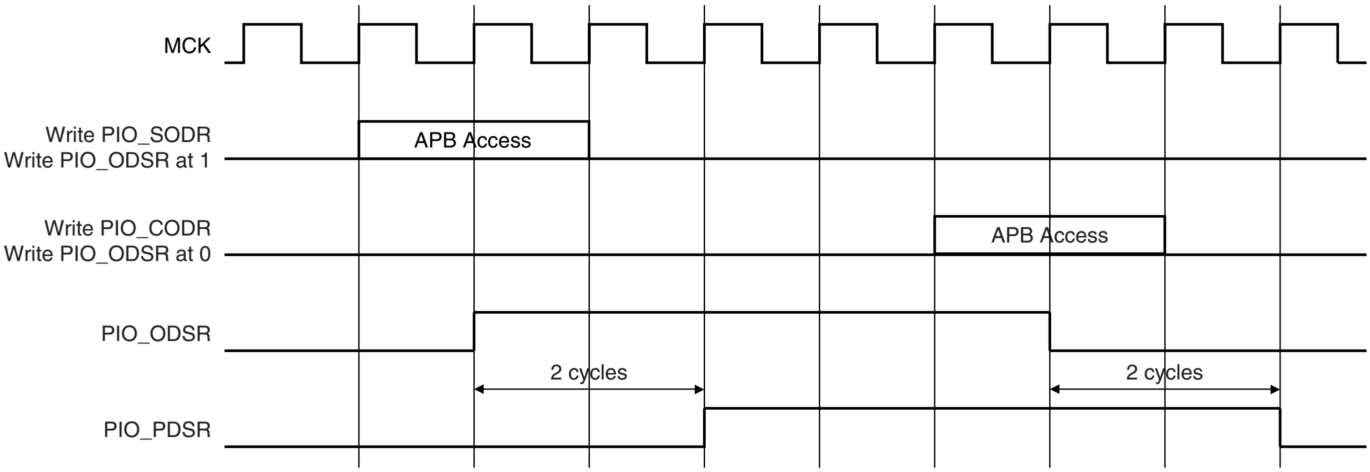
The Multi Drive feature is controlled by PIO_MDER (Multi-driver Enable Register) and PIO_MDDR (Multi-driver Disable Register). The Multi Drive can be selected whether the I/O line is controlled by the PIO controller or assigned to a peripheral function. PIO_MDSR (Multi-driver Status Register) indicates 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.

33.4.7 Output Line Timings

Figure 33-4 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 33-4 also shows when the feedback in PIO_PDSR is available.

Figure 33-4. Output Line Timings



33.4.8 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.

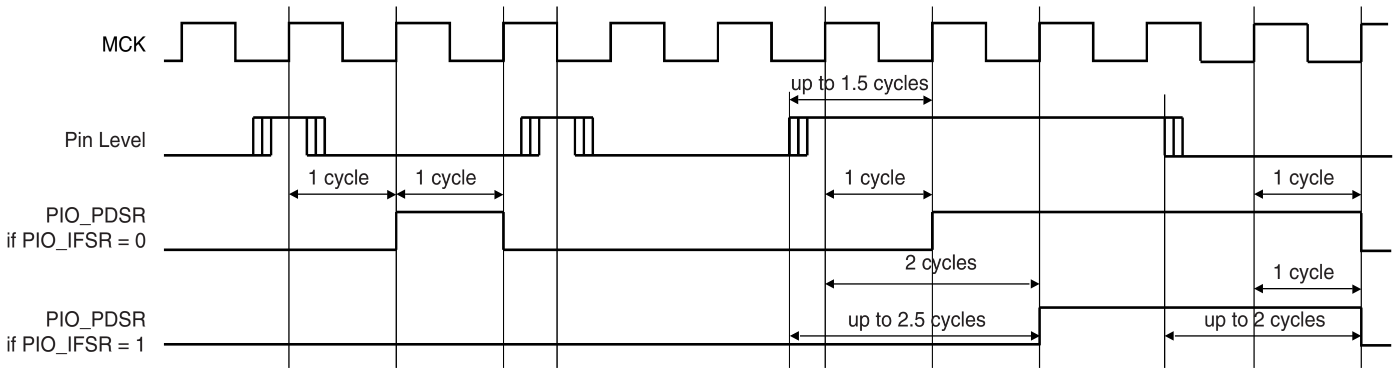
33.4.9 Input Glitch Filtering

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 33-5.

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 33-5. Input Glitch Filter Timing



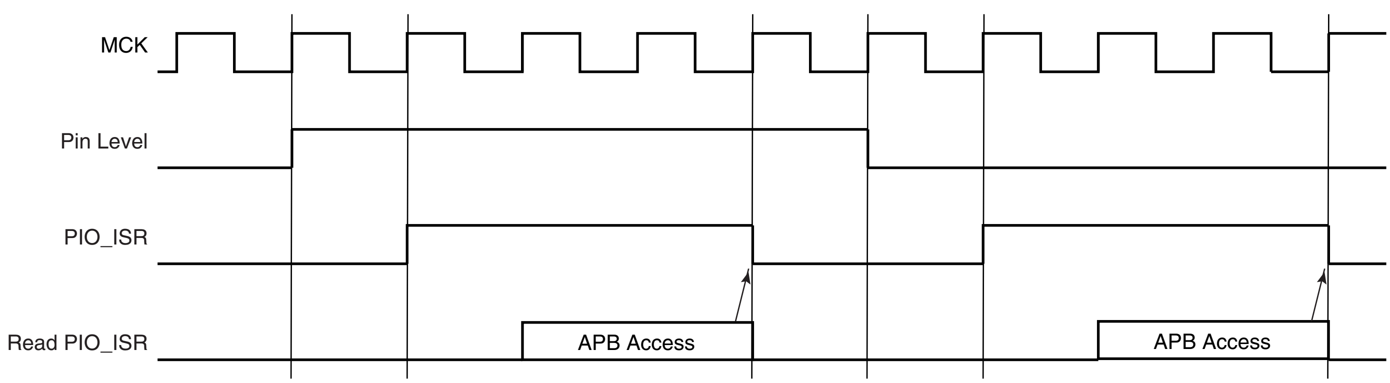
33.4.10 Input Change Interrupt

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 33-6. Input Change Interrupt Timings



33.5 I/O Lines Programming Example

The programing example as shown in Table 33-1 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 33-1. 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

33.6 Parallel Input/Output Controller (PIO) User Interface

Each I/O line controlled by the PIO Controller is associated with a bit in each of the PIO Controller User Interface registers. Each register is 32 bits wide. If a parallel I/O line is not defined, writing to the corresponding bits has no effect. Undefined bits read zero. If the I/O line is not multiplexed with any peripheral, the I/O line is controlled by the PIO Controller and PIO_PSR returns 1 systematically.

Table 33-2. Register Mapping

Offset	Register	Name	Access	Reset
0x0000	PIO Enable Register	PIO_PER	Write-only	–
0x0004	PIO Disable Register	PIO_PDR	Write-only	–
0x0008	PIO Status Register	PIO_PSR	Read-only	(1)
0x000C	Reserved			
0x0010	Output Enable Register	PIO_OER	Write-only	–
0x0014	Output Disable Register	PIO_ODR	Write-only	–
0x0018	Output Status Register	PIO_OSR	Read-only	0x0000 0000
0x001C	Reserved			
0x0020	Glitch Input Filter Enable Register	PIO_IFER	Write-only	–
0x0024	Glitch Input Filter Disable Register	PIO_IFDR	Write-only	–
0x0028	Glitch Input Filter Status Register	PIO_IFSR	Read-only	0x0000 0000
0x002C	Reserved			
0x0030	Set Output Data Register	PIO_SODR	Write-only	–
0x0034	Clear Output Data Register	PIO_CODR	Write-only	
0x0038	Output Data Status Register	PIO_ODSR	Read-only or[2] Read/Write	–
0x003C	Pin Data Status Register	PIO_PDSR	Read-only	(3)
0x0040	Interrupt Enable Register	PIO_IER	Write-only	–
0x0044	Interrupt Disable Register	PIO_IDR	Write-only	–
0x0048	Interrupt Mask Register	PIO_IMR	Read-only	0x00000000
0x004C	Interrupt Status Register[4]	PIO_ISR	Read-only	0x00000000
0x0050	Multi-driver Enable Register	PIO_MDER	Write-only	–
0x0054	Multi-driver Disable Register	PIO_MDDR	Write-only	–
0x0058	Multi-driver Status Register	PIO_MDSR	Read-only	0x00000000
0x005C	Reserved			
0x0060	Pull-up Disable Register	PIO_PUDR	Write-only	–
0x0064	Pull-up Enable Register	PIO_PUER	Write-only	–
0x0068	Pad Pull-up Status Register	PIO_PUSR	Read-only	0x00000000
0x006C	Reserved			

Table 33-2. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x0070	Peripheral A Select Register[5]	PIO_ASR	Write-only	–
0x0074	Peripheral B Select Register[5]	PIO_BSR	Write-only	–
0x0078	AB Status Register[5]	PIO_ABSR	Read-only	0x00000000
0x007C to 0x009C	Reserved			
0x00A0	Output Write Enable	PIO_OWER	Write-only	–
0x00A4	Output Write Disable	PIO_OWDR	Write-only	–
0x00A8	Output Write Status Register	PIO_OWSR	Read-only	0x00000000
0x00AC	Reserved			

Notes: 1. Reset value of PIO_PSR depends on the product implementation.

2. PIO_ODSR is Read-only or Read/Write depending on PIO_OWSR I/O lines.
3. Reset value of PIO_PDSR depends on the level of the I/O lines. Reading the I/O line levels requires the clock of the PIO Controller to be enabled, otherwise PIO_PDSR reads the levels present on the I/O line at the time the clock was disabled.
4. PIO_ISR is reset at 0x0. However, the first read of the register may read a different value as input changes may have occurred.
5. Only this set of registers clears the status by writing 1 in the first register and sets the status by writing 1 in the second register.

33.6.1 PIO Controller PIO Enable Register

Name: PIO_PER

Addresses: 0xFFFFF200 (PIOA), 0xFFFFF400 (PIOB), 0xFFFFF600 (PIOC), 0xFFFFF800 (PIOD)

Access: 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).

33.6.2 PIO Controller PIO Disable Register

Name: PIO_PDR

Addresses: 0xFFFFF204 (PIOA), 0xFFFFF404 (PIOB), 0xFFFFF604 (PIOC), 0xFFFFF804 (PIOD)

Access: 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).

33.6.3 PIO Controller PIO Status Register

Name: PIO_PSR

Addresses: 0xFFFFF208 (PIOA), 0xFFFFF408 (PIOB), 0xFFFFF608 (PIOC), 0xFFFFF808 (PIOD)

Access: 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).

33.6.4 PIO Controller Output Enable Register

Name: PIO_OER

Addresses: 0xFFFFF210 (PIOA), 0xFFFFF410 (PIOB), 0xFFFFF610 (PIOC), 0xFFFFF810 (PIOD)

Access: 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.

33.6.5 PIO Controller Output Disable Register

Name: PIO_ODR

Addresses: 0xFFFFF214 (PIOA), 0xFFFFF414 (PIOB), 0xFFFFF614 (PIOC), 0xFFFFF814 (PIOD)

Access: 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.

33.6.6 PIO Controller Output Status Register

Name: PIO_OSR

Addresses: 0xFFFFF218 (PIOA), 0xFFFFF418 (PIOB), 0xFFFFF618 (PIOC), 0xFFFFF818 (PIOD)

Access: 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.

33.6.7 PIO Controller Input Filter Enable Register

Name: PIO_IFER

Addresses: 0xFFFFF220 (PIOA), 0xFFFFF420 (PIOB), 0xFFFFF620 (PIOC), 0xFFFFF820 (PIOD)

Access: 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.

33.6.8 PIO Controller Input Filter Disable Register

Name: PIO_IFDR

Addresses: 0xFFFFF224 (PIOA), 0xFFFFF424 (PIOB), 0xFFFFF624 (PIOC), 0xFFFFF824 (PIOD)

Access: 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.

33.6.9 PIO Controller Input Filter Status Register

Name: PIO_IFSR

Addresses: 0xFFFFF228 (PIOA), 0xFFFFF428 (PIOB), 0xFFFFF628 (PIOC), 0xFFFFF828 (PIOD)

Access: Read-only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24

23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16

15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: Input Filer Status**

0 = The input glitch filter is disabled on the I/O line.

1 = The input glitch filter is enabled on the I/O line.

33.6.10 PIO Controller Set Output Data Register

Name: PIO_SODR

Addresses: 0xFFFFF230 (PIOA), 0xFFFFF430 (PIOB), 0xFFFFF630 (PIOC), 0xFFFFF830 (PIOD)

Access: 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.

33.6.11 PIO Controller Clear Output Data Register

Name: PIO_CODR

Addresses: 0xFFFFF234 (PIOA), 0xFFFFF434 (PIOB), 0xFFFFF634 (PIOC), 0xFFFFF834 (PIOD)

Access: 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.

33.6.12 PIO Controller Output Data Status Register

Name: PIO_ODSR

Addresses: 0xFFFFF238 (PIOA), 0xFFFFF438 (PIOB), 0xFFFFF638 (PIOC), 0xFFFFF838 (PIOD)

Access: 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.

33.6.13 PIO Controller Pin Data Status Register

Name: PIO_PDSR

Addresses: 0xFFFFF23C (PIOA), 0xFFFFF43C (PIOB), 0xFFFFF63C (PIOC), 0xFFFFF83C (PIOD)

Access: 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.

33.6.14 PIO Controller Interrupt Enable Register

Name: PIO_IER

Addresses: 0xFFFFF240 (PIOA), 0xFFFFF440 (PIOB), 0xFFFFF640 (PIOC), 0xFFFFF840 (PIOD)

Access: 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.

33.6.15 PIO Controller Interrupt Disable Register

Name: PIO_IDR

Addresses: 0xFFFFF244 (PIOA), 0xFFFFF444 (PIOB), 0xFFFFF644 (PIOC), 0xFFFFF844 (PIOD)

Access: 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.

33.6.16 PIO Controller Interrupt Mask Register

Name: PIO_IMR

Addresses: 0xFFFFF248 (PIOA), 0xFFFFF448 (PIOB), 0xFFFFF648 (PIOC), 0xFFFFF848 (PIOD)

Access: 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.

33.6.17 PIO Controller Interrupt Status Register

Name: PIO_ISR

Addresses: 0xFFFFF24C (PIOA), 0xFFFFF44C (PIOB), 0xFFFFF64C (PIOC), 0xFFFFF84C (PIOD)

Access: 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.

33.6.18 PIO Multi-driver Enable Register

Name: PIO_MDER

Addresses: 0xFFFFF250 (PIOA), 0xFFFFF450 (PIOB), 0xFFFFF650 (PIOC), 0xFFFFF850 (PIOD)

Access: 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.

33.6.19 PIO Multi-driver Disable Register

Name: PIO_MDDR

Addresses: 0xFFFFF254 (PIOA), 0xFFFFF454 (PIOB), 0xFFFFF654 (PIOC), 0xFFFFF854 (PIOD)

Access: 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.

33.6.20 PIO Multi-driver Status Register

Name: PIO_MDSR

Addresses: 0xFFFFF258 (PIOA), 0xFFFFF458 (PIOB), 0xFFFFF658 (PIOC), 0xFFFFF858 (PIOD)

Access: 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.

33.6.21 PIO Pull Up Disable Register

Name: PIO_PUDR

Addresses: 0xFFFFF260 (PIOA), 0xFFFFF460 (PIOB), 0xFFFFF660 (PIOC), 0xFFFFF860 (PIOD)

Access: 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.

33.6.22 PIO Pull Up Enable Register

Name: PIO_PUER

Addresses: 0xFFFFF264 (PIOA), 0xFFFFF464 (PIOB), 0xFFFFF664 (PIOC), 0xFFFFF864 (PIOD)

Access: 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.

33.6.23 PIO Pull Up Status Register

Name: PIO_PUSR

Addresses: 0xFFFFF268 (PIOA), 0xFFFFF468 (PIOB), 0xFFFFF668 (PIOC), 0xFFFFF868 (PIOD)

Access: 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.

33.6.24 PIO Peripheral A Select Register

Name: PIO_ASR

Addresses: 0xFFFFF270 (PIOA), 0xFFFFF470 (PIOB), 0xFFFFF670 (PIOC), 0xFFFFF870 (PIOD)

Access: 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.

33.6.25 PIO Peripheral B Select Register

Name: PIO_BSR

Addresses: 0xFFFFF274 (PIOA), 0xFFFFF474 (PIOB), 0xFFFFF674 (PIOC), 0xFFFFF874 (PIOD)

Access: 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.

33.6.26 PIO Peripheral A B Status Register

Name: PIO_ABSR

Addresses: 0xFFFFF278 (PIOA), 0xFFFFF478 (PIOB), 0xFFFFF678 (PIOC), 0xFFFFF878 (PIOD)

Access: 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.

33.6.27 PIO Output Write Enable Register

Name: PIO_OWER

Addresses: 0xFFFFF2A0 (PIOA), 0xFFFFF4A0 (PIOB), 0xFFFFF6A0 (PIOC), 0xFFFFF8A0 (PIOD)

Access: 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.

33.6.28 PIO Output Write Disable Register

Name: PIO_OWDR

Addresses: 0xFFFFF2A4 (PIOA), 0xFFFFF4A4 (PIOB), 0xFFFFF6A4 (PIOC), 0xFFFFF8A4 (PIOD)

Access: 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.

33.6.29 PIO Output Write Status Register

Name: PIO_OWSR

Addresses: 0xFFFFF2A8 (PIOA), 0xFFFFF4A8 (PIOB), 0xFFFFF6A8 (PIOC), 0xFFFFF8A8 (PIOD)

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.

34. Serial Peripheral Interface (SPI)

34.1 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.

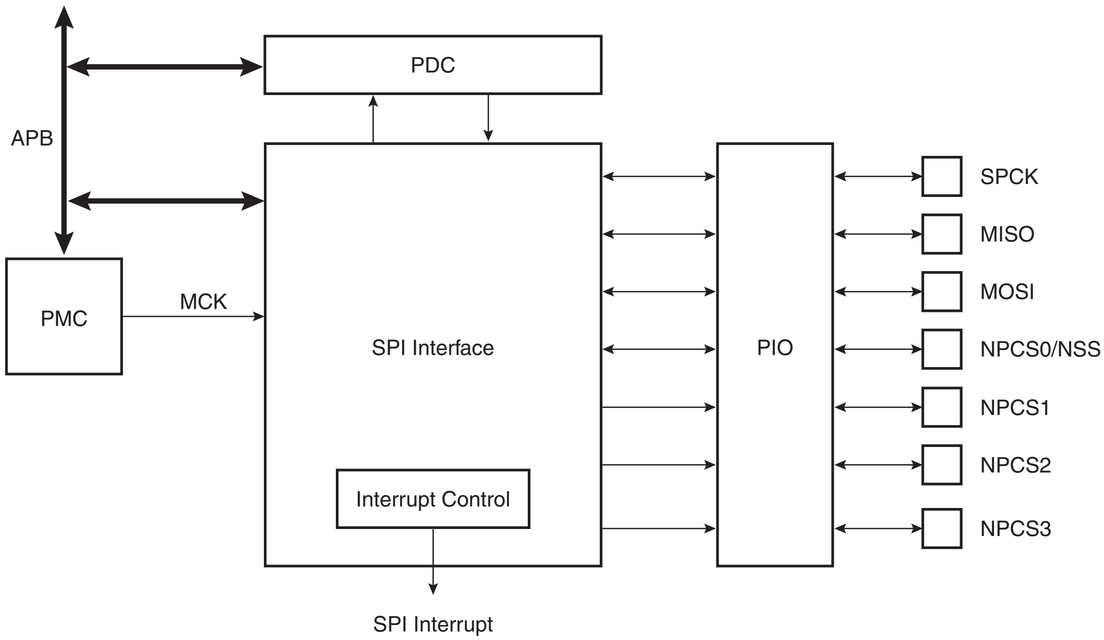
A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

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.

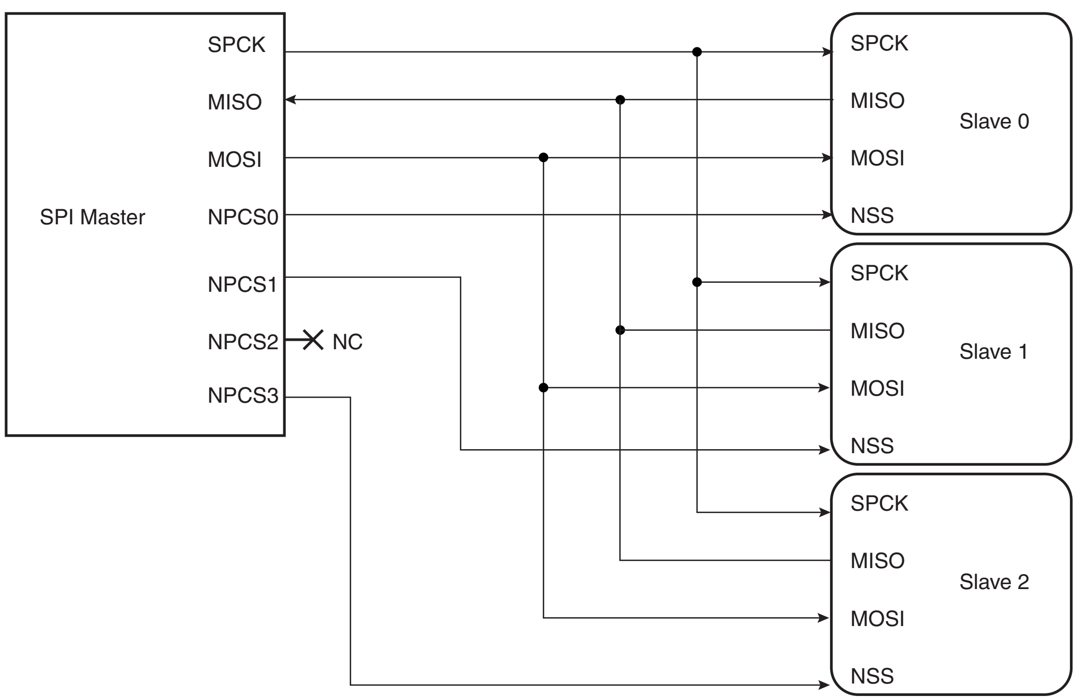
34.2 Block Diagram

Figure 34-1. Block Diagram



34.3 Application Block Diagram

Figure 34-2. Application Block Diagram: Single Master/Multiple Slave Implementation



34.4 Signal Description

Table 34-1. 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

34.5 Product Dependencies

34.5.1 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.

Table 34-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
SPI0	SPI0_MISO	PA0	B
SPI0	SPI0_MOSI	PA1	B
SPI0	SPI0_NPCS0	PA5	B
SPI0	SPI0_NPCS1	PA3	B
SPI0	SPI0_NPCS2	PA4	B
SPI0	SPI0_NPCS2	PD0	B
SPI0	SPI0_NPCS3	PA28	A
SPI0	SPI0_NPCS3	PD1	B
SPI0	SPI0_SPCK	PA2	B
SPI1	SPI1_MISO	PB12	A
SPI1	SPI1_MOSI	PB13	A
SPI1	SPI1_NPCS0	PB15	A
SPI1	SPI1_NPCS1	PB16	A
SPI1	SPI1_NPCS2	PB17	A
SPI1	SPI1_NPCS2	PD2	B
SPI1	SPI1_NPCS3	PB18	A
SPI1	SPI1_NPCS3	PD3	B
SPI1	SPI1_SPCK	PB14	A

34.5.2 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.

34.5.3 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.

Table 34-3. Peripheral IDs

Instance	ID
SPI0	15
SPI1	16

34.6 Functional Description

34.6.1 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.

34.6.2 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 34-4 shows the four modes and corresponding parameter settings.

Table 34-4. SPI Bus Protocol Mode

SPI Mode	CPOL	NCPHA
0	0	1
1	0	0
2	1	1
3	1	0

Figure 34-3 and Figure 34-4 show examples of data transfers.

Figure 34-3. SPI Transfer Format (NCPHA = 1, 8 bits per transfer)

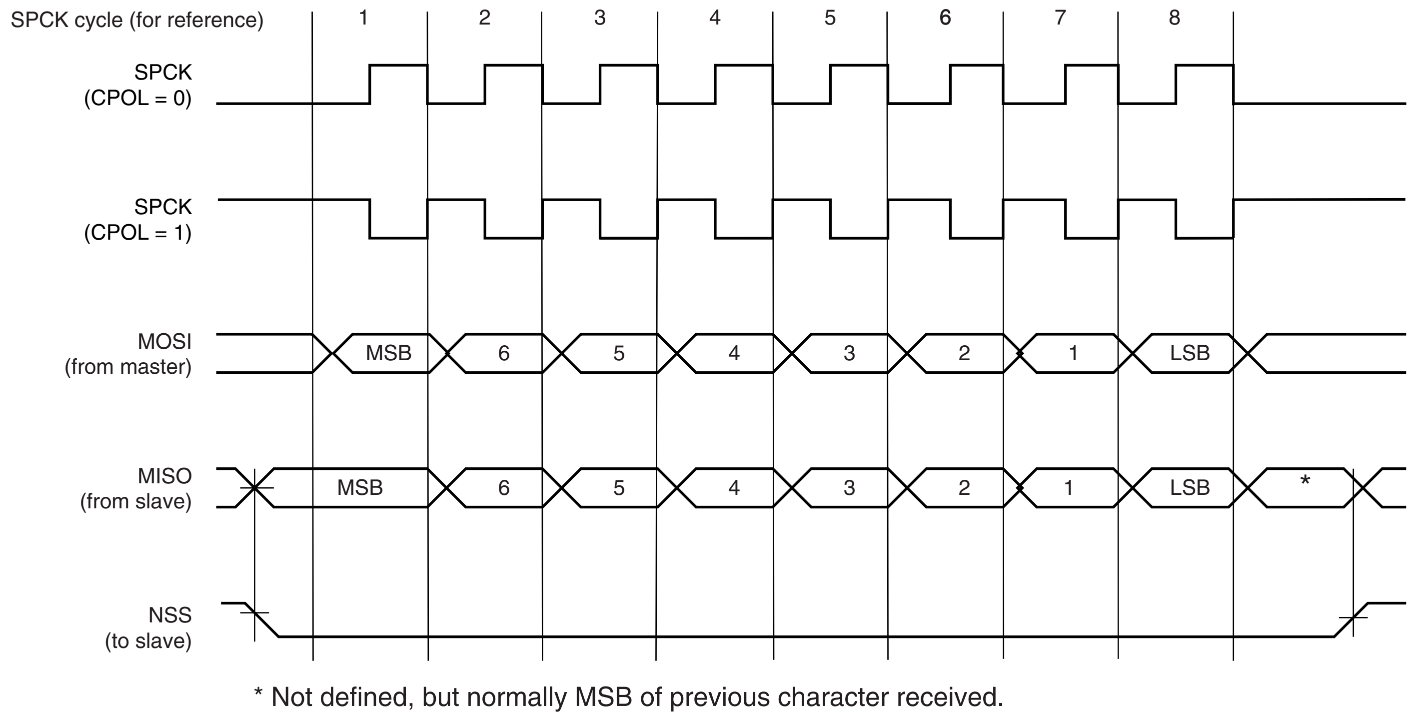
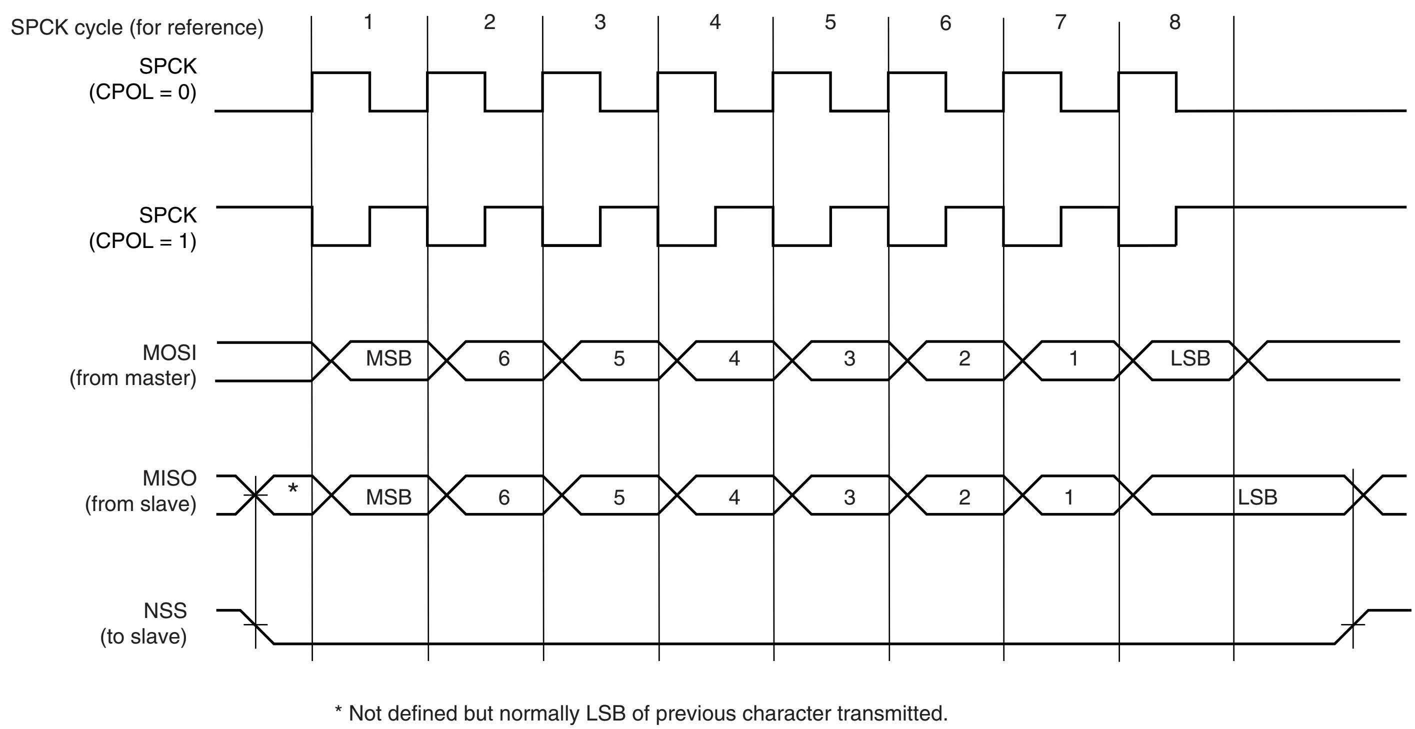


Figure 34-4. SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



34.6.3 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.

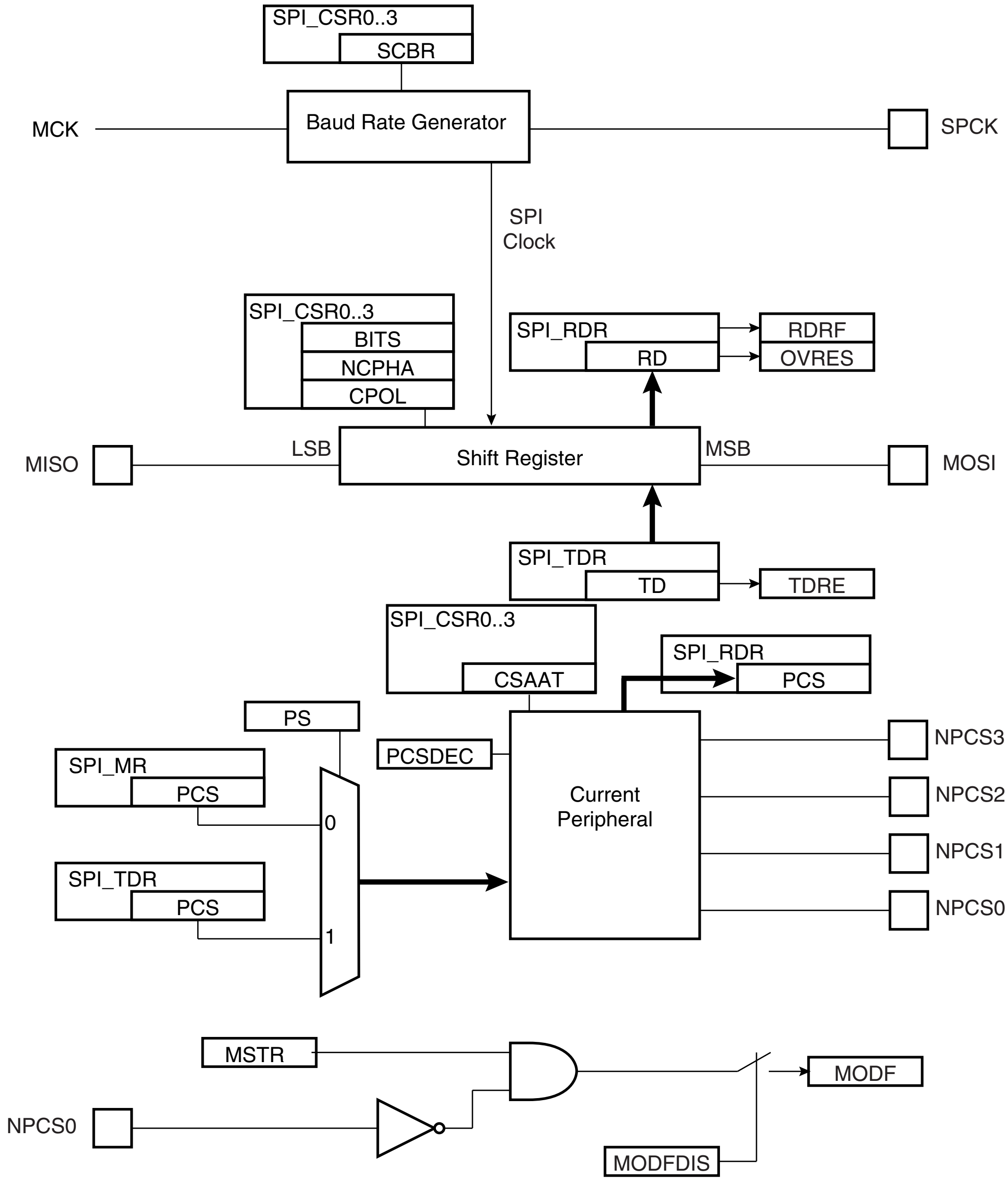
The transfer of received data from the Shift Register in SPI_RDR is indicated by the RDRF bit (Receive Data Register Full) in the Status Register (SPI_SR). When the received data is read, the RDRF bit is cleared.

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 34-5, shows a block diagram of the SPI when operating in Master Mode. Figure 34-6 on page 511 shows a flow chart describing how transfers are handled.

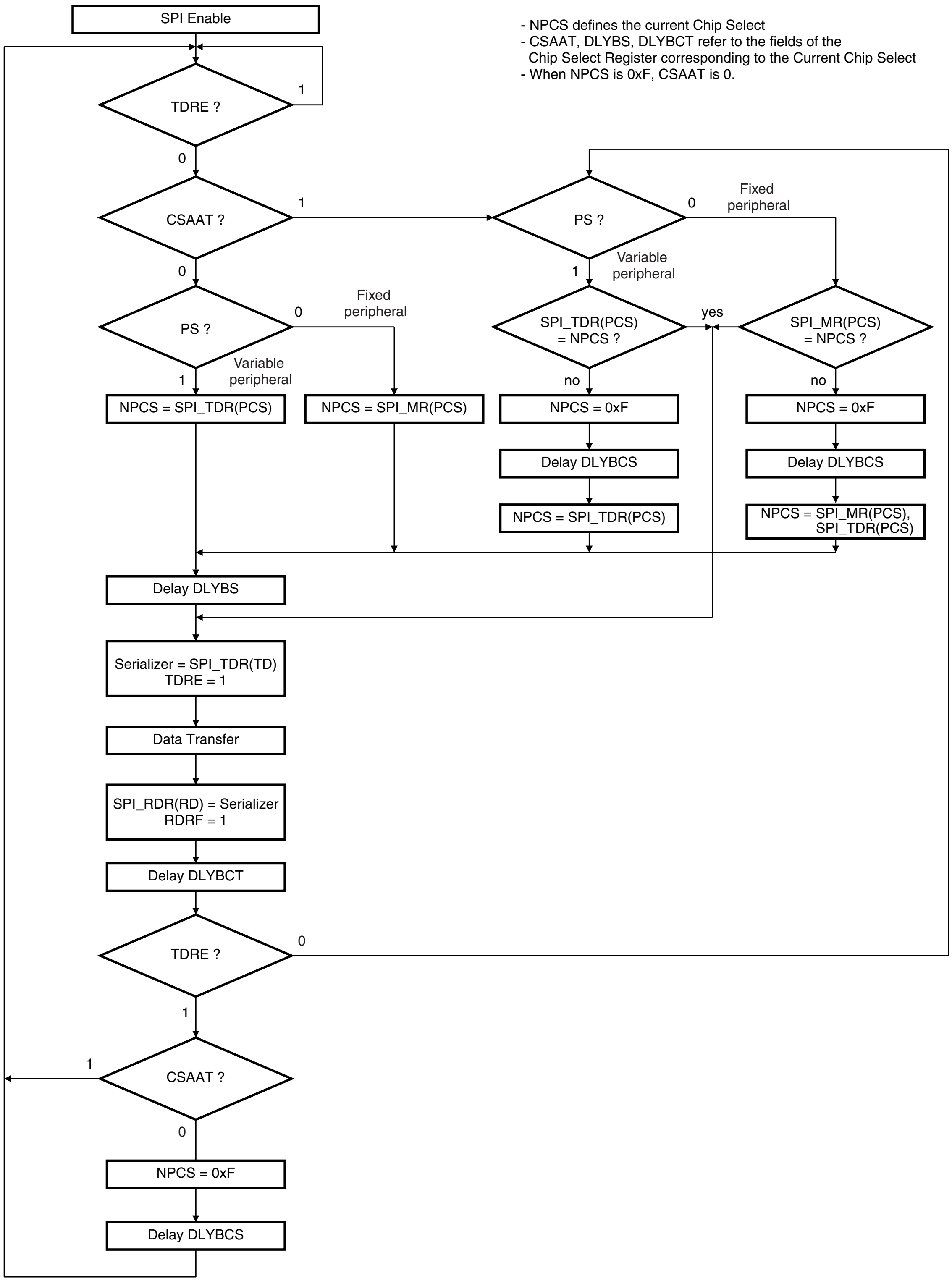
34.6.3.1 Master Mode Block Diagram

Figure 34-5. Master Mode Block Diagram



34.6.3.2 Master Mode Flow Diagram

Figure 34-6. Master Mode Flow Diagram



34.6.3.3 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.

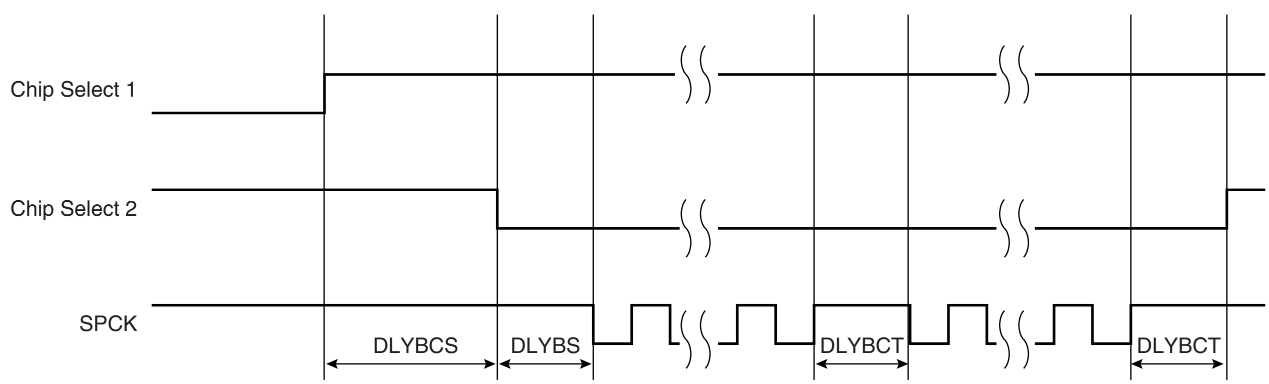
34.6.3.4 Transfer Delays

Figure 34-7 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 34-7. Programmable Delays



34.6.3.5 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 Lisps and the PCS and LASTXFER fields in the MSBs, however the SPI still controls the number of bits (8 to16) to be transferred through MISO and MOSI lines with the chip select configuration registers. This is not the optimal means in term of memory size for the buffers, but it provides a very effective means to exchange data with several peripherals without any intervention of the processor.

34.6.3.6 Peripheral Chip Select Decoding

The user can program the SPI to operate with up to 15 peripherals by decoding the four Chip Select lines, NPCS0 to NPCS3 with an external logic. This can be enabled by writing the PCSDEC bit at 1 in the Mode Register (SPI_MR).

When operating without decoding, the SPI makes sure that in any case only one chip select line is activated, i.e. driven low at a time. If two bits are defined low in a PCS field, only the lowest numbered chip select is driven low.

When operating with decoding, the SPI directly outputs the value defined by the PCS field of either the Mode Register or the Transmit Data Register (depending on PS).

As the SPI sets a default value of 0xF on the chip select lines (i.e. all chip select lines at 1) when not processing any transfer, only 15 peripherals can be decoded.

The SPI has only four Chip Select Registers, not 15. As a result, when decoding is activated, each chip select defines the characteristics of up to four peripherals. As an example, SPI_CRS0 defines the characteristics of the externally decoded peripherals 0 to 3, corresponding to the PCS values 0x0 to 0x3. Thus, the user has to make sure to connect compatible peripherals on the decoded chip select lines 0 to 3, 4 to 7, 8 to 11 and 12 to 14.

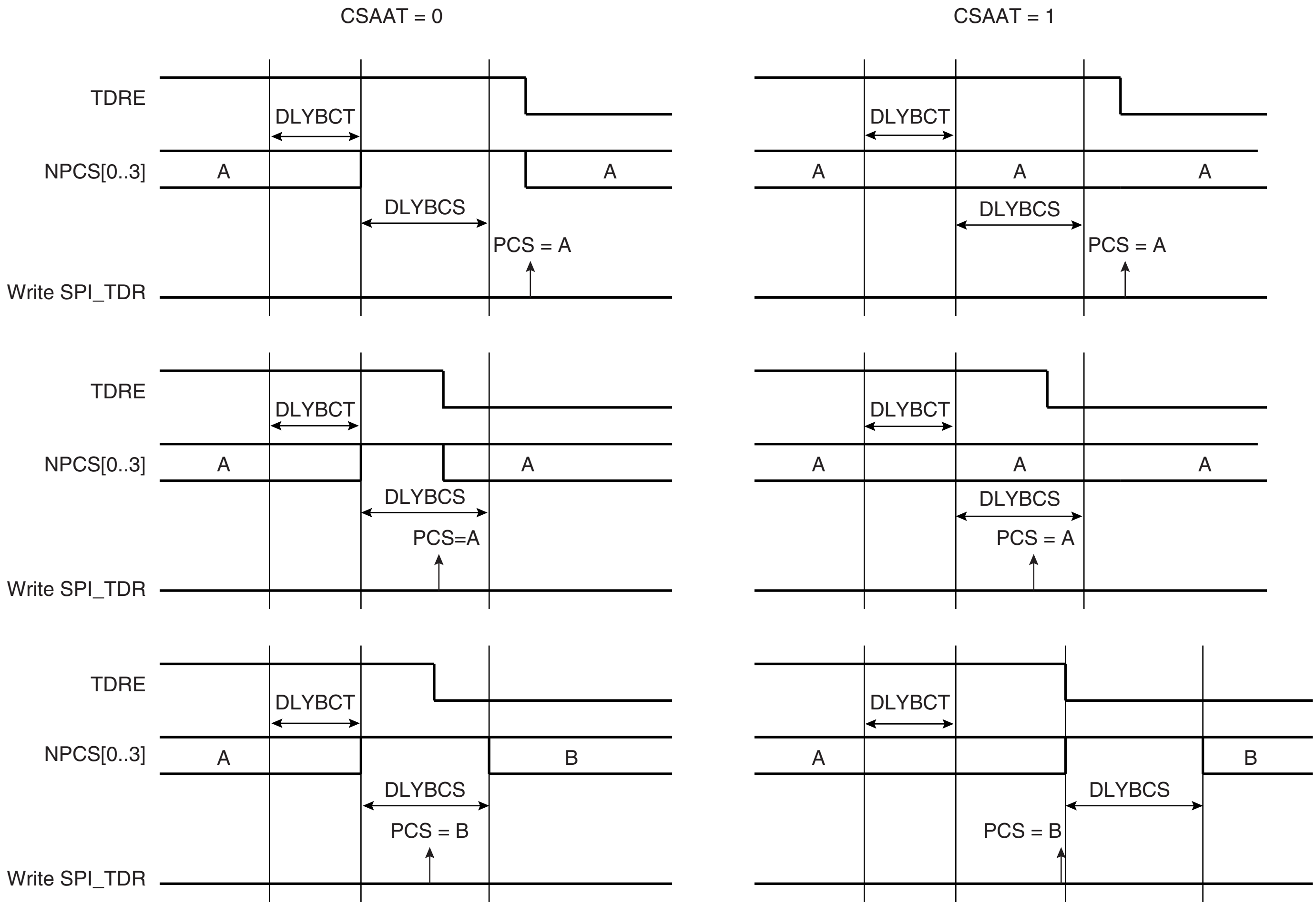
34.6.3.7 Peripheral Deselection

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 34-8 shows different peripheral deselection cases and the effect of the CSAAT bit.

Figure 34-8. Peripheral Deselection



34.6.3.8 *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).

34.6.4 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.

(For more information on BITS field, see also, the (Note:) below the register table; Section 34.7.9 “SPI Chip Select Register” on page 527.)

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.

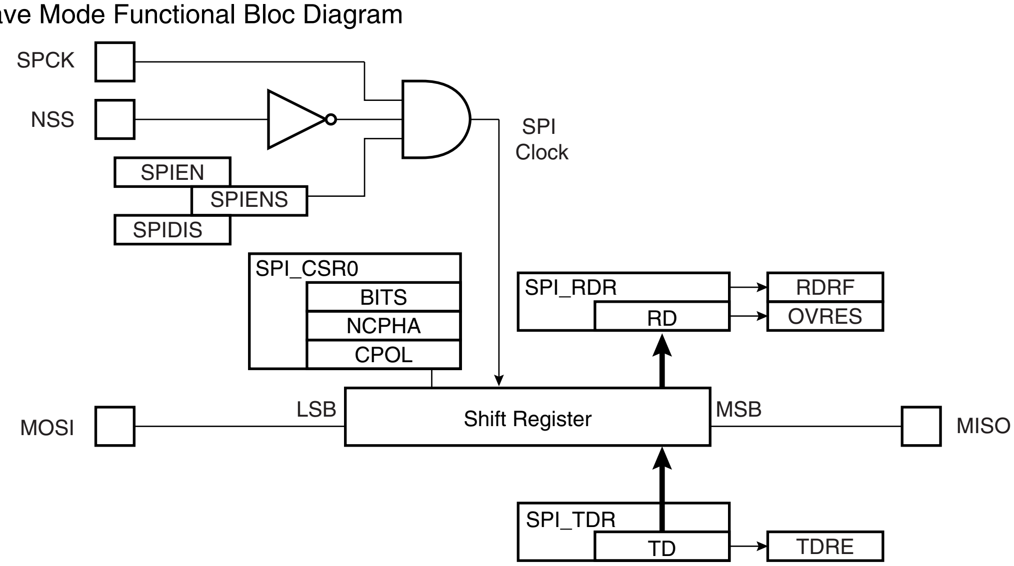
When a transfer starts, the data shifted out is the data present in the Shift Register. If no data has been written in the Transmit Data Register (SPI_TDR), the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift Register resets 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 34-9 shows a block diagram of the SPI when operating in Slave Mode.

Figure 34-9. Slave Mode Functional Bloc Diagram



34.7 Serial Peripheral Interface (SPI) User Interface

Table 34-5. Register Mapping

Offset	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	–	–	–
0x100 - 0x124	Reserved for the PDC			

34.7.1 SPI Control Register

Name: SPI_CR

Addresses: 0xFFFA4000 (0), 0xFFFA8000 (1)

Access: 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.

34.7.2 SPI Mode Register

Name: SPI_MR

Addresses: 0xFFFA4004 (0), 0xFFFA8004 (1)

Access: 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 = xxx0NPCS[3:0] = 1110

PCS = xx01NPCS[3:0] = 1101

PCS = x011NPCS[3:0] = 1011

PCS = 0111NPCS[3:0] = 0111

PCS = 1111forbidden (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:

$$\text{Delay Between Chip Selects} = \frac{DLYBCS}{MCK}$$

34.7.3 SPI Receive Data Register

Name: SPI_RDR

Addresses: 0xFFFA4008 (0), 0xFFFA8008 (1)

Access: 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.

34.7.4 SPI Transmit Data Register

Name: SPI_TDR

Addresses: 0xFFFA400C (0), 0xFFFA800C (1)

Access: 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 = xxx0NPCS[3:0] = 1110

PCS = xx01NPCS[3:0] = 1101

PCS = x011NPCS[3:0] = 1011

PCS = 0111NPCS[3:0] = 0111

PCS = 1111forbidden (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).

34.7.5 SPI Status Register

Name: SPI_SR

Addresses: 0xFFFA4010 (0), 0xFFFA8010 (1)

Access: Read-only

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

- **RDRF: Receive Data Register Full**

0 = No data has been received since the last read of SPI_RDR

1 = Data has been received and the received data has been transferred from the serializer to SPI_RDR since the last read of SPI_RDR.

- **TDRE: Transmit Data Register Empty**

0 = Data has been written to SPI_TDR and not yet transferred to the serializer.

1 = The last data written in the Transmit Data Register has been transferred to the serializer.

TDRE equals zero when the SPI is disabled or at reset. The SPI enable command sets this bit to one.

- **MODF: Mode Fault Error**

0 = No Mode Fault has been detected since the last read of SPI_SR.

1 = A Mode Fault occurred since the last read of the SPI_SR.

- **OVRES: Overrun Error Status**

0 = No overrun has been detected since the last read of SPI_SR.

1 = An overrun has occurred since the last read of SPI_SR.

An overrun occurs when SPI_RDR is loaded at least twice from the serializer since the last read of the SPI_RDR.

- **ENDRX: End of RX buffer**

0 = The Receive Counter Register has not reached 0 since the last write in SPI_RCR[(1)] or SPI_RNCR[(1)].

1 = The Receive Counter Register has reached 0 since the last write in SPI_RCR[(1)] or SPI_RNCR[(1)].

- **ENDTX: End of TX buffer**

0 = The Transmit Counter Register has not reached 0 since the last write in SPI_TCR[(1)] or SPI_TNCR[(1)].

1 = The Transmit Counter Register has reached 0 since the last write in SPI_TCR[(1)] or SPI_TNCR[(1)].

- **RXBUFF: RX Buffer Full**

0 = SPI_RCR[(1)] or SPI_RNCR[(1)] has a value other than 0.

1 = Both SPI_RCR[(1)] and SPI_RNCR[(1)] have a value of 0.

- **TXBUFE: TX Buffer Empty**

0 = SPI_TCR[1] or SPI_TNCR[1] has a value other than 0.

1 = Both SPI_TCR[1] and SPI_TNCR[1] have a value of 0.

- **NSSR: NSS Rising**

0 = No rising edge detected on NSS pin since last read.

1 = A rising edge occurred on NSS pin since last read.

- **TXEMPTY: Transmission Registers Empty**

0 = As soon as data is written in SPI_TDR.

1 = SPI_TDR and internal shifter are empty. If a transfer delay has been defined, TXEMPTY is set after the completion of such delay.

- **SPIENS: SPI Enable Status**

0 = SPI is disabled.

1 = SPI is enabled.

Note: 1. SPI_RCR, SPI_RNCR, SPI_TCR, SPI_TNCR are physically located in the PDC.

34.7.6 SPI Interrupt Enable Register

Name: SPI_IER

Addresses: 0xFFFA4014 (0), 0xFFFA8014 (1)

Access: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR

7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

0 = No effect.

1 = Enables the corresponding interrupt.

- **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**

- **NSSR: NSS Rising Interrupt Enable**

- **TXEMPTY: Transmission Registers Empty Enable**

34.7.7 SPI Interrupt Disable Register

Name: SPI_IDR

Addresses: 0xFFFA4018 (0), 0xFFFA8018 (1)

Access: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR

7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

0 = No effect.

1 = Disables the corresponding interrupt.

- **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**
- **NSSR: NSS Rising Interrupt Disable**
- **TXEMPTY: Transmission Registers Empty Disable**

34.7.8 SPI Interrupt Mask Register

Name: SPI_IMR

Addresses: 0xFFFA401C (0), 0xFFFA801C (1)

Access: Read-only

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

0 = The corresponding interrupt is not enabled.

1 = The corresponding interrupt is enabled.

- **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**
- **NSSR: NSS Rising Interrupt Mask**
- **TXEMPTY: Transmission Registers Empty Mask**

34.7.9 SPI Chip Select Register

Name: SPI_CSR0... SPI_CSR3

Addresses: 0xFFFA4030 (0), 0xFFFA8030 (1)

Access: 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

Note: SPI_CSRx registers must be written even if the user wants to use the defaults. The BITS field will not be updated with the translated value unless the register is written.

- **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** (See the (Note:) below the register table; Section 34.7.9 "SPI Chip Select Register" on page 527.)

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

BITS	Bits Per Transfer
1000	16
1001	Reserved
1010	Reserved
1011	Reserved
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

- **SCBR: Serial Clock Baud Rate**

In Master Mode, the SPI Interface uses a modulus counter to derive the SPCK baud rate from the Master Clock MCK. The Baud rate is selected by writing a value from 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:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{MCK}$$

35. Two-wire Interface (TWI)

35.1 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 I²C 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. 20

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 35-1 lists the compatibility level of the Atmel Two-wire Interface in Master Mode and a full I2C compatible device.

Table 35-1. 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[(1)]	Not Supported
Repeated Start (Sr) Condition	Supported
ACK and NACK Management	Supported
Slope control and input filtering (Fast mode)	Not Supported
Clock stretching	Supported
Multi Master Capability	Supported

Note: 1. START + b000000001 + Ack + Sr

35.2 List of Abbreviations

Table 35-2. Abbreviations

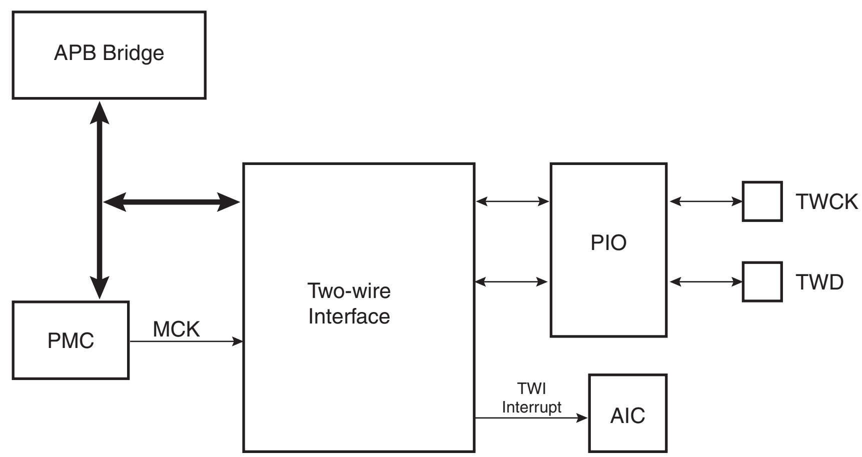
Abbreviation	Description
TWI	Two-wire Interface
A	Acknowledge
NA	Non Acknowledge
P	Stop
S	Start
Sr	Repeated Start
SADR	Slave Address

Abbreviation	Description
ADR	Any address except SADR
R	Read
W	Write

Table 35-2. Abbreviations

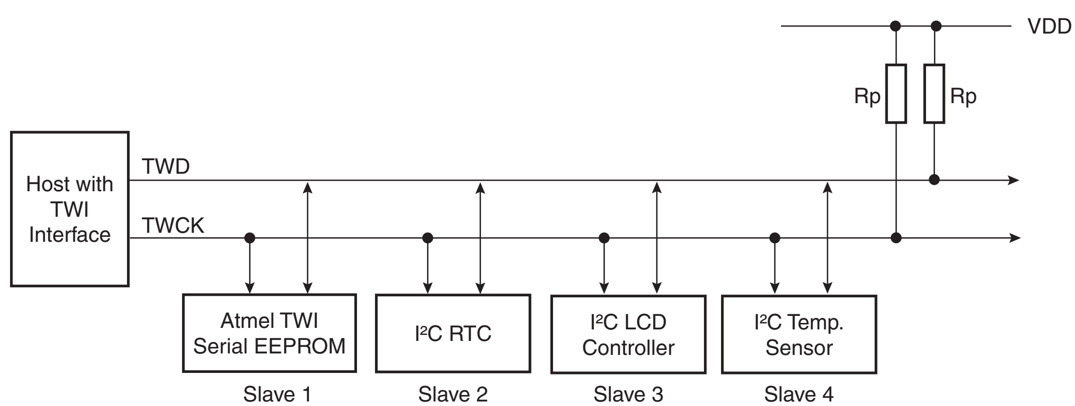
35.3 Block Diagram

Figure 35-1. Block Diagram



35.4 Application Block Diagram

Figure 35-2. Application Block Diagram



Rp: Pull up value as given by the I²C Standard

35.5 I/O Lines Description

I/O Lines Description

Pin Name	Pin Description	Type
TWD	Two-wire Serial Data	Input/Output
TWCK	Two-wire Serial Clock	Input/Output

35.6 Product Dependencies

35.6.1 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 35-2 on page 530). 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 step:

- Program the PIO controller to dedicate TWD and TWCK as peripheral lines.

The user must not program TWD and TWCK as open-drain. It is already done by the hardware.

Table 35-3. I/O Lines

Instance	Signal	I/O Line	Peripheral
TWI	TWCK	PB5	B
TWI	TWD	PB4	B

35.6.2 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.

35.6.3 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.

Table 35-4. Peripheral IDs

Instance	ID
TWI	14

35.7 Functional Description

35.7.1 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 35-4).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 35-3).

- 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 35-3. START and STOP Conditions

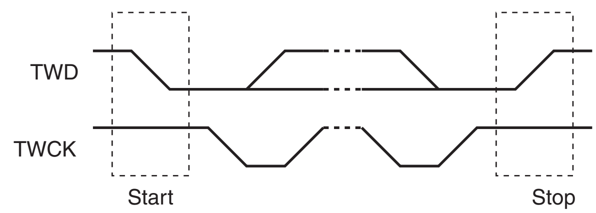
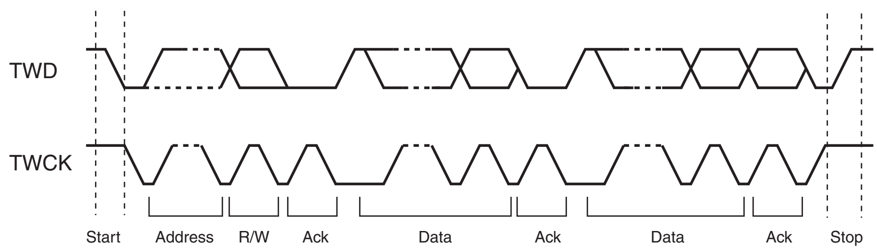


Figure 35-4. Transfer Format



35.7.2 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.

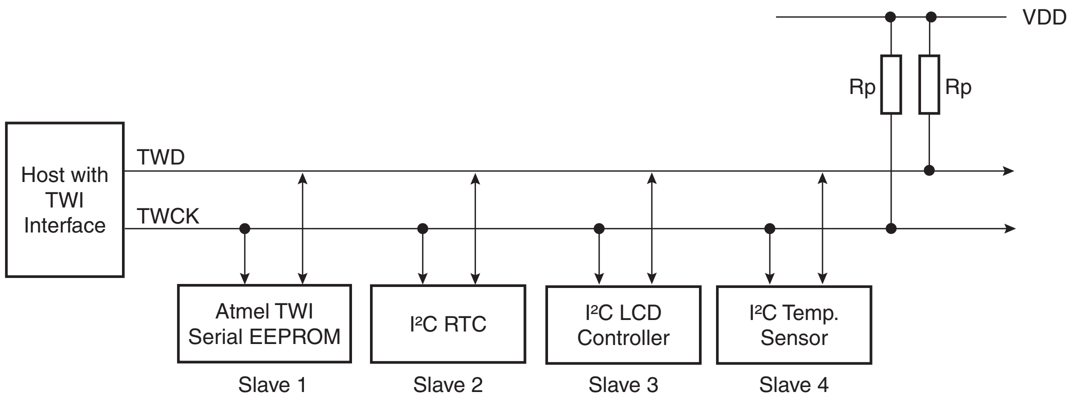
35.8 Master Mode

35.8.1 Definition

The Master is the device that starts a transfer, generates a clock and stops it.

35.8.2 Application Block Diagram

Figure 35-5. Master Mode Typical Application Block Diagram



Rp: Pull up value as given by the I²C Standard

35.8.3 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.

35.8.4 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.

While no new data is written in the TWI_THR, the Serial Clock Line is tied low. When new data is written in the TWI_THR, the SCL is released and the data is sent. To generate a STOP event, the STOP command must be performed by writing in the STOP field of TWI_CR.

After a Master Write transfer, the Serial Clock line is stretched (tied low) while no new data is written in the TWI_THR or until a STOP command is performed.

See Figure 35-6, Figure 35-7, and Figure 35-8.

Figure 35-6. Master Write with One Data Byte

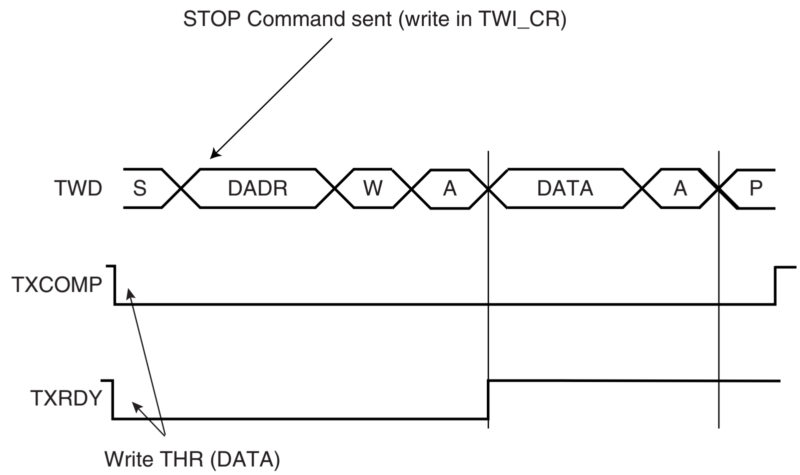


Figure 35-7. Master Write with Multiple Data Bytes

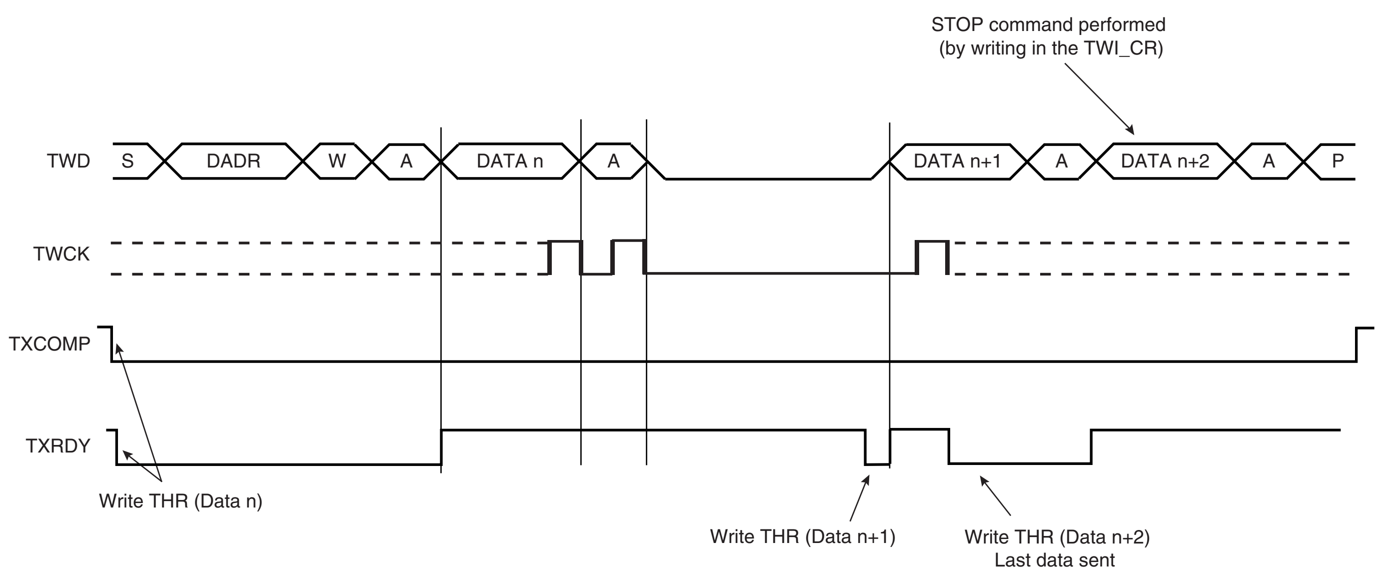
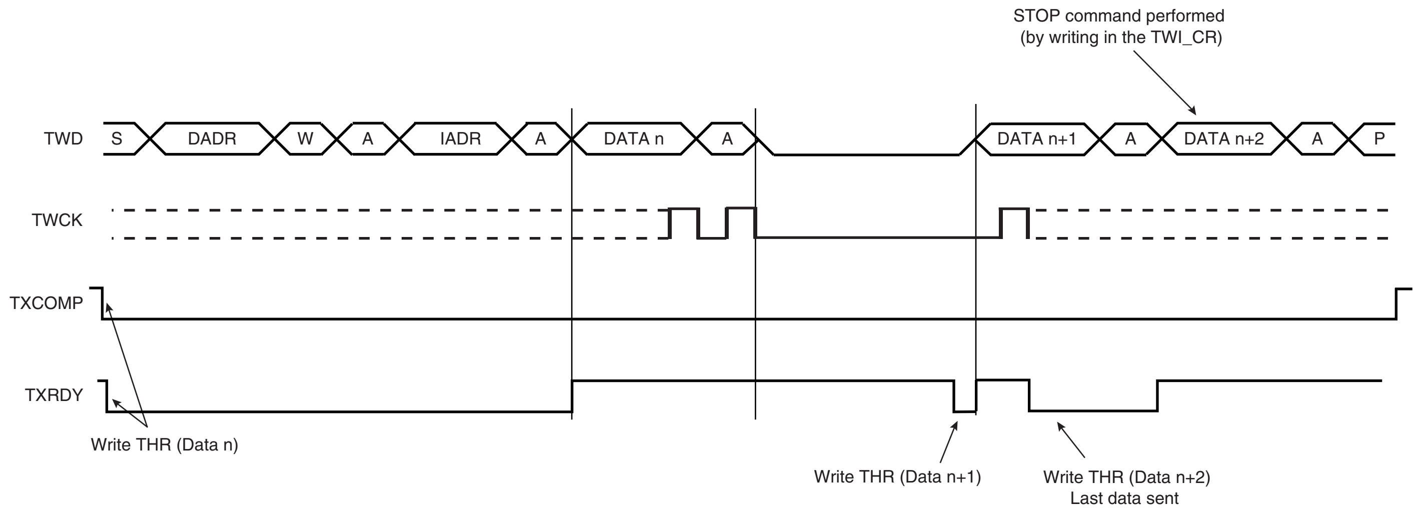


Figure 35-8. Master Write with One Byte Internal Address and Multiple Data Bytes



TXRDY is used as Transmit Ready for the PDC transmit channel.

35.8.5 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 35-9. 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 35-9. 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 35-10. For Internal Address usage see Section 35.8.6.

Figure 35-9. Master Read with One Data Byte

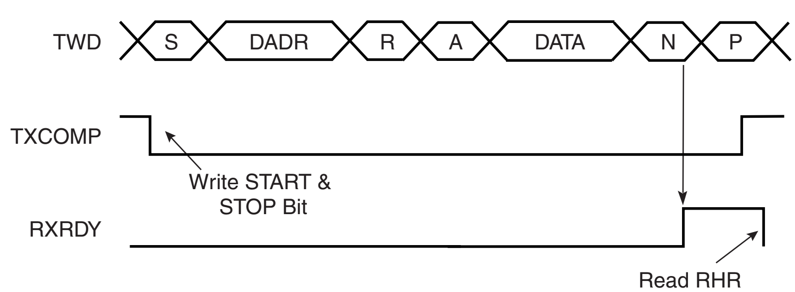
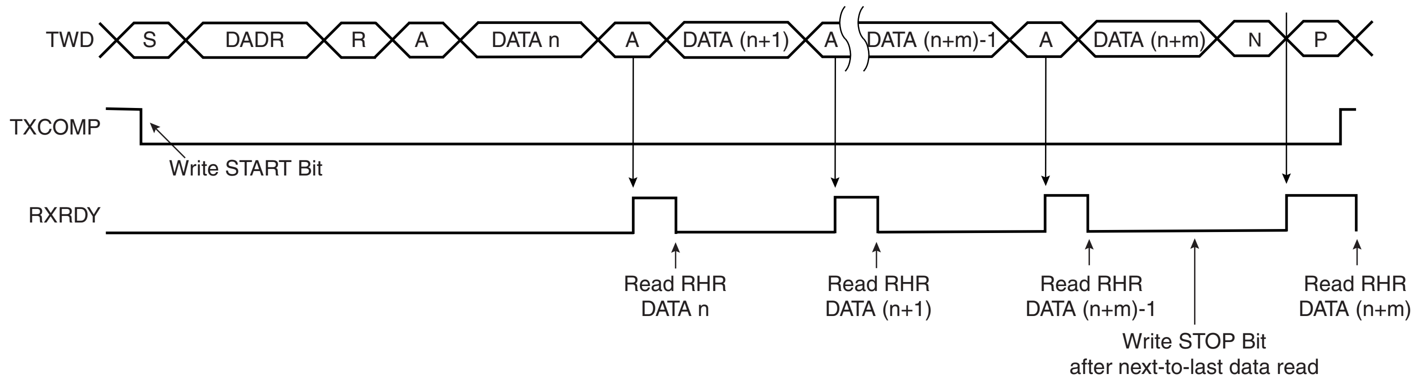


Figure 35-10. Master Read with Multiple Data Bytes



RXRDY is used as Receive Ready for the PDC receive channel.

35.8.6 Internal Address

The TWI interface can perform various transfer formats: Transfers with 7-bit slave address devices and 10-bit slave address devices.

35.8.6.1 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 35-12. See Figure 35-11 and Figure 35-13 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:

- S Start
- Sr Repeated Start
- P Stop
- W Write
- R Read
- A Acknowledge
- N Not Acknowledge
- DADR Device Address
- IADR Internal Address

Figure 35-11. Master Write with One, Two or Three Bytes Internal Address and One Data Byte

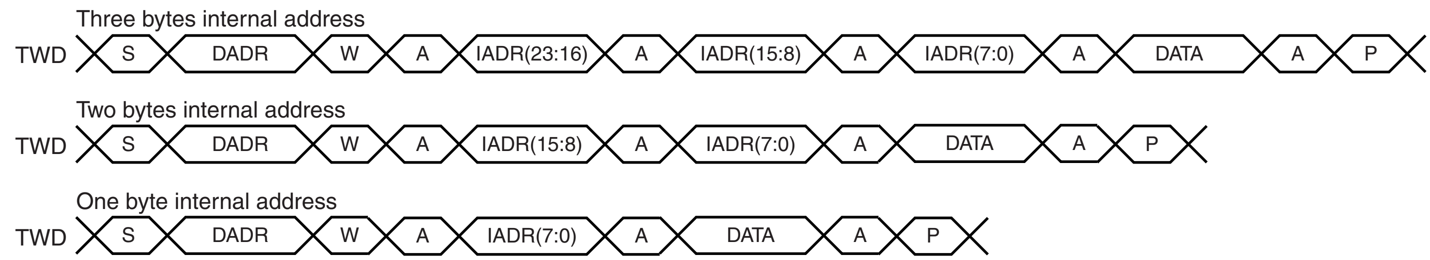
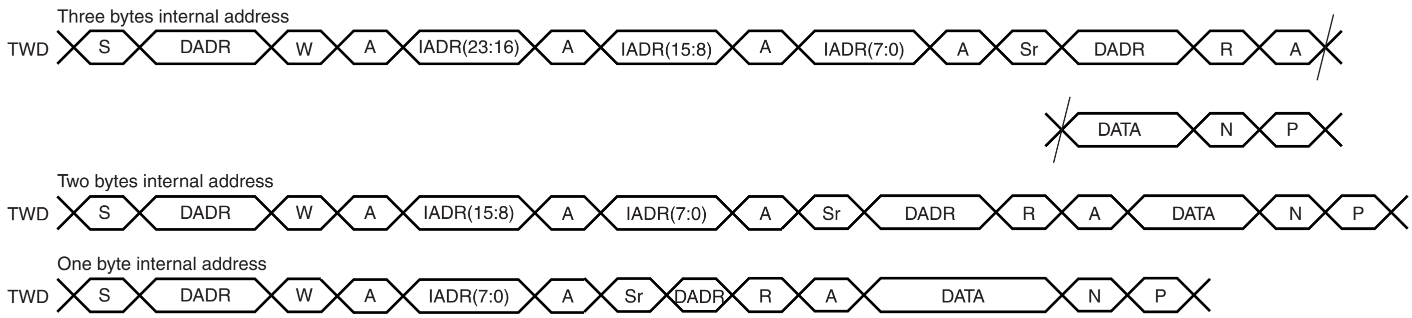


Figure 35-12. Master Read with One, Two or Three Bytes Internal Address and One Data Byte



35.8.6.2 *10-bit Slave Addressing*

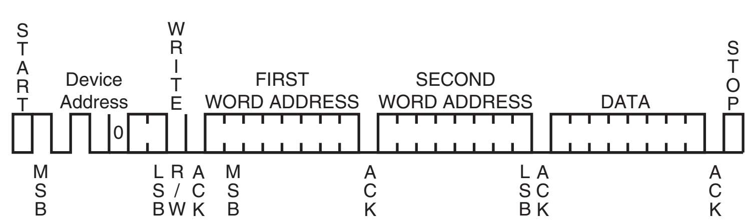
For a slave address higher than 7 bits, the user must configure the address size **(IADRSZ**) and set the other slave address bits in the internal address register (TWI_IADR). The two remaining Internal address bytes, IADR[15:8] and IADR[23:16] can be used the same as in 7-bit Slave Addressing.

<u>**Example:** Address a 10-bit device</u> (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 35-13 below shows a byte write to an Atmel AT24LC512 EEPROM. This demonstrates the use of internal addresses to access the device.

Figure 35-13. Internal Address Usage



35.8.7 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:

35.8.7.1 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.

35.8.7.2 Data Receive with the PDC

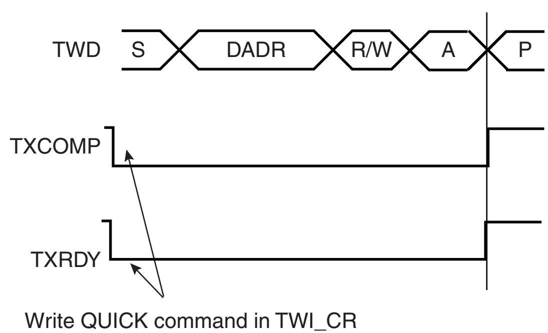
1. Initialize the receive PDC (memory pointers, size - 1, etc.).
2. Configure the master mode (DADR, CKDIV, etc.).
3. Start the transfer by setting the PDC RXTEN bit.
4. Wait for the PDC end RX flag.
5. Disable the PDC by setting the PDC RXDIS bit.

35.8.8 SMBUS Quick Command (Master Mode Only)

The TWI interface can perform a Quick Command:

1. Configure the master mode (DADR, CKDIV, etc.).
2. Write the MREAD bit in the TWI_MMR register at the value of the one-bit command to be sent.
3. Start the transfer by setting the QUICK bit in the TWI_CR.

Figure 35-14. SMBUS Quick Command



35.8.9 Read-write Flowcharts

The following flowcharts shown in Figure 35-16 on page 540, Figure 35-17 on page 541, Figure 35-18 on page 542, Figure 35-19 on page 543 and Figure 35-20 on page 544 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 35-15. TWI Write Operation with Single Data Byte without Internal Address

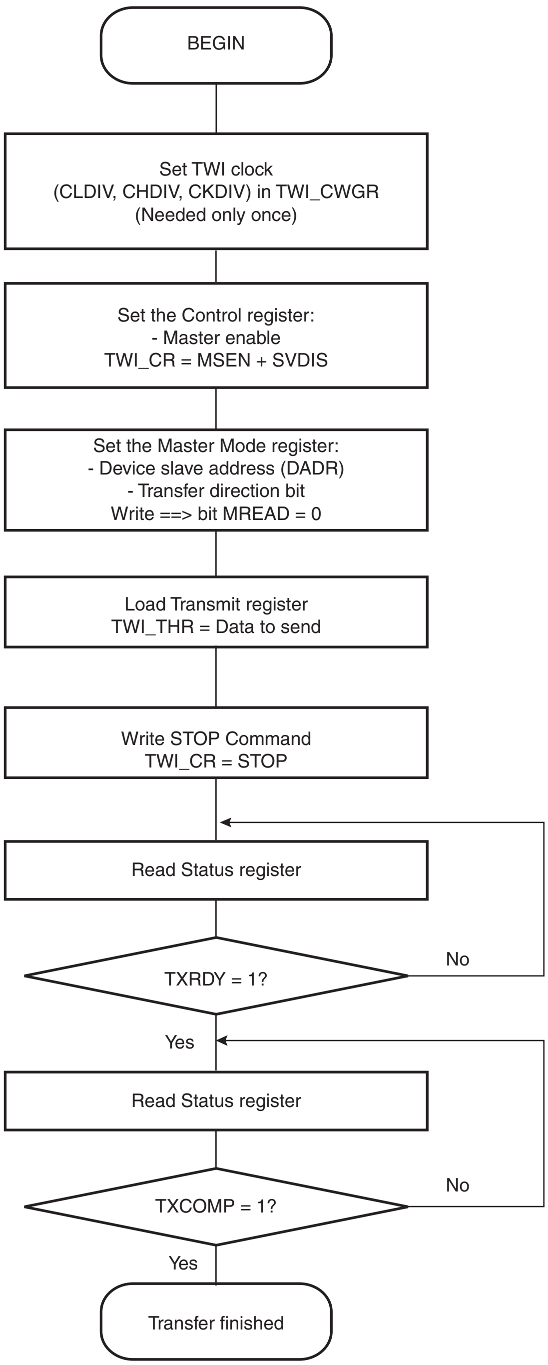


Figure 35-16. TWI Write Operation with Single Data Byte and Internal Address

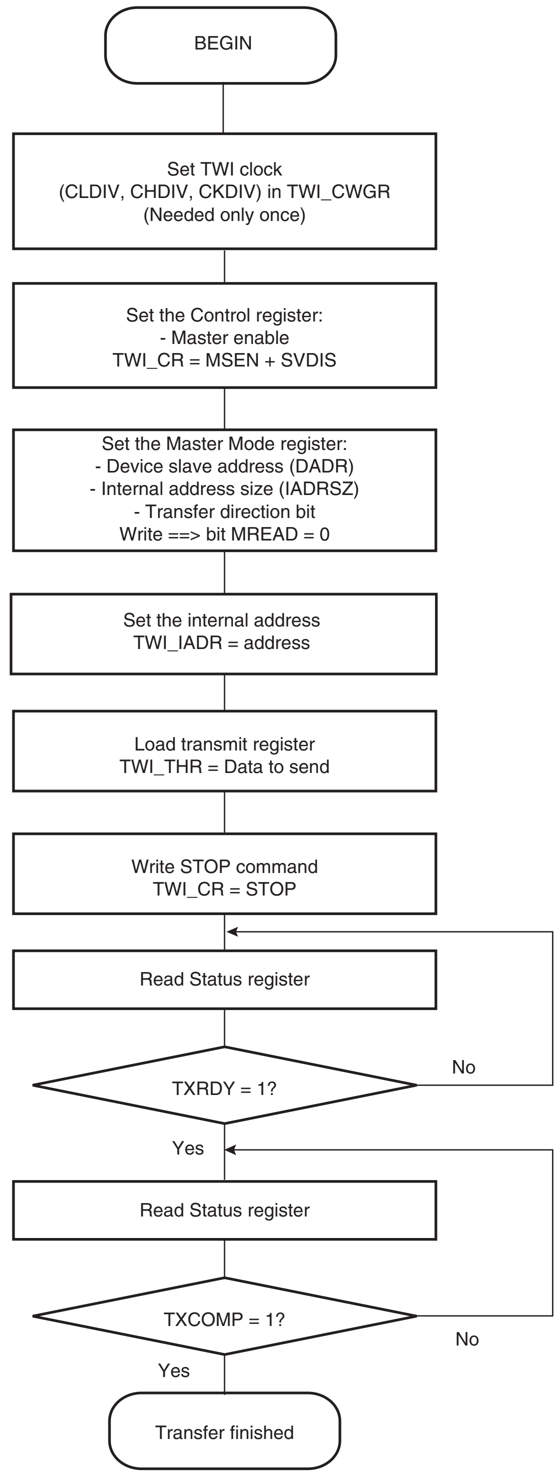


Figure 35-17. TWI Write Operation with Multiple Data Bytes with or without Internal Address

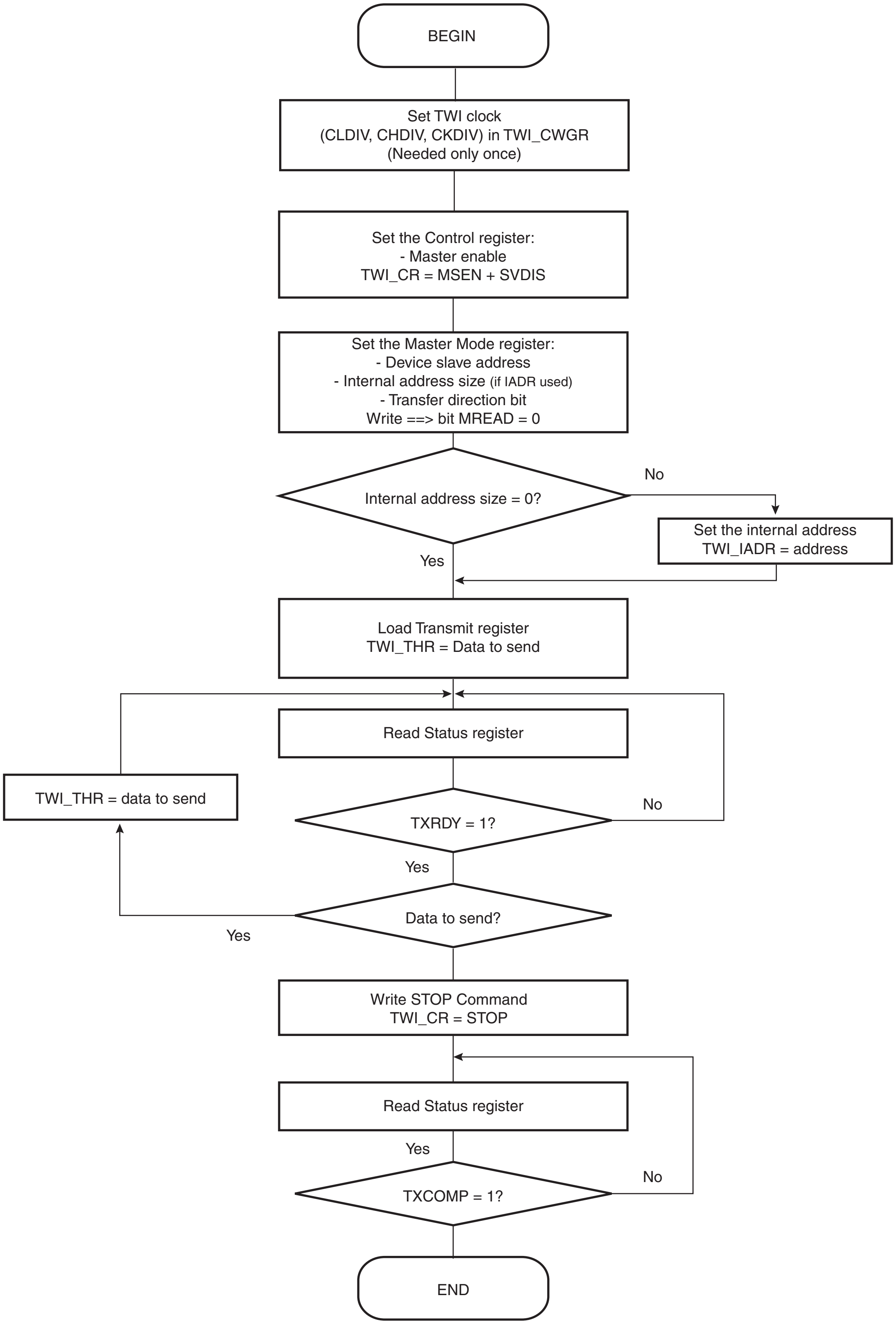


Figure 35-18. TWI Read Operation with Single Data Byte without Internal Address

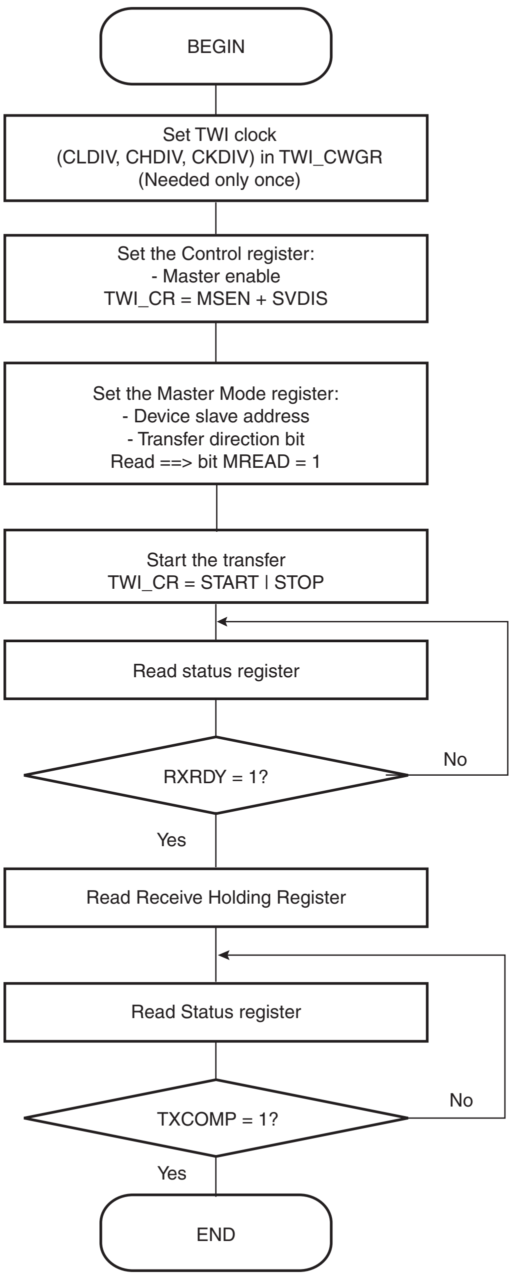


Figure 35-19. TWI Read Operation with Single Data Byte and Internal Address

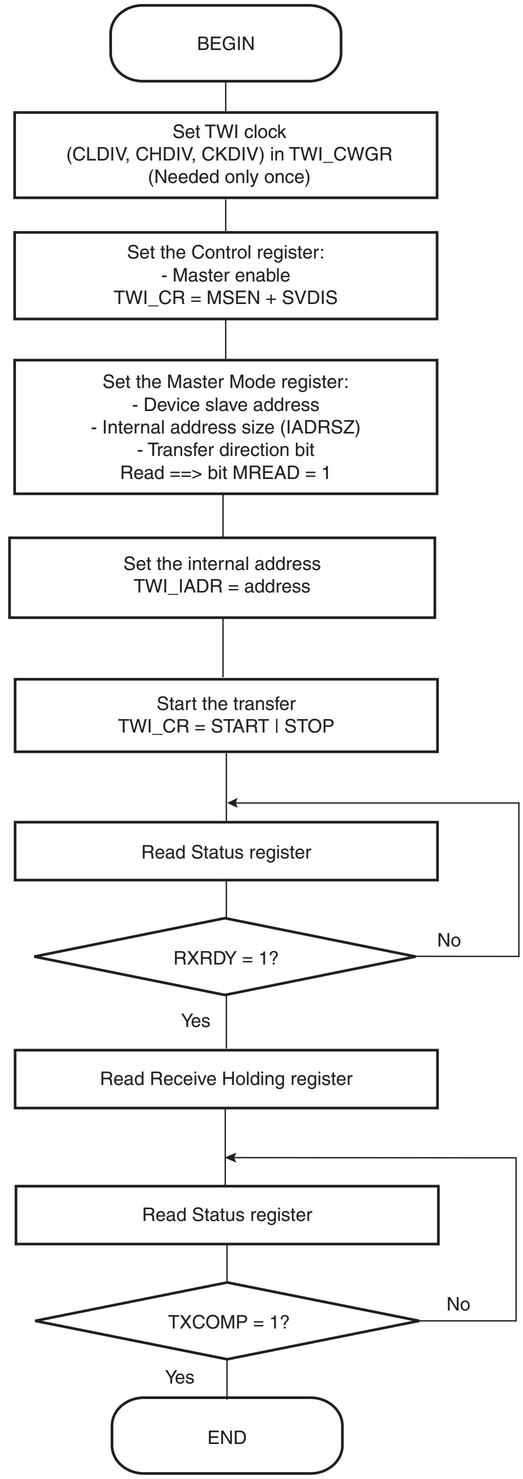
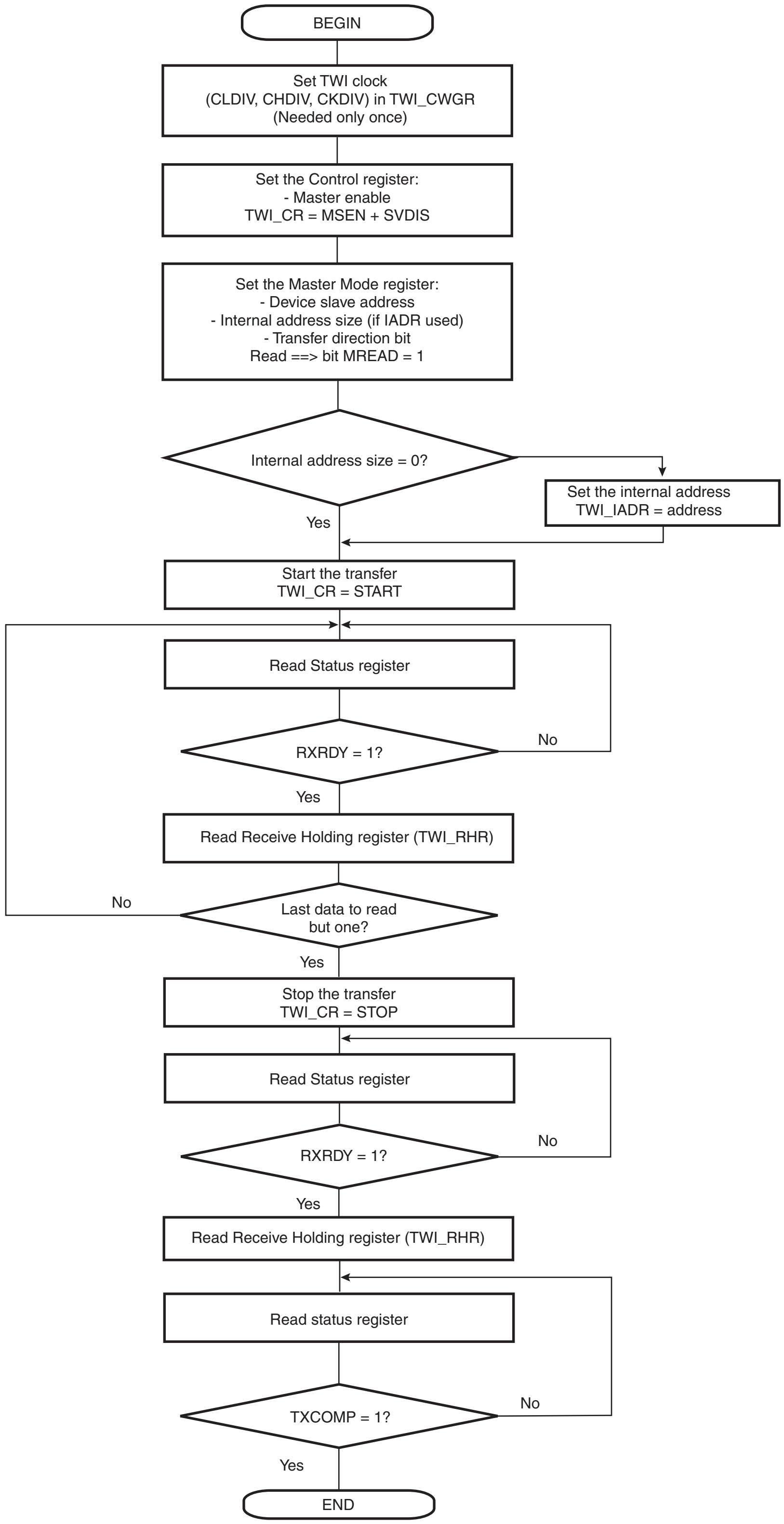


Figure 35-20. TWI Read Operation with Multiple Data Bytes with or without Internal Address



35.9 Multi-master Mode

35.9.1 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 35-22 on page 546.

35.9.2 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.

35.9.2.1 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 35-21 on page 546).

Note: The state of the bus (busy or free) is not indicated in the user interface.

35.9.2.2 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.
7. 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 35-21. Programmer Sends Data While the Bus is Busy

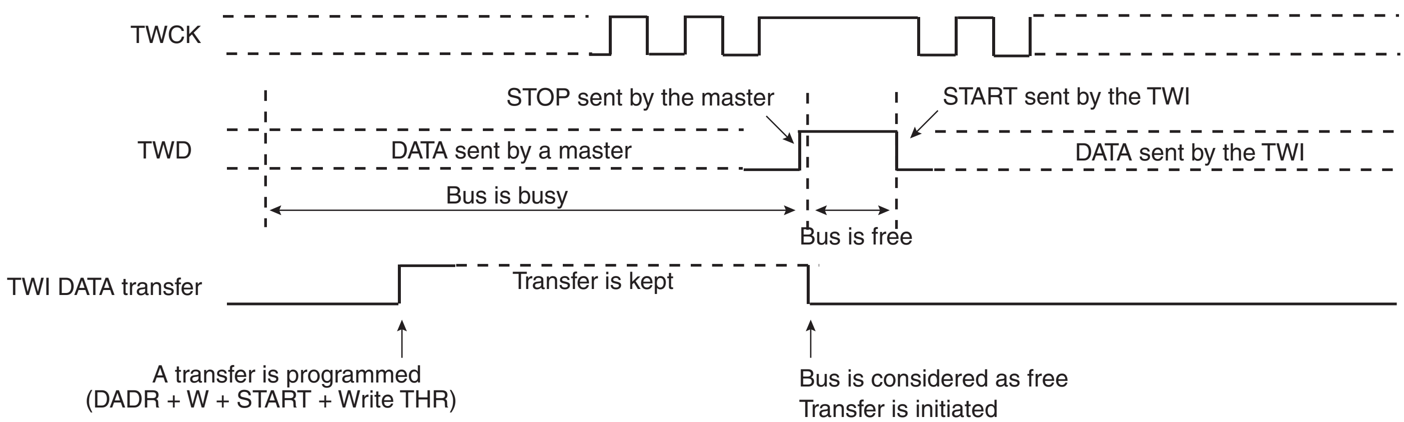
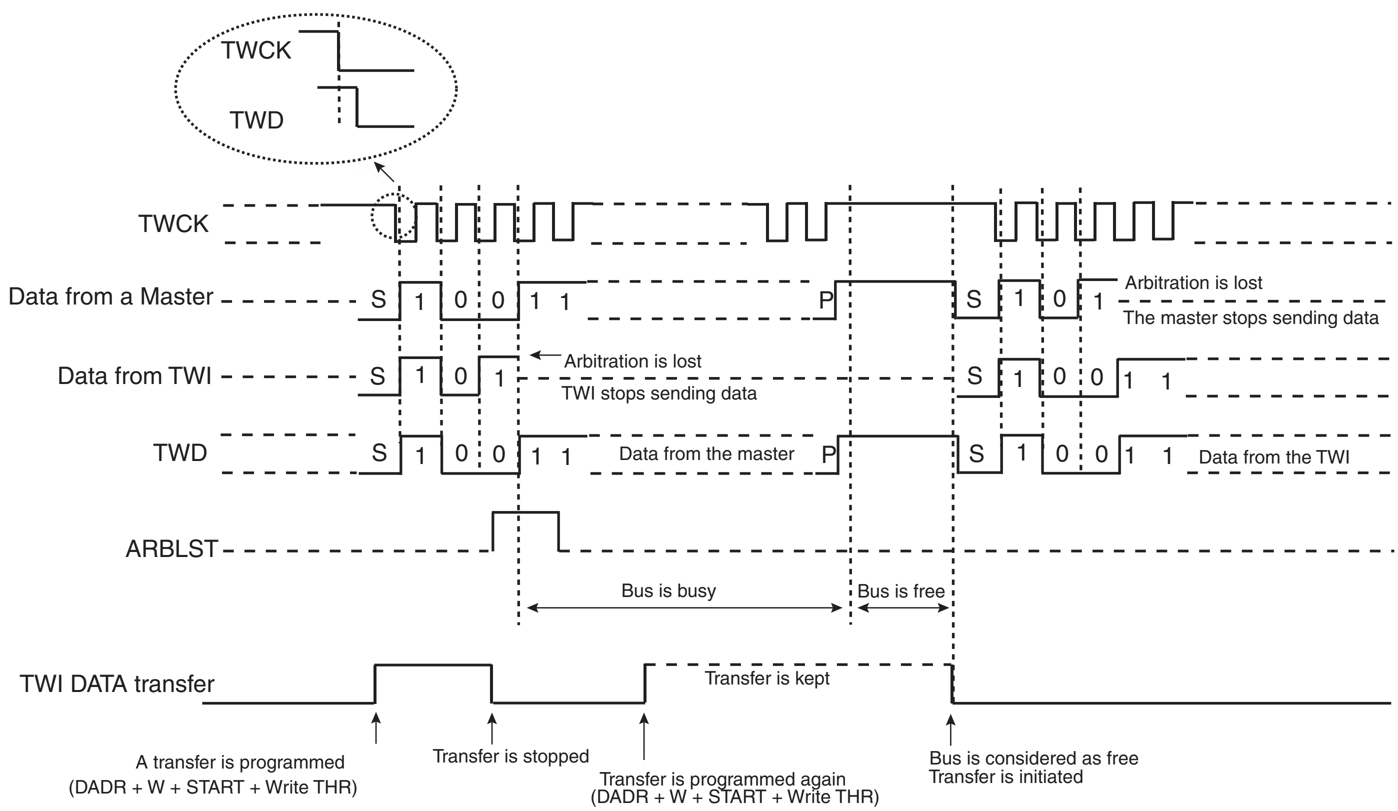
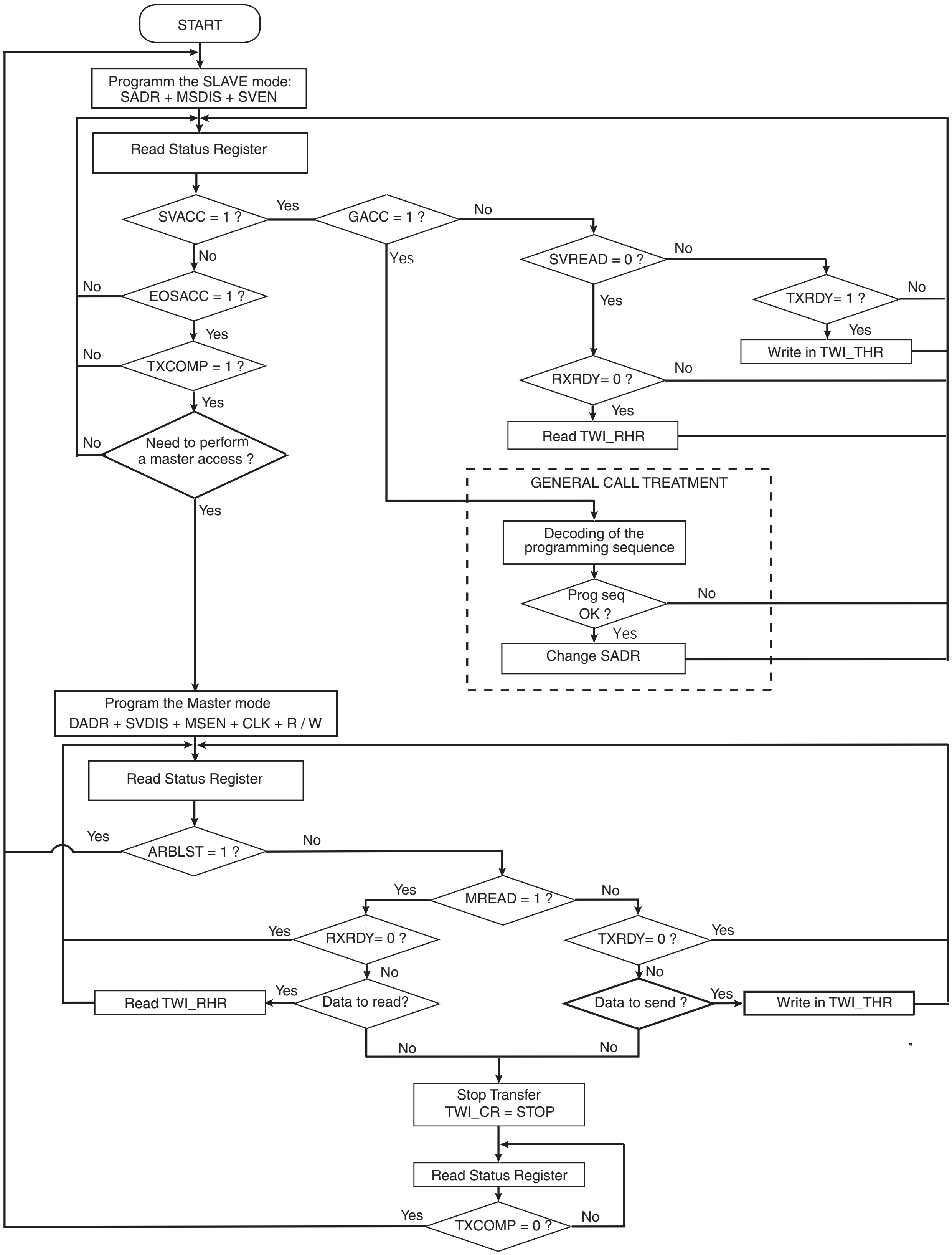


Figure 35-22. Arbitration Cases



The flowchart shown in Figure 35-23 on page 547 gives an example of read and write operations in Multi-master mode.

Figure 35-23. Multi-master Flowchart



35.10 Slave Mode

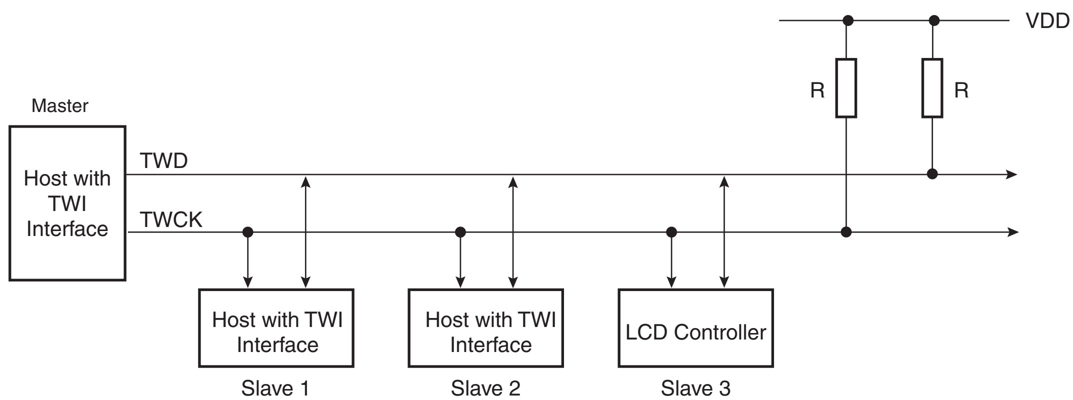
35.10.1 Definition

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).

35.10.2 Application Block Diagram

Figure 35-24. Slave Mode Typical Application Block Diagram



35.10.3 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.

35.10.4 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.

35.10.4.1 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 35-25 on page 550.

35.10.4.2 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 35-26 on page 550.

35.10.4.3 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 35-28 on page 552 and Figure 35-29 on page 553.

35.10.4.4 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 35-27 on page 551.

35.10.4.5 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.

35.10.5 Data Transfer

35.10.5.1 Read Operation

The read mode is defined as a data requirement from the master.

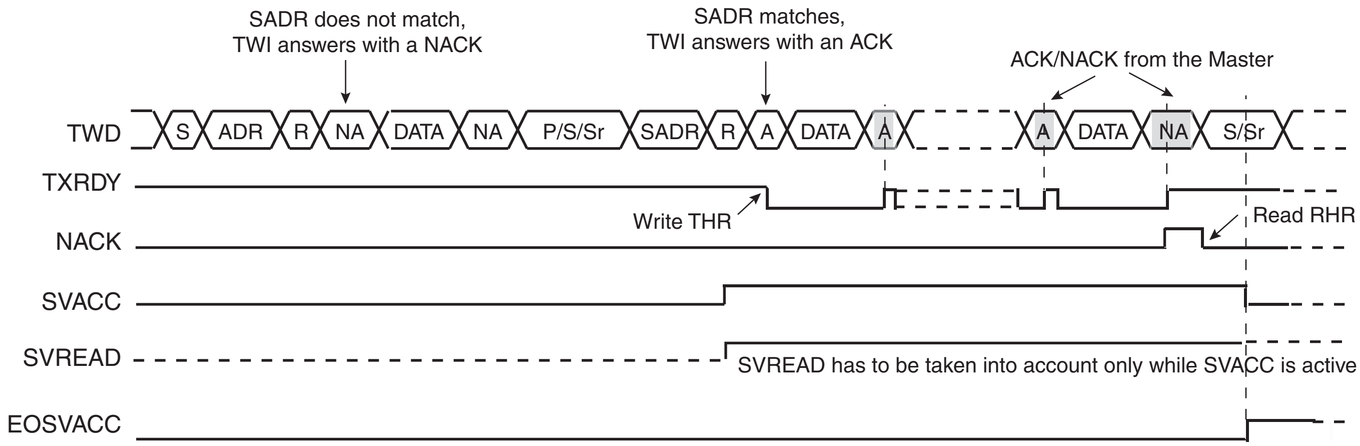
After a START or a REPEATED START condition is detected, the decoding of the address starts. If the slave address (SADR) is decoded, SVACC is set and SVREAD indicates the direction of the transfer.

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 35-25 on page 550 describes the write operation.

Figure 35-25. 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.

35.10.5.2 Write Operation

The write mode is defined as a data transmission from the master.

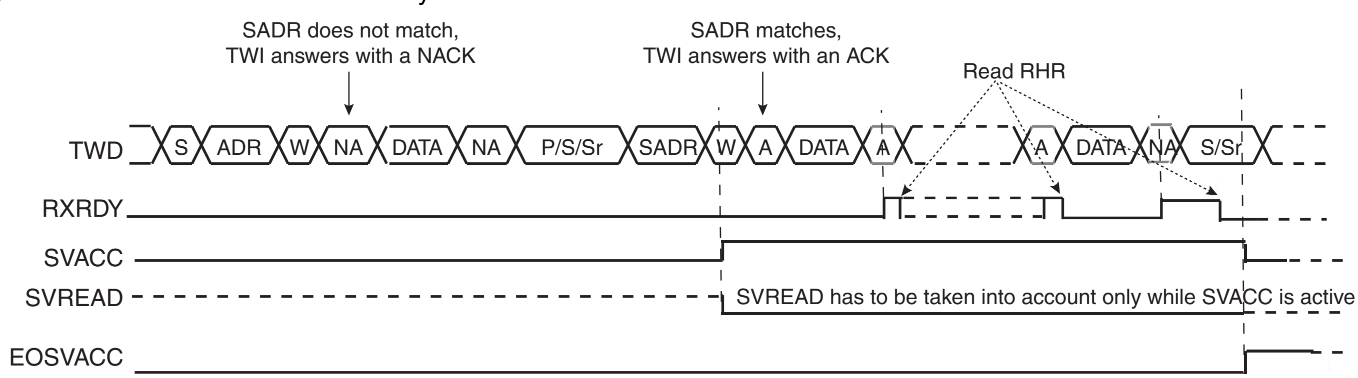
After a START or a REPEATED START, the decoding of the address starts. If the slave address is decoded, SVACC is set and SVREAD indicates the direction of the transfer (SVREAD is low in this case).

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 35-26 on page 550 describes the Write operation.

Figure 35-26. 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.

35.10.5.3 General Call

The general call is performed in order to change the address of the slave.

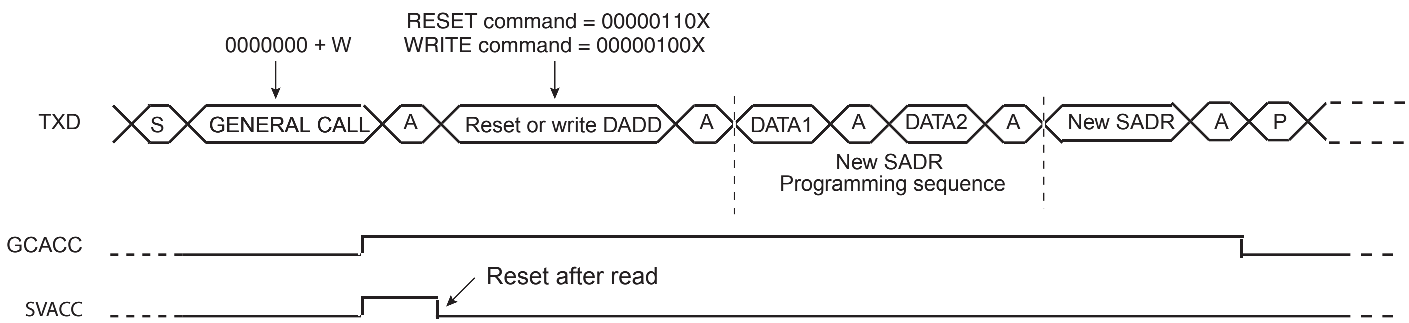
If a GENERAL CALL is detected, GACC is set.

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 35-27 on page 551 describes the General Call access.

Figure 35-27. 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.

35.10.5.4 Clock Synchronization

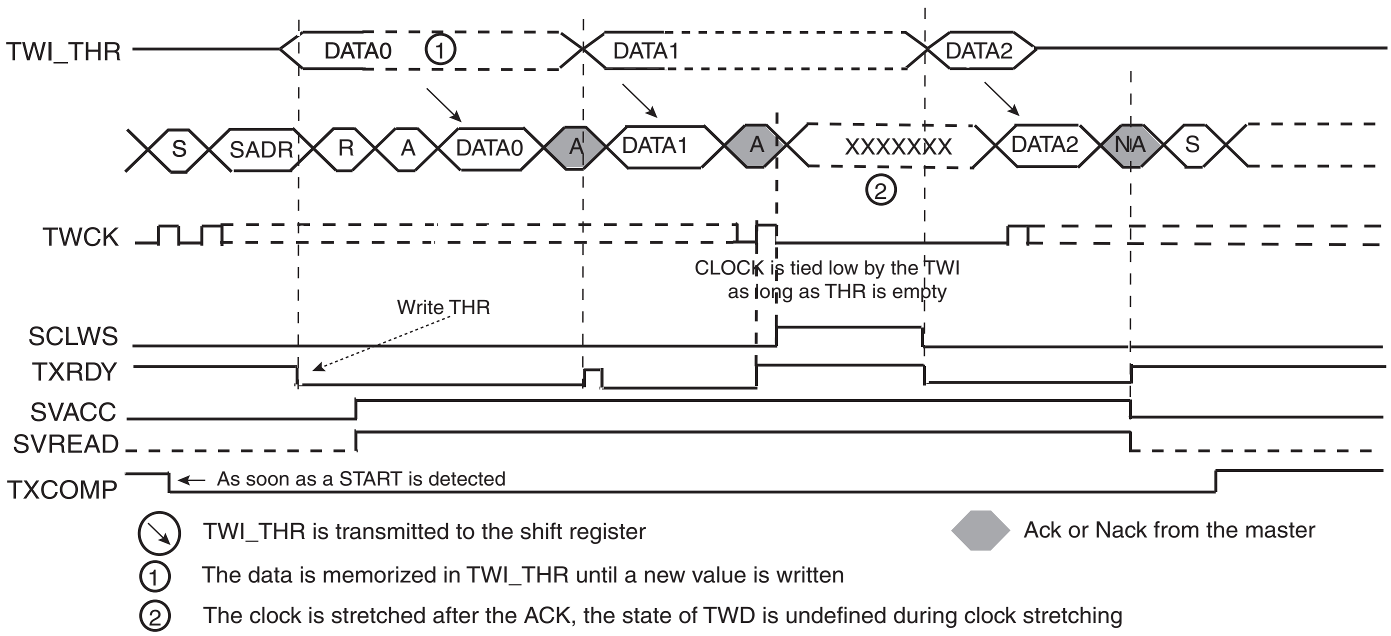
In both read and write modes, it may happen that TWI_THR/TWI_RHR buffer is not filled /emptied before the emission/reception of a new character. In this case, to avoid sending/receiving undesired data, a clock stretching mechanism is implemented.

35.10.5.5 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 35-28 on page 552 describes the clock synchronization in Read mode.

Figure 35-28. Clock Synchronization in Read Mode



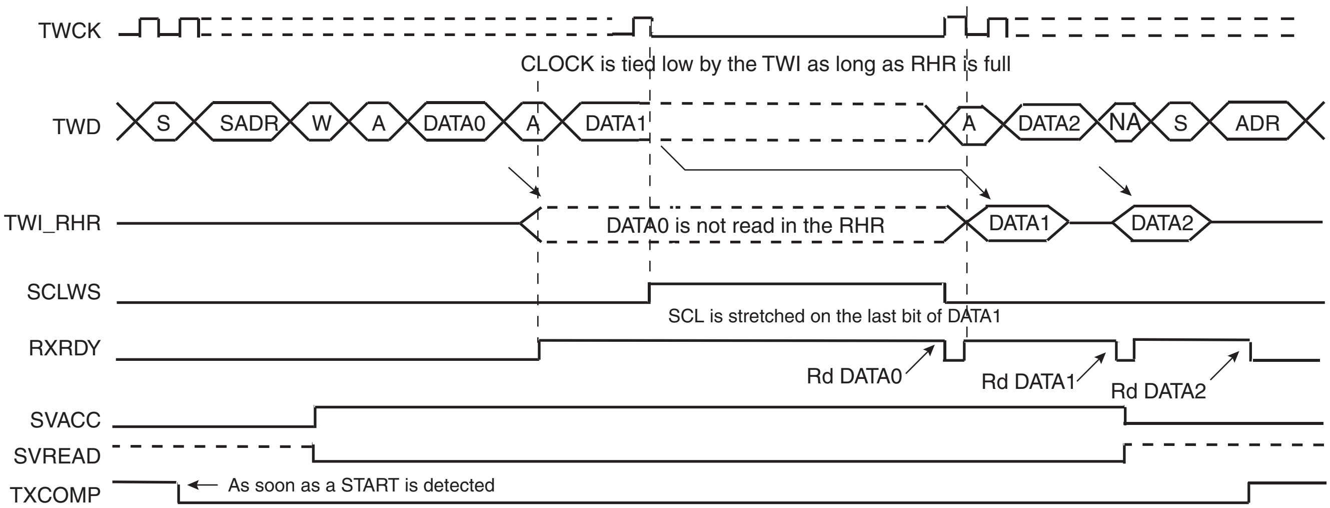
Notes:
1. TXRDY is reset when data has been written in the TWI_THR to the shift register and set when this data has been acknowledged or non acknowledged.
2. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.
3. SCLWS is automatically set when the clock synchronization mechanism is started.

35.10.5.6 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 35-29 on page 553 describes the clock synchronization in Read mode.

Figure 35-29. 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.

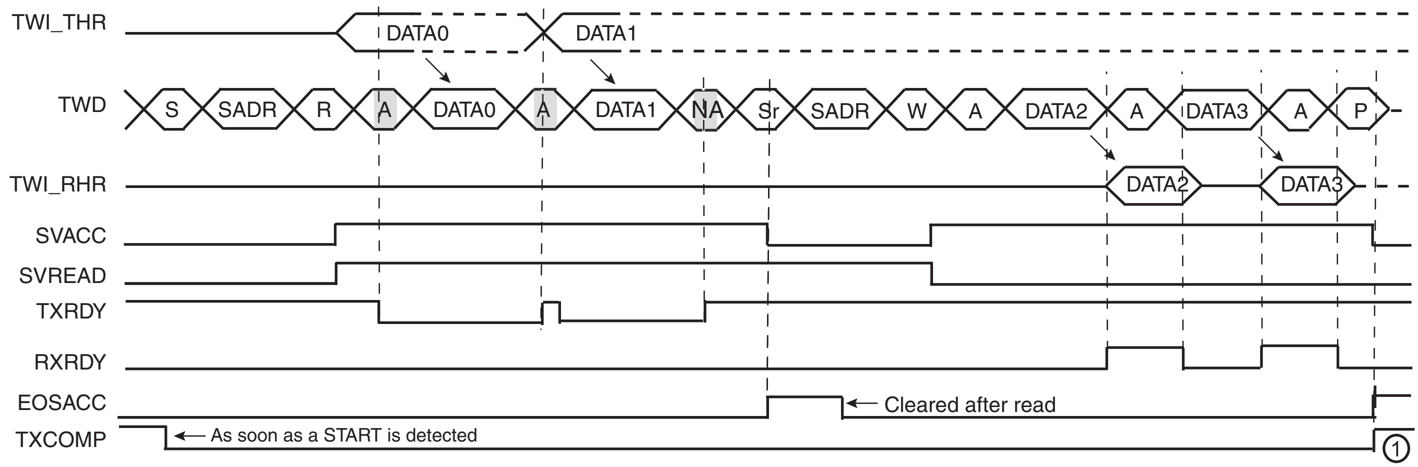
35.10.5.7 Reversal after a Repeated Start

35.10.5.8 Reversal of Read to Write

The master initiates the communication by a read command and finishes it by a write command.

Figure 35-30 on page 554 describes the repeated start + reversal from Read to Write mode.

Figure 35-30. Repeated Start + Reversal from Read to Write Mode

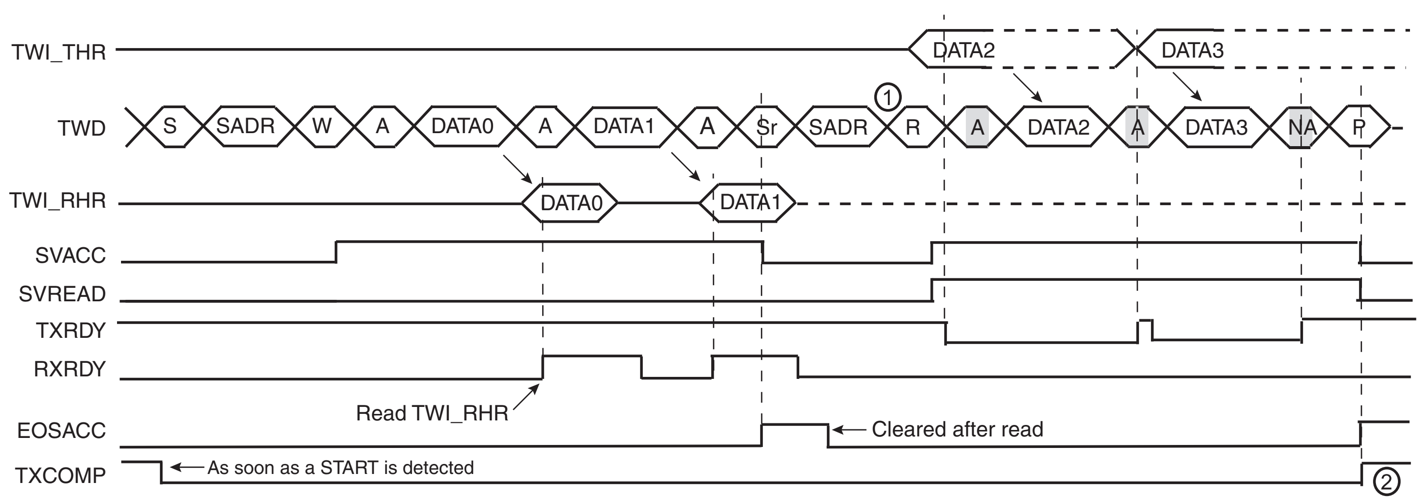


1. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

35.10.5.9 Reversal of Write to Read

The master initiates the communication by a write command and finishes it by a read command. Figure 35-31 on page 554 describes the repeated start + reversal from Write to Read mode.

Figure 35-31. Repeated Start + Reversal from Write to Read Mode



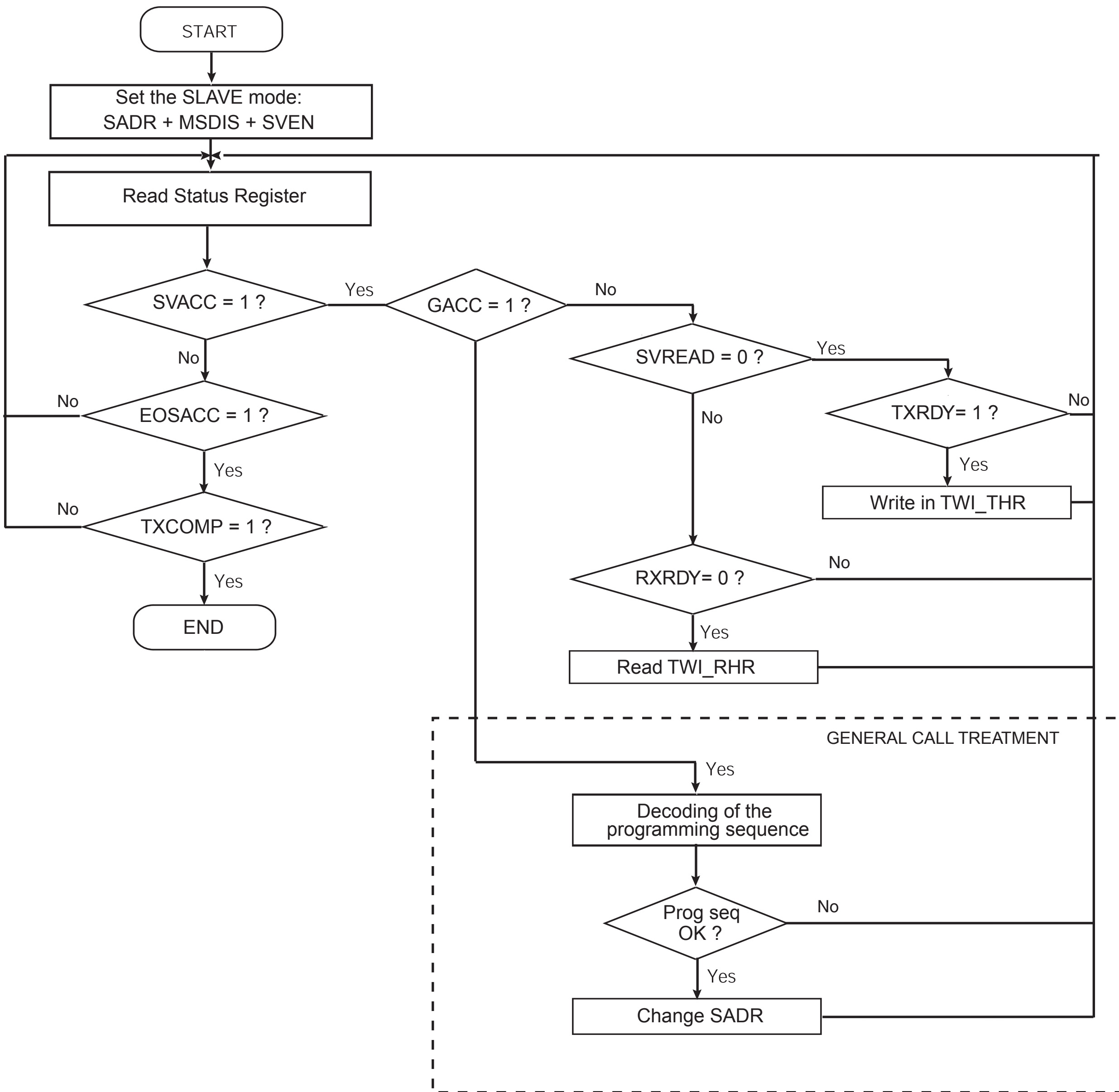
Notes: 1. In this case, if TWI_THR has not been written at the end of the read command, the clock is automatically stretched before the ACK.

2. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

35.10.6 Read Write Flowcharts

The flowchart shown in Figure 35-32 on page 555 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 35-32. Read Write Flowchart in Slave Mode



35.11 Two-wire Interface (TWI) User Interface

Table 35-5. Register Mapping

Offset	Register	Name	Access	Reset
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	–	–	–

35.11.1 TWI Control Register

Name: TWI_CR

Address: 0xFFF88000

Access: Write-only

Reset: 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	QUICK	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 master data write operation, a STOP condition will be sent after the transmission of the current data is finished.

- **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.

- **QUICK: SMBUS Quick Command**

0 = No effect.

1 = If Master mode is enabled, a SMBUS Quick Command is sent.

- **SWRST: Software Reset**

0 = No effect.

1 = Equivalent to a system reset.

35.11.2 TWI Master Mode Register

Name: TWI_MMR

Address: 0xFFF88004

Access: Read-write

Reset: 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.

35.11.3 TWI Slave Mode Register

Name: TWI_SMR

Address: 0xFFF88008

Access: Read-write

Reset: 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.

35.11.4 TWI Internal Address Register

Name: TWI_IADR

Address: 0xFFF8800C

Access: Read-write

Reset: 0x00000000

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

23	22	21	20	19	18	17	16
IADR							

15	14	13	12	11	10	9	8
IADR							

7	6	5	4	3	2	1	0
IADR							

- **IADR: Internal Address**

0, 1, 2 or 3 bytes depending on IADRSZ.

35.11.5 TWI Clock Waveform Generator Register

Name: TWI_CWGR

Address: 0xFFF88010

Access: Read-write

Reset: 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:

$$T_{low} = ((\text{CLDIV} \times 2^{\text{CKDIV}}) + 4) \times T_{MCK}$$

- **CHDIV: Clock High Divider**

The SCL high period is defined as follows:

$$T_{high} = ((\text{CHDIV} \times 2^{\text{CKDIV}}) + 4) \times T_{MCK}$$

- **CKDIV: Clock Divider**

The CKDIV is used to increase both SCL high and low periods.

35.11.6 TWI Status Register

Name TWI_SR

Address: 0xFFF88020

Access: Read-only

Reset: 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 35-8 on page 535 and in Figure 35-10 on page 536.

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 35-28 on page 552, Figure 35-29 on page 553, Figure 35-30 on page 554 and Figure 35-31 on page 554.

- **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 35-10 on page 536.

RXRDY behavior in Slave mode can be seen in Figure 35-26 on page 550, Figure 35-29 on page 553, Figure 35-30 on page 554 and Figure 35-31 on page 554.

- **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 35-8 on page 535.

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 35-25 on page 550, Figure 35-28 on page 552, Figure 35-30 on page 554 and Figure 35-31 on page 554.

- **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 35-25 on page 550, Figure 35-26 on page 550, Figure 35-30 on page 554 and Figure 35-31 on page 554.

- **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 35-25 on page 550, Figure 35-26 on page 550, Figure 35-30 on page 554 and Figure 35-31 on page 554.

- **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, if need be, the programmer may acknowledge this access and decode the following bytes and respond according to the value of the bytes.

GACC behavior can be seen in Figure 35-27 on page 551.

- **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 35-28 on page 552 and Figure 35-29 on page 553.

- **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 35-30 on page 554 and Figure 35-31 on page 554

- **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.

35.11.7 TWI Interrupt Enable Register

Name: TWI_IER

Address: 0xFFF88024

Access: Write-only

Reset: 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**
- **EOSACC: End Of Slave Access 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.

35.11.8 TWI Interrupt Disable Register

Name: TWI_IDR

Address: 0xFFF88028

Access: Write-only

Reset: 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.

35.11.9 TWI Interrupt Mask Register

Name: TWI_IMR

Address: 0xFFF8802C

Access: Read-only

Reset: 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.

35.11.10 TWI Receive Holding Register

Name: TWI_RHR

Address: 0xFFF88030

Access: Read-only

Reset: 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**

35.11.11 TWI Transmit Holding Register

Name: TWI_THR

Address: 0xFFF88034

Access: Read-write

Reset: 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**

36. Universal Synchronous Asynchronous Receiver Transceiver (USART)

36.1 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.

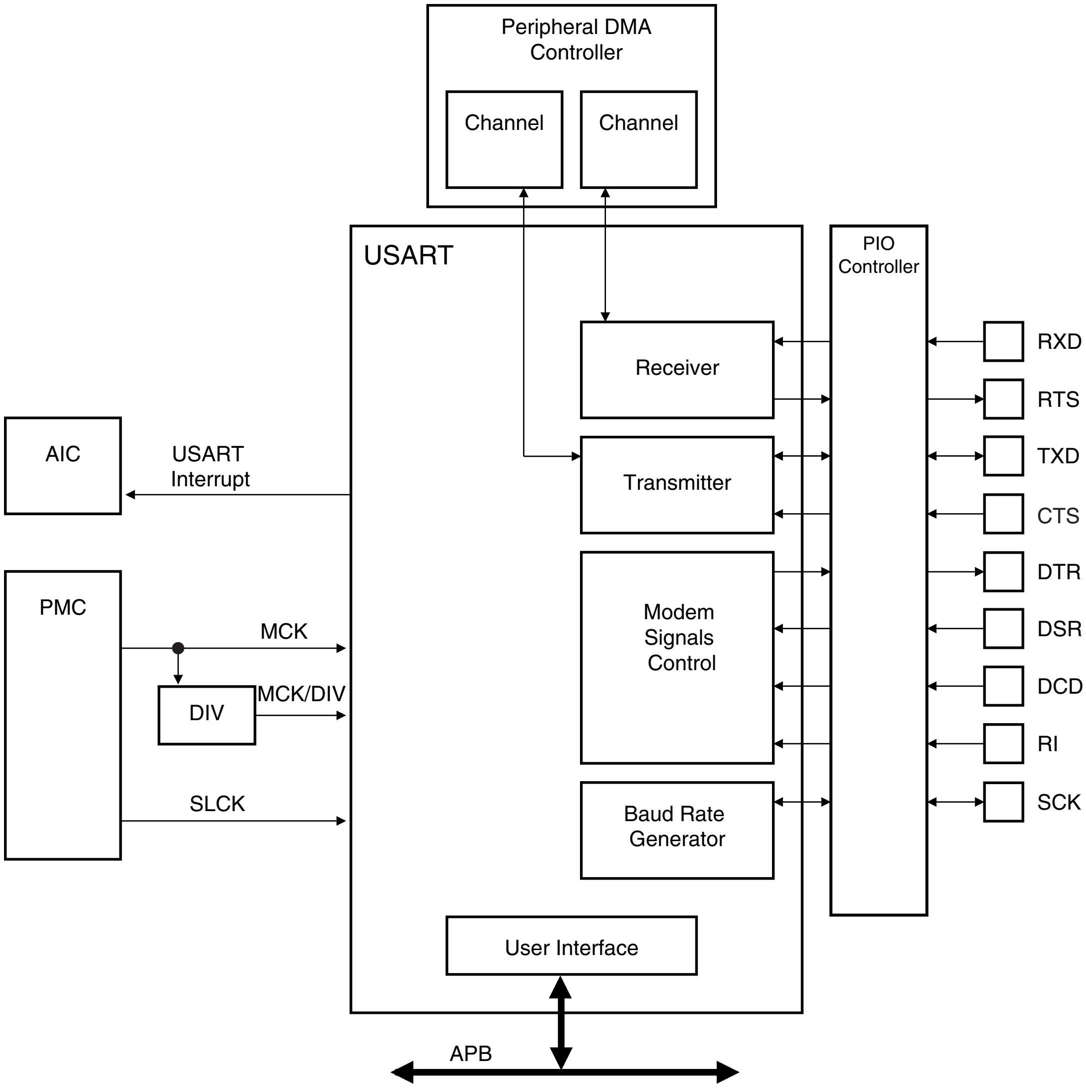
The USART features three test modes: remote loopback, local loopback and automatic echo.

The USART supports specific operating modes providing interfaces on RS485 buses, with ISO7816 T = 0 or T = 1 smart card slots and infrared transceivers. 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.

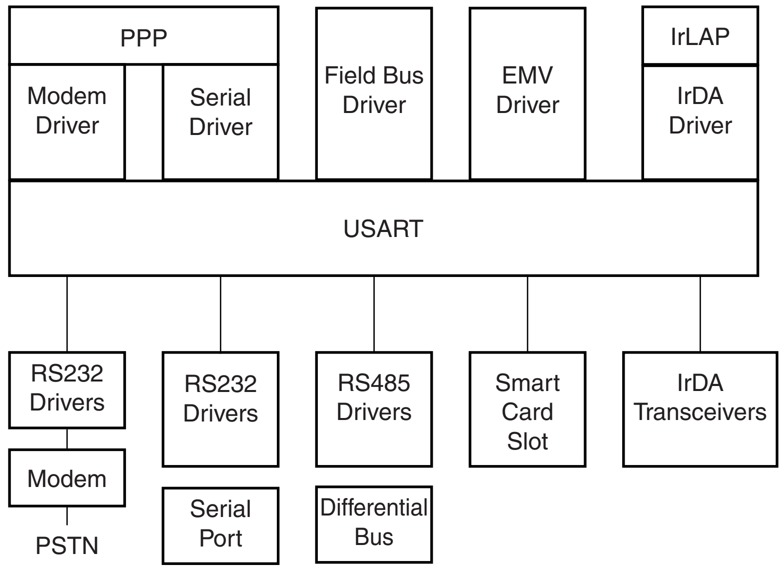
36.2 Block Diagram

Figure 36-1. USART Block Diagram



36.3 Application Block Diagram

Figure 36-2. Application Block Diagram



36.4 I/O Lines Description

Table 36-1. 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	
CTS	Clear to Send	Input	Low
RTS	Request to Send	Output	Low

36.5 Product Dependencies

36.5.1 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 is used, the internal pull up on TXD must also be enabled.

Only USART0 is fully equipped with all the modem signals.

36.5.2 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.

36.5.3 Interrupt

The USART interrupt line is connected on one of the internal sources of the Advanced Inter-rupt 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.

Table 36-2. Peripheral IDs

Instance	ID
USART0	8
USART1	9
USART2	10

36.6 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 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 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

36.6.1 Baud Rate Generator

The Baud Rate Generator provides the bit period clock named the Baud Rate Clock to both the receiver and the transmitter.

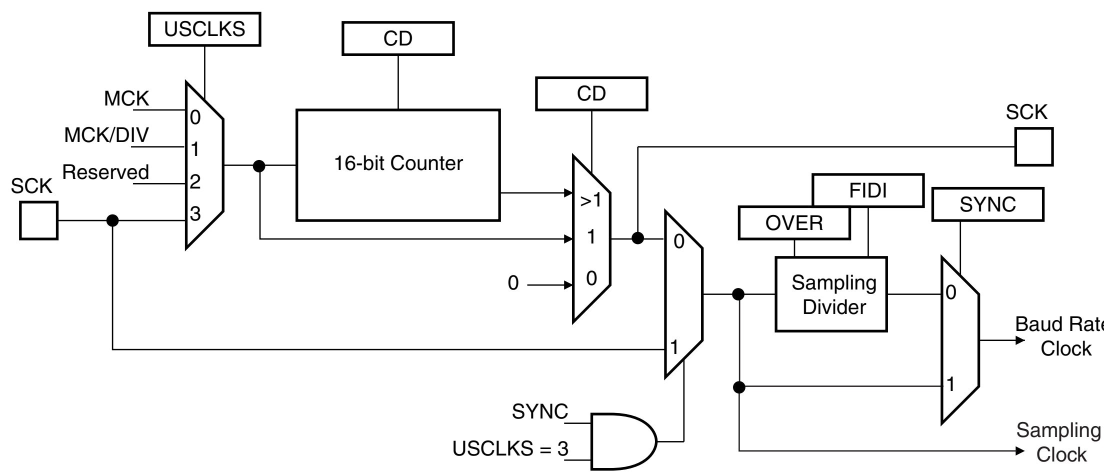
The Baud Rate Generator clock source can be selected by setting the USCLKS field in the Mode Register (US_MR) between:

- 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 36-3. Baud Rate Generator



36.6.1.1 *Baud Rate in Asynchronous Mode*

If the USART is programmed to operate in asynchronous mode, the selected clock is first divided by CD, which is field programmed in the Baud Rate Generator Register (US_BRGR). The resulting clock is provided to the receiver as a sampling clock and then divided by 16 or 8, depending on the programming of the OVER bit in US_MR.

If OVER is set to 1, the receiver sampling is 8 times higher than the baud rate clock. If OVER is cleared, the sampling is performed at 16 times the baud rate clock.

The following formula performs the calculation of the Baud Rate.

$$Baudrate = \frac{SelectedClock}{(8(2 - Over)CD)}$$

This gives a maximum baud rate of MCK divided by 8, assuming that MCK is the highest possible clock and that OVER is programmed at 1.

36.6.1.2 Baud Rate Calculation Example

Table 36-3 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 36-3. Baud Rate Example (OVER = 0)

Source Clock	Expected Baud Rate	Calculation Result	CD	Actual Baud Rate	Error
MHz	Bit/s			Bit/s	
3 686 400	38 400	6.00	6	38 400.00	0.00%
4 915 200	38 400	8.00	8	38 400.00	0.00%
5 000 000	38 400	8.14	8	39 062.50	1.70%
7 372 800	38 400	12.00	12	38 400.00	0.00%
8 000 000	38 400	13.02	13	38 461.54	0.16%
12 000 000	38 400	19.53	20	37 500.00	2.40%
12 288 000	38 400	20.00	20	38 400.00	0.00%
14 318 180	38 400	23.30	23	38 908.10	1.31%
14 745 600	38 400	24.00	24	38 400.00	0.00%
18 432 000	38 400	30.00	30	38 400.00	0.00%
24 000 000	38 400	39.06	39	38 461.54	0.16%
24 576 000	38 400	40.00	40	38 400.00	0.00%
25 000 000	38 400	40.69	40	38 109.76	0.76%
32 000 000	38 400	52.08	52	38 461.54	0.16%
32 768 000	38 400	53.33	53	38 641.51	0.63%
33 000 000	38 400	53.71	54	38 194.44	0.54%
40 000 000	38 400	65.10	65	38 461.54	0.16%
50 000 000	38 400	81.38	81	38 580.25	0.47%
60 000 000	38 400	97.66	98	38 265.31	0.35%
70 000 000	38 400	113.93	114	38 377.19	0.06%

The baud rate is calculated with the following formula:

$$BaudRate = MCK / CD \times 16$$

The baud rate error is calculated with the following formula. It is not recommended to work with an error higher than 5%.

$$Error = 1 - \left(\frac{ExpectedBaudRate}{ActualBaudRate}\right)$$

36.6.1.3 Fractional Baud Rate in Asynchronous Mode

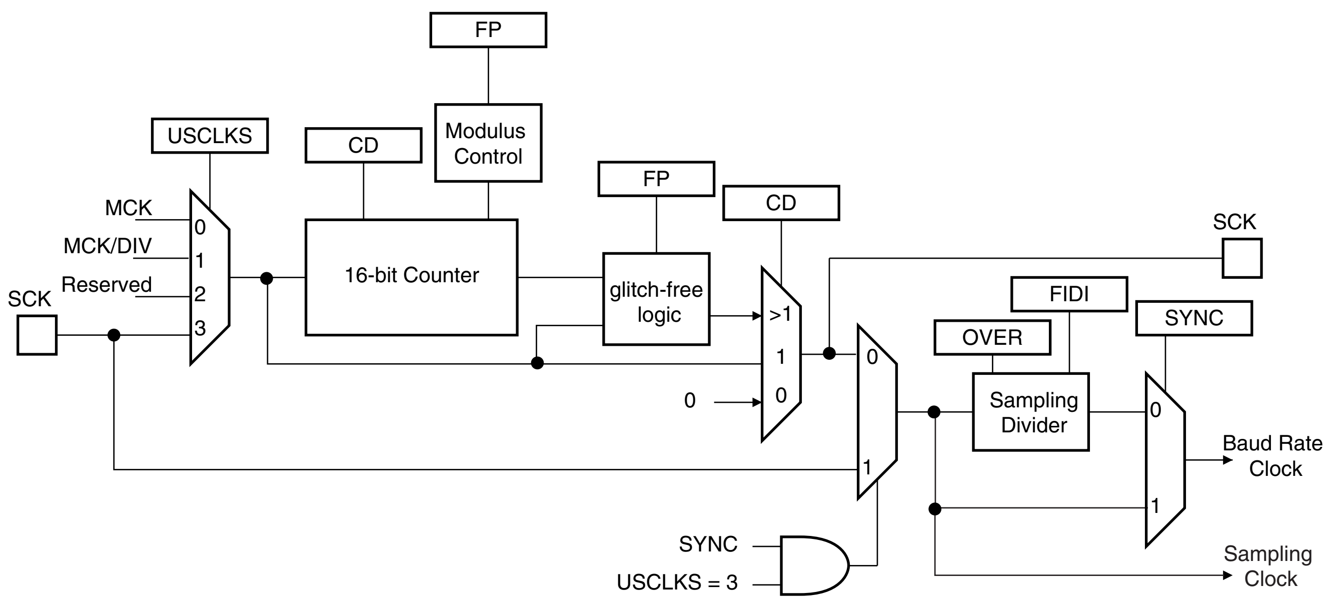
The Baud Rate generator previously defined is subject to the following limitation: the output frequency changes by only integer multiples of the reference frequency. An approach to this problem is to integrate a fractional N clock generator that has a high resolution. The generator architecture is modified 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 36-4. Fractional Baud Rate Generator



36.6.1.4 Baud Rate in Synchronous Mode

If the USART is programmed to operate in synchronous mode, the selected clock is simply divided by the field CD in US_BRGR.

$$BaudRate = \frac{SelectedClock}{CD}$$

In synchronous mode, if the external clock is selected (USCLKS = 3), the clock is provided directly by the signal on the USART SCK pin. No division is active. The value written in US_BRGR has no effect. The external clock frequency must be at least 4.5 times lower than the system clock.

When either the external clock SCK or the internal clock divided (MCK/DIV) is selected, the value programmed in CD must be even if the user has to ensure a 50:50 mark/space ratio on the SCK pin. If the internal clock MCK is selected, the Baud Rate Generator ensures a 50:50 duty cycle on the SCK pin, even if the value programmed in CD is odd.

36.6.1.5 Baud Rate in ISO 7816 Mode

The ISO7816 specification defines the bit rate with the following formula:

$$B = \frac{Di}{Fi} \times f$$

where:

- B is the bit rate
- Di is the bit-rate adjustment factor
- Fi is the clock frequency division factor
- f is the ISO7816 clock frequency (Hz)

Di is a binary value encoded on a 4-bit field, named DI, as represented in Table 36-4.

Table 36-4. 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 36-5.

Table 36-5. 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 36-6 shows the resulting Fi/Di Ratio, which is the ratio between the ISO7816 clock and the baud rate clock.

Table 36-6. 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

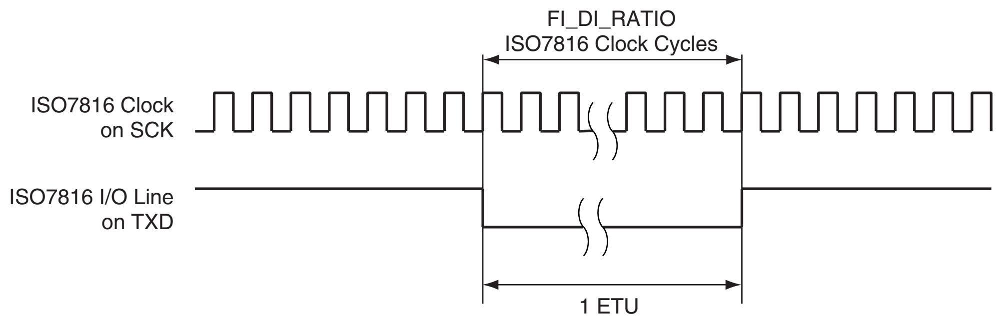
If the USART is configured in ISO7816 Mode, the clock selected by the USCLKS field in the Mode Register (US_MR) is first divided by the value programmed in the field CD in the Baud Rate Generator Register (US_BRGR). The resulting clock can be provided to the SCK pin to feed the smart card clock inputs. 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 36-5 shows the relation between the Elementary Time Unit, corresponding to a bit time, and the ISO 7816 clock.

Figure 36-5. Elementary Time Unit (ETU)



36.6.2 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.

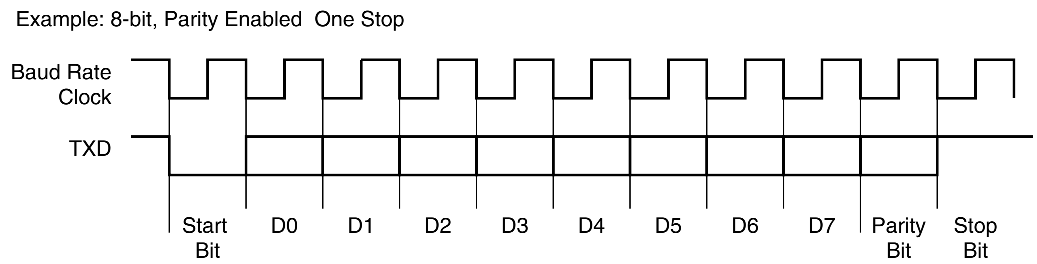
36.6.3 Synchronous and Asynchronous Modes

36.6.3.1 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.

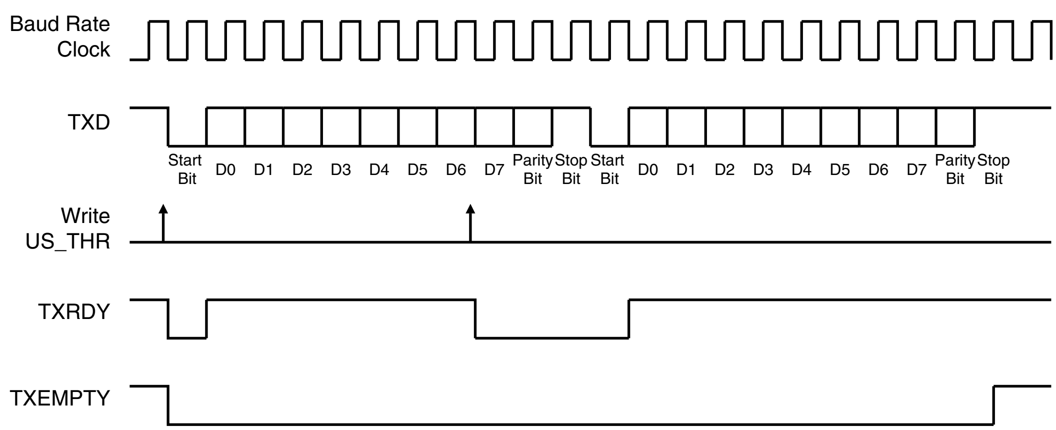
Figure 36-6. Character Transmit



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 rises.

Both TXRDY and TXEMPTY bits are low when the transmitter is disabled. Writing a character in US_THR while TXRDY is low has no effect and the written character is lost.

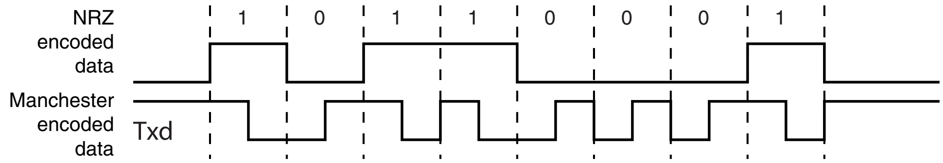
Figure 36-7. Transmitter Status



36.6.3.2 Manchester Encoder

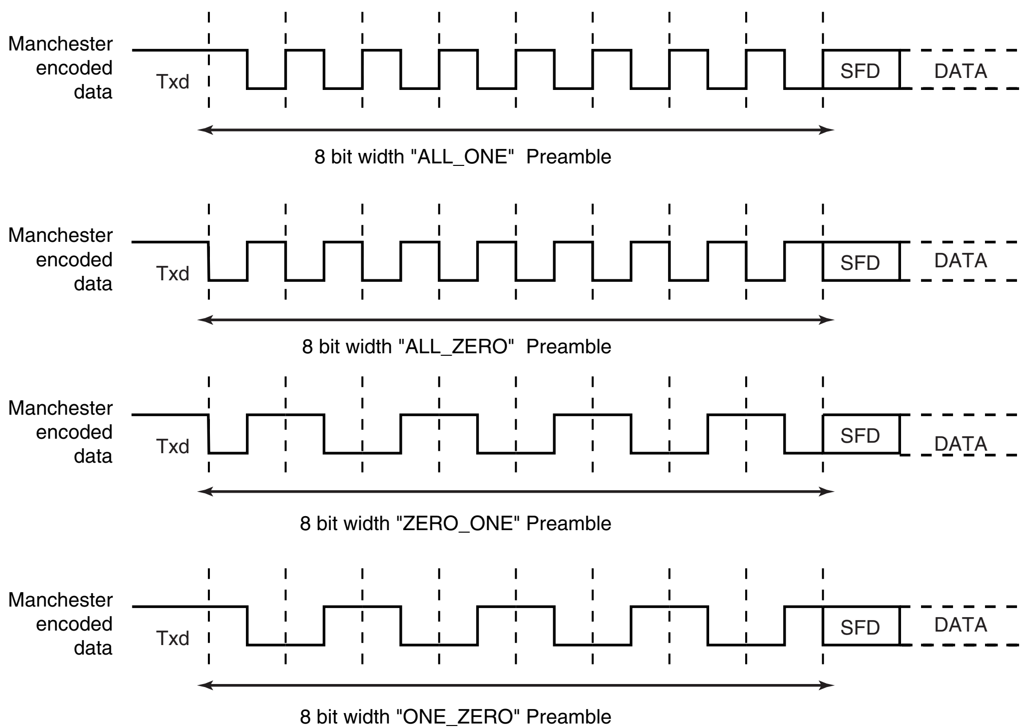
When the Manchester encoder is in use, characters transmitted through the USART are encoded based on biphase Manchester II format. To enable this mode, set the MAN field in the US_MR register to 1. Depending on polarity configuration, a logic level (zero or one), is transmitted as a coded signal one-to-zero 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 36-8 illustrates this coding scheme.

Figure 36-8. NRZ to Manchester Encoding



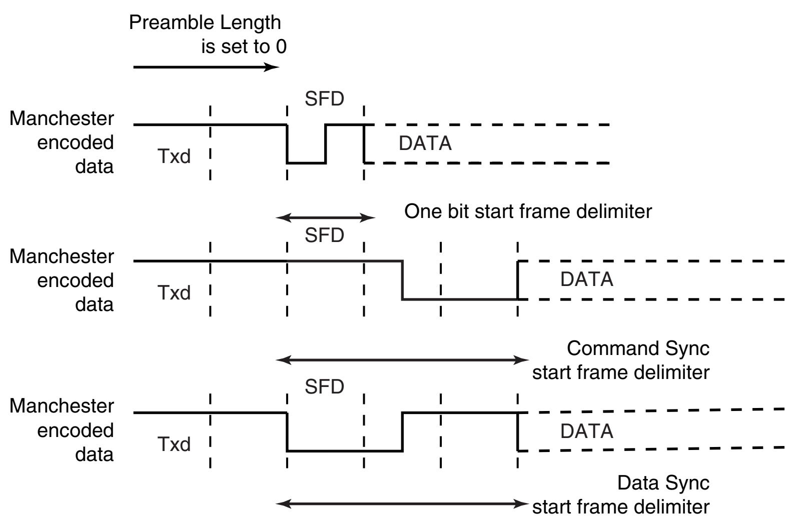
The Manchester encoded character can also be encapsulated by adding both a configurable preamble and a start frame delimiter pattern. Depending on the configuration, the preamble is a training sequence, composed of a pre-defined pattern with a programmable length from 1 to 15 bit times. If the preamble 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 36-9 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 36-9. 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 36-10 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.

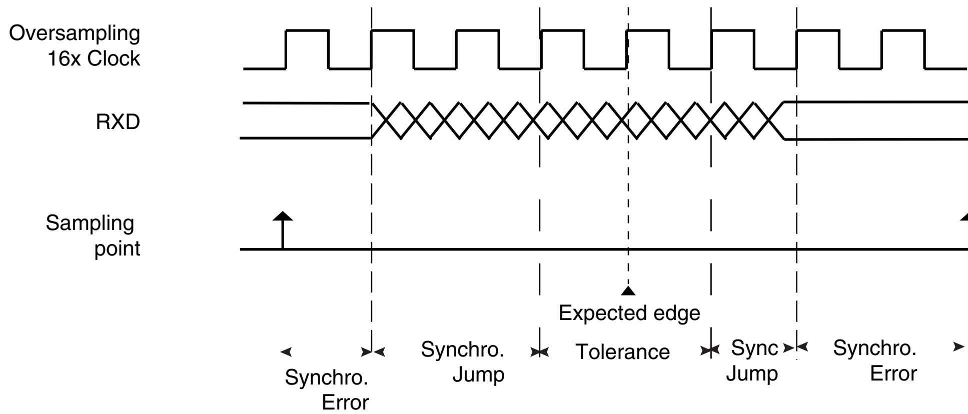
Figure 36-10. Start Frame Delimiter



36.6.3.3 *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 36-11. Bit Resynchronization



36.6.3.4 *Asynchronous Receiver*

If the USART is programmed in asynchronous operating mode (SYNC = 0), the receiver over-samples 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 36-12 and Figure 36-13 illustrate start detection and character reception when USART operates in asynchronous mode.

Figure 36-12. Asynchronous Start Detection

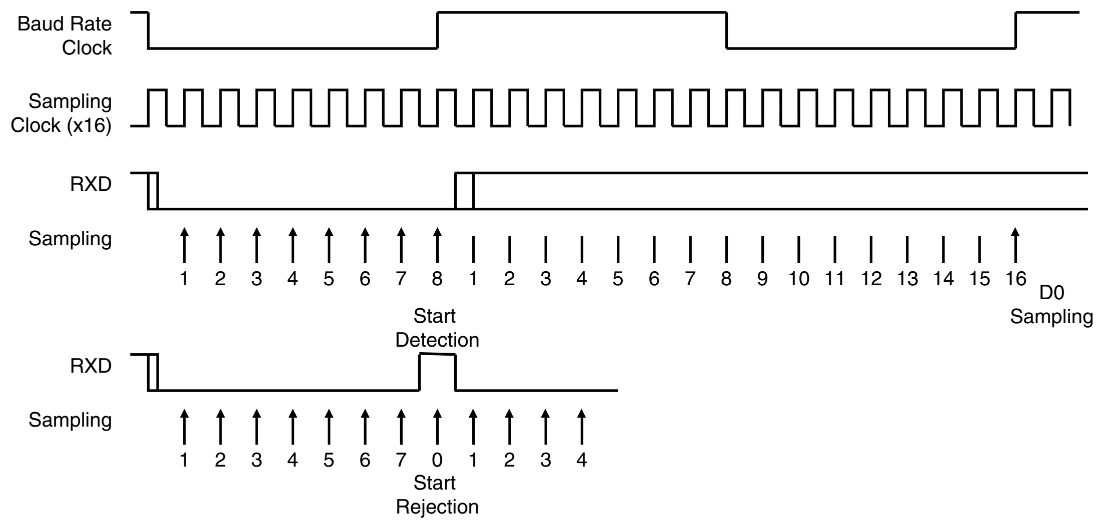
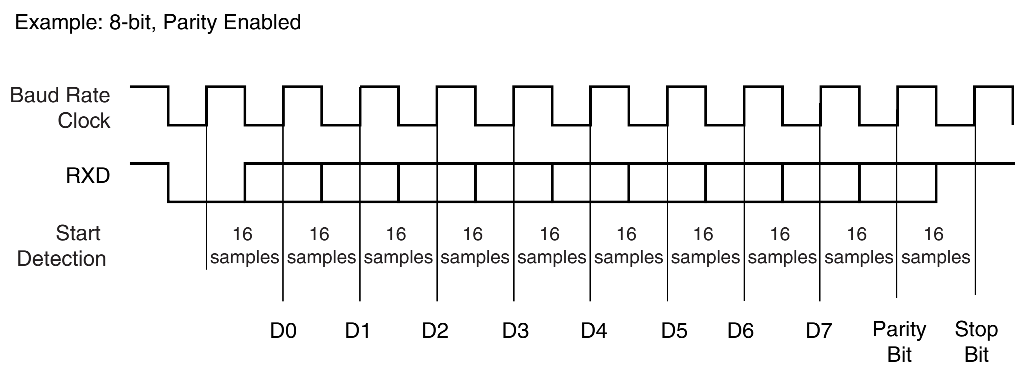


Figure 36-13. Asynchronous Character Reception



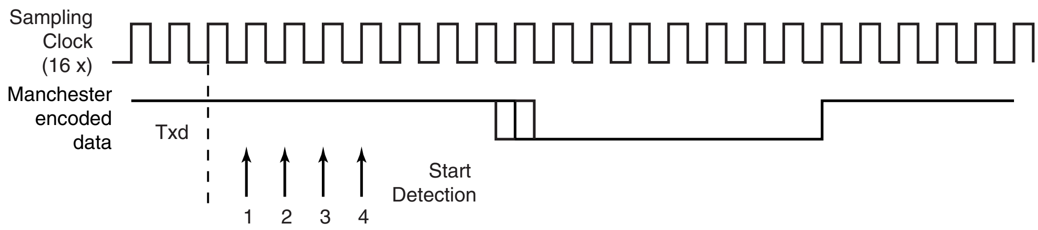
36.6.3.5 *Manchester Decoder*

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 36-9 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 36-14. The sample pulse rejection mechanism applies.

Figure 36-14. 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 36-15 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 36-16 for an example of Manchester error detection during data phase.

Figure 36-15. Preamble Pattern Mismatch

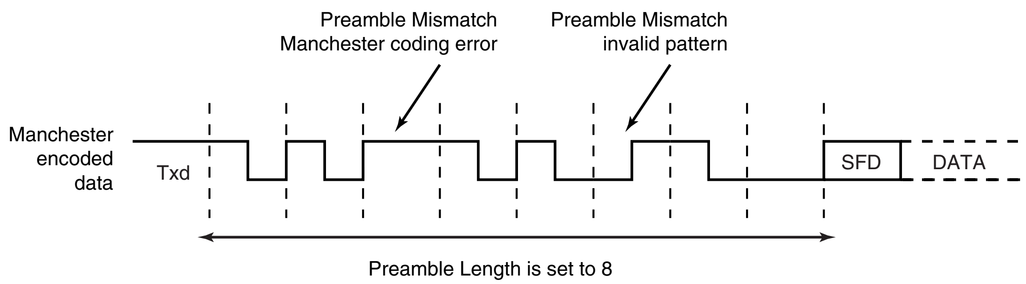
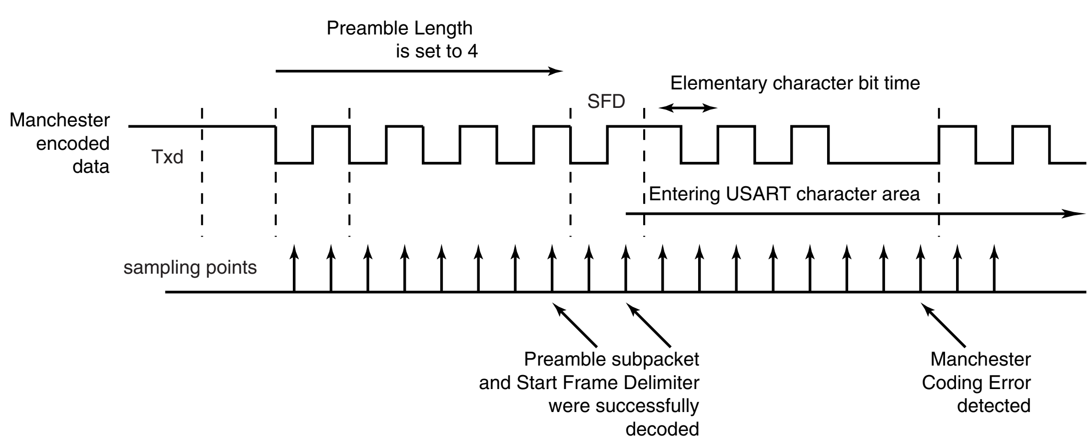


Figure 36-16. Manchester Error Flag



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.

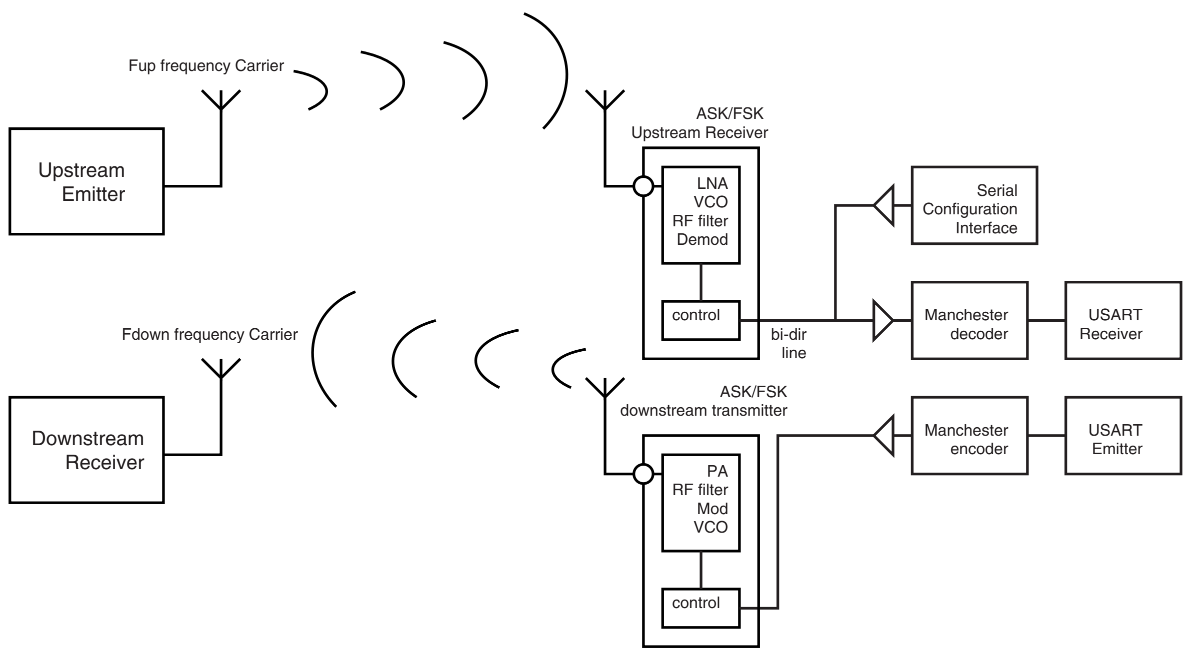
As the decoder is setup to be used in unipolar mode, the first bit of the frame has to be a zero-to-one transition.

36.6.3.6 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 36-17.

Figure 36-17. 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 36-18 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 36-19.

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 36-18. ASK Modulator Output

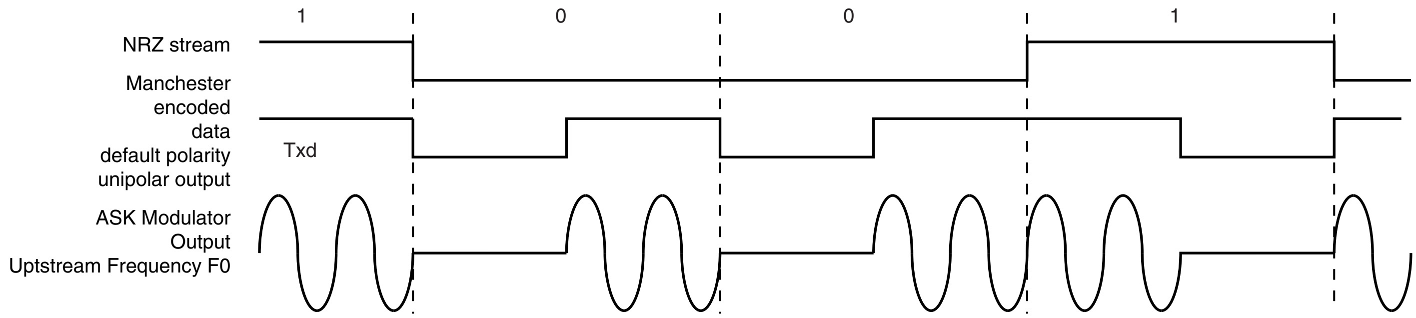
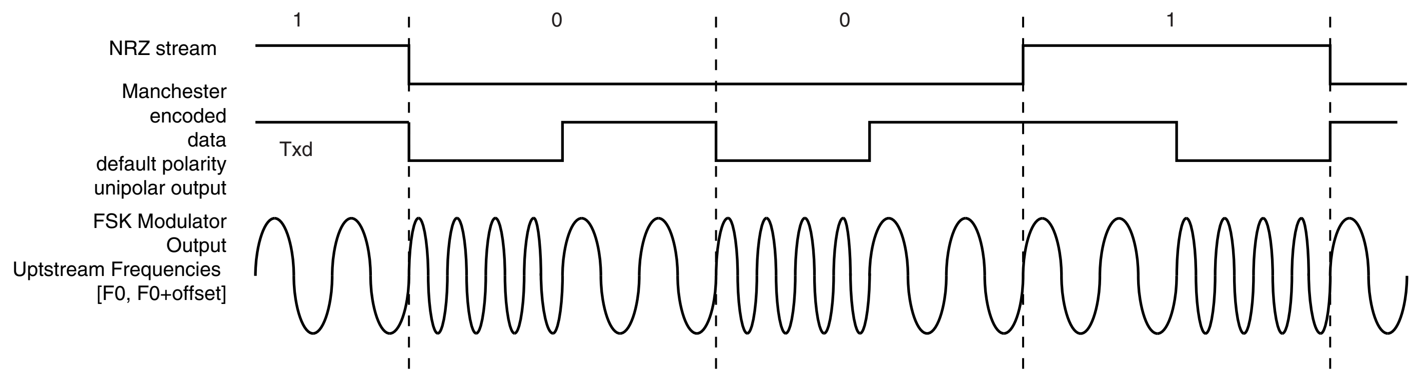


Figure 36-19. FSK Modulator Output



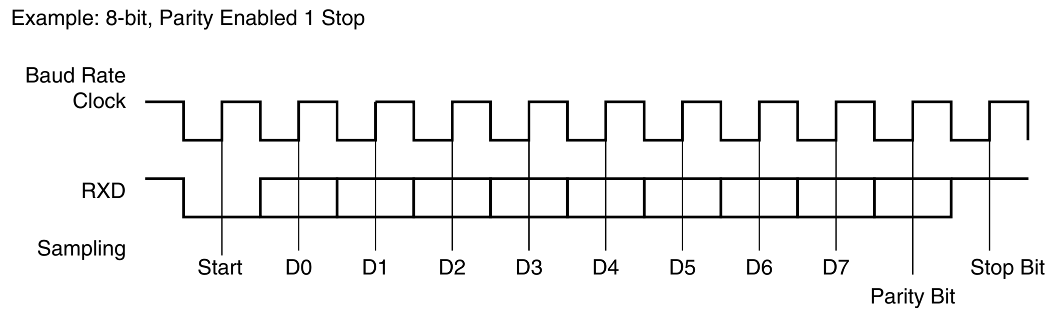
36.6.3.7 *Synchronous Receiver*

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 36-20 illustrates a character reception in synchronous mode.

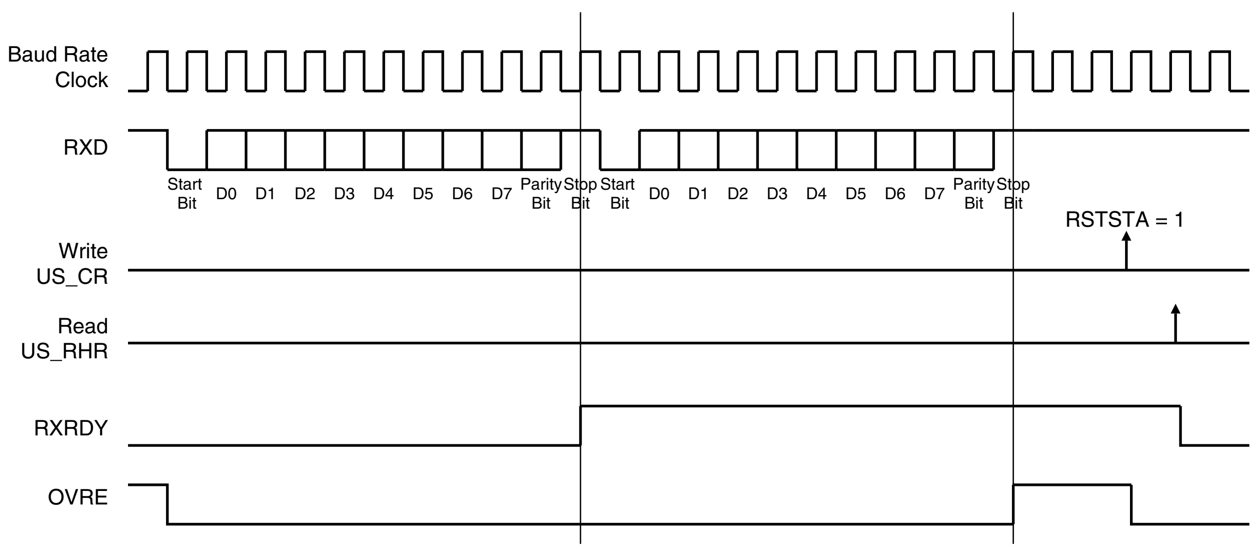
Figure 36-20. Synchronous Mode Character Reception



36.6.3.8 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 36-21. Receiver Status



36.6.3.9 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 591. 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.

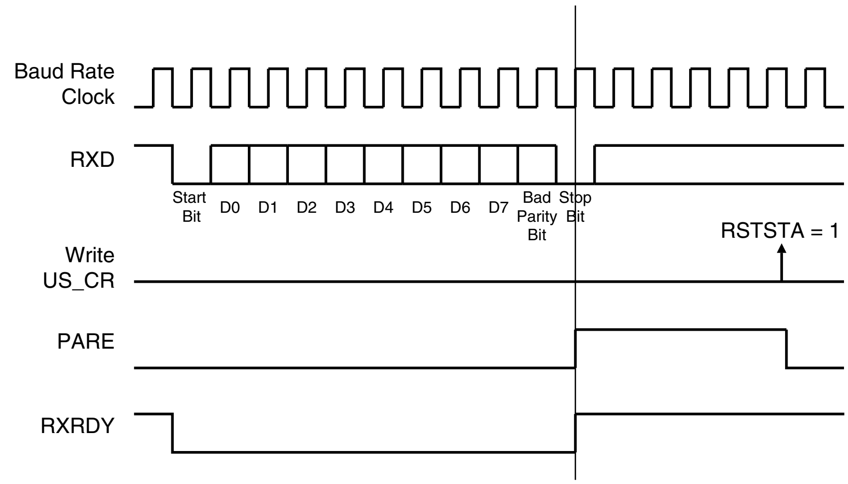
Table 36-7 shows an example of the parity bit for the character 0x41 (character ASCII "A") depending on the configuration of the USART. Because there are two bits at 1, 1 bit is added when a parity is odd, or 0 is added when a parity is even.

Table 36-7. 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 36-22 illustrates the parity bit status setting and clearing.

Figure 36-22. Parity Error



36.6.3.10 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.

36.6.3.11 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 36-23, 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 36-23. Timeguard Operations

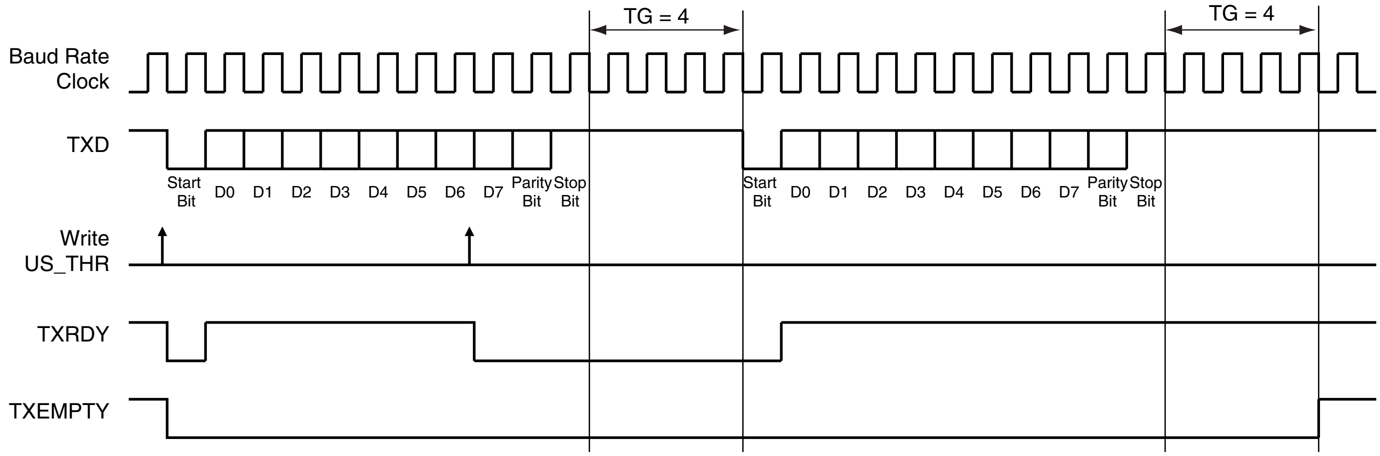


Table 36-8 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the Baud Rate.

Table 36-8. 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

36.6.3.12 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 36-24 shows the block diagram of the Receiver Time-out feature.

Figure 36-24. Receiver Time-out Block Diagram

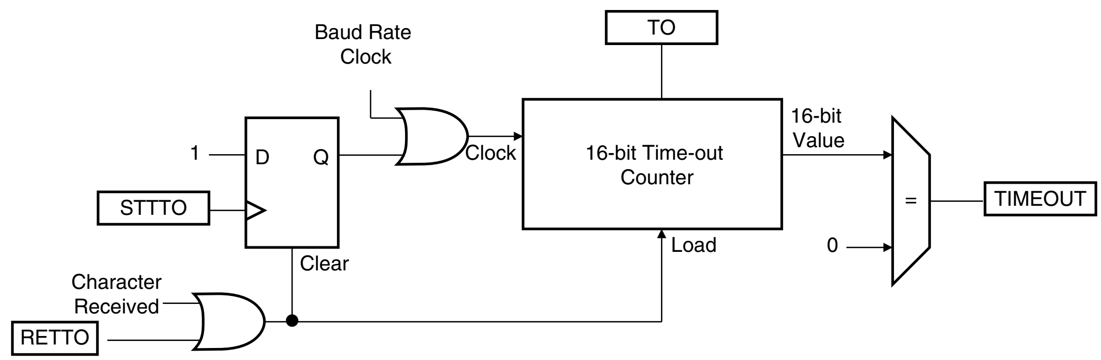


Table 36-9 gives the maximum time-out period for some standard baud rates.

Table 36-9. 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 36-9. Maximum Time-out Period (Continued)

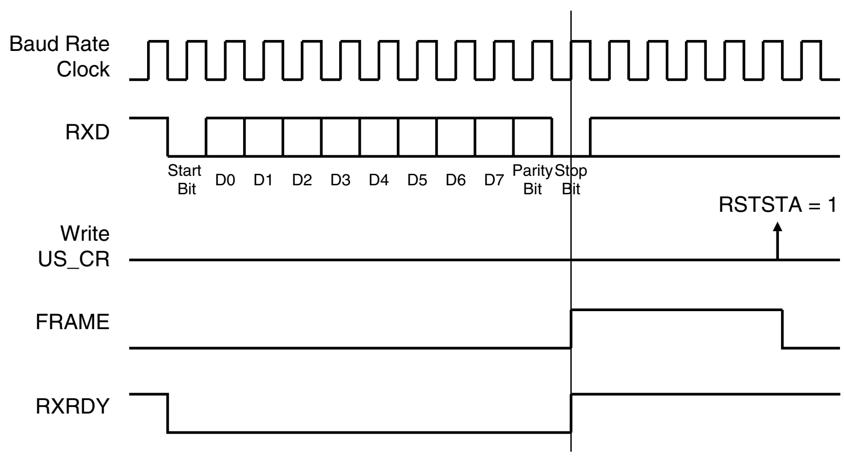
Baud Rate	Bit Time	Time-out
56000	18	1 170
57600	17	1 138
200000	5	328

36.6.3.13 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 36-25. Framing Error Status



36.6.3.14 Transmit Break

The user can request the transmitter to generate a break condition on the TXD line. A break condition drives the TXD line low during at least one complete character. It appears the same as a 0x00 character sent with the parity and the stop bits at 0. However, the transmitter holds the TXD line at least during one character until the user requests the break condition to be removed.

A break is transmitted by writing the Control Register (US_CR) with the STTBRK bit at 1. This can be performed at any time, either while the transmitter is empty (no character in either the Shift Register or in US_THR) or when a character is being transmitted. If a break is requested while a character is being shifted out, the character is first completed before the TXD line is held low.

Once STTBRK command is requested further STTBRK commands are ignored until the end of the break is completed.

The break condition is removed by writing US_CR with the STPBRK bit at 1. If the STPBRK is requested before the end of the minimum break duration (one character, including start, data, parity and stop bits), the transmitter ensures that the break condition completes.

The transmitter considers the break as though it is a character, i.e. the STTBRK and STPBRK commands are taken into account only if the TXRDY bit in US_CSR is at 1 and the start of the break condition clears the TXRDY and TXEMPTY bits as if a character is processed.

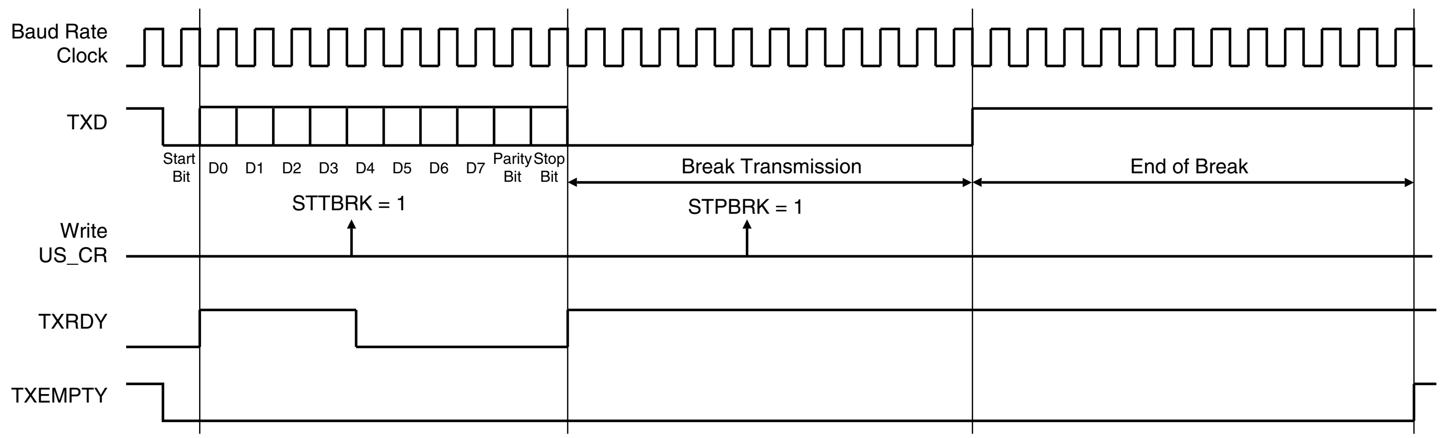
Writing US_CR with the both STTBRK and STPBRK bits at 1 can lead to an unpredictable result. All STPBRK commands requested without a previous STTBRK command are ignored. A byte written into the Transmit Holding Register while a break is pending, but not started, is ignored.

After the break condition, the transmitter returns the TXD line to 1 for a minimum of 12 bit times. Thus, the transmitter ensures that the remote receiver detects correctly the end of break and the start of the next character. If the timeguard is programmed with a value higher than 12, the TXD line is held high for the timeguard period.

After holding the TXD line for this period, the transmitter resumes normal operations.

Figure 36-26 illustrates the effect of both the Start Break (STTBRK) and Stop Break (STPBRK) commands on the TXD line.

Figure 36-26. Break Transmission



36.6.3.15 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.

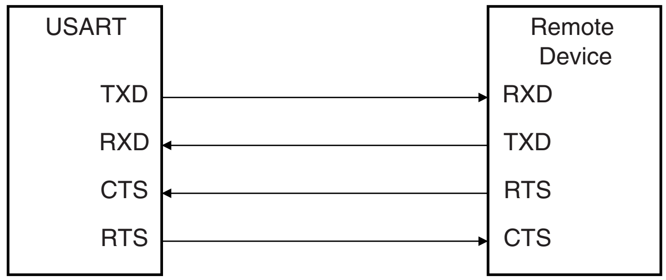
When the low stop bit is detected, the receiver asserts the RXBRK bit in US_CSR. This bit may be cleared by writing the Control Register (US_CR) with the bit RSTSTA at 1.

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.

36.6.3.16 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 36-27.

Figure 36-27. 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 36-28 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 36-28. Receiver Behavior when Operating with Hardware Handshaking

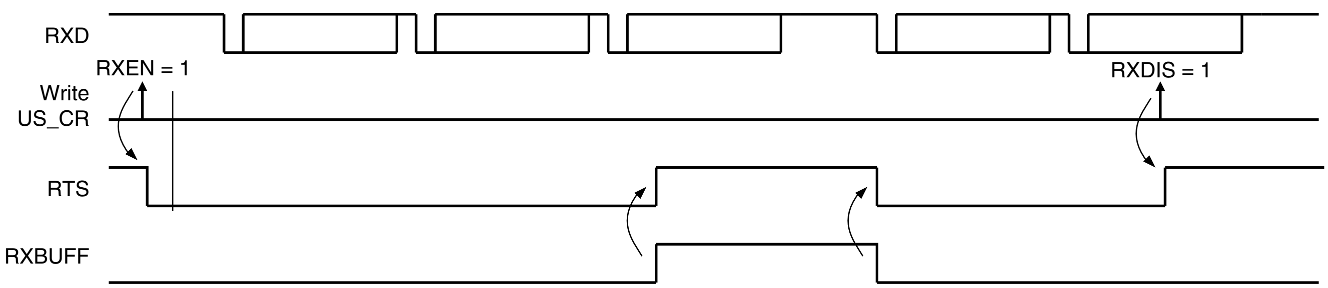
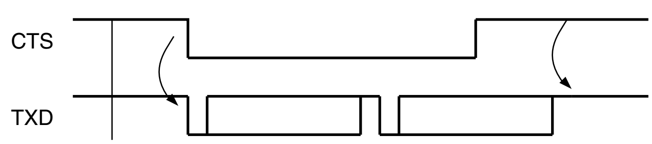


Figure 36-29 shows how the transmitter operates if hardware handshaking is enabled. The CTS pin disables the transmitter. If a character is being processing, the transmitter is disabled only after the completion of the current character and transmission of the next character happens as soon as the pin CTS falls.

Figure 36-29. Transmitter Behavior when Operating with Hardware Handshaking



36.6.4 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.

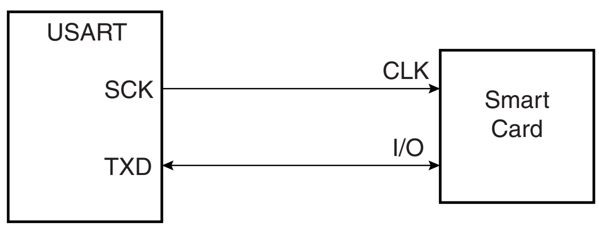
Setting the USART in ISO7816 mode is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x4 for protocol T = 0 and to the value 0x5 for protocol T = 1.

36.6.4.1 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 576).

The USART connects to a smart card as shown in Figure 36-30. 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 36-30. 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 608 and "PAR: Parity Type" on page 609.

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).

36.6.4.2 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 36-31.

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 36-32. 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 36-31. T = 0 Protocol without Parity Error

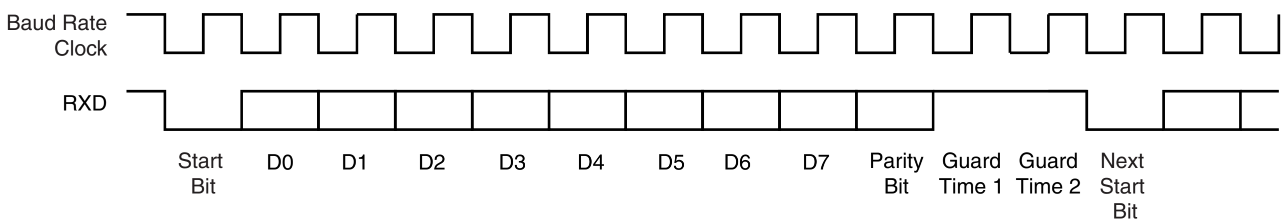
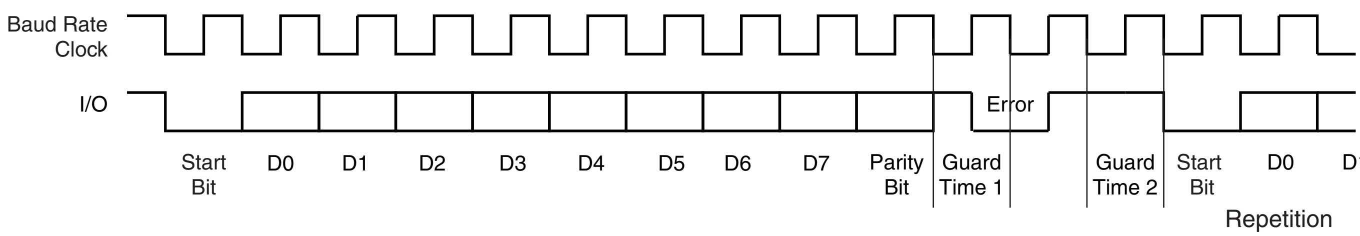


Figure 36-32. T = 0 Protocol with Parity Error



36.6.4.3 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.

36.6.4.4 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.

36.6.4.5 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.

36.6.4.6 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.

36.6.4.7 Protocol T = 1

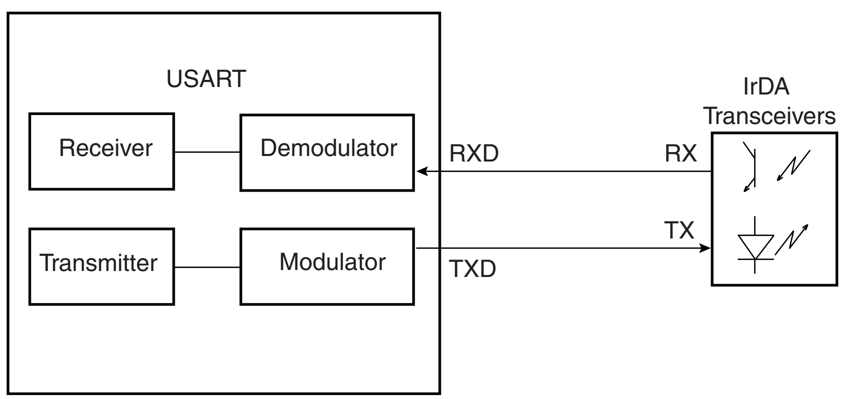
When operating in ISO7816 protocol T = 1, the transmission is similar to an asynchronous format with only one stop bit. The parity is generated when transmitting and checked when receiving. Parity error detection sets the PARE bit in the Channel Status Register (US_CSR).

36.6.5 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 36-33. 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 36-33. Connection to IrDA Transceivers



The receiver and the transmitter must be enabled or disabled according to the direction of the transmission to be managed.

To receive IrDA signals, the following needs to be done:

- Disable TX and Enable RX

- Configure the TXD pin as PIO and set it as an output at 0 (to avoid LED emission). Disable the internal pull-up (better for power consumption).
- Receive data

36.6.5.1 IrDA Modulation

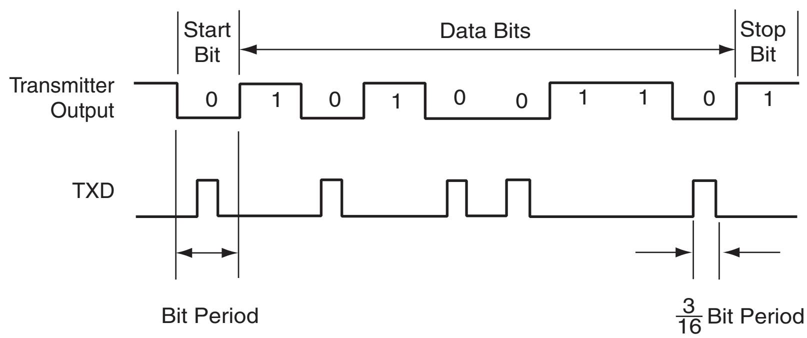
For baud rates up to and including 115.2 Kbits/sec, the RZI modulation scheme is used. “0” is represented by a light pulse of 3/16th of a bit time. Some examples of signal pulse duration are shown in Table 36-10.

Table 36-10. IrDA Pulse Duration

Baud Rate	Pulse Duration (3/16)
2.4 Kb/s	78.13 µs
9.6 Kb/s	19.53 µs
19.2 Kb/s	9.77 µs
38.4 Kb/s	4.88 µs
57.6 Kb/s	3.26 µs
115.2 Kb/s	1.63 µs

Figure 36-34 shows an example of character transmission.

Figure 36-34. IrDA Modulation



36.6.5.2 IrDA Baud Rate

Table 36-11 gives some examples of CD values, baud rate error and pulse duration. Note that the requirement on the maximum acceptable error of ±1.87% must be met.

Table 36-11. 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

Table 36-11. IrDA Baud Rate Error (Continued)

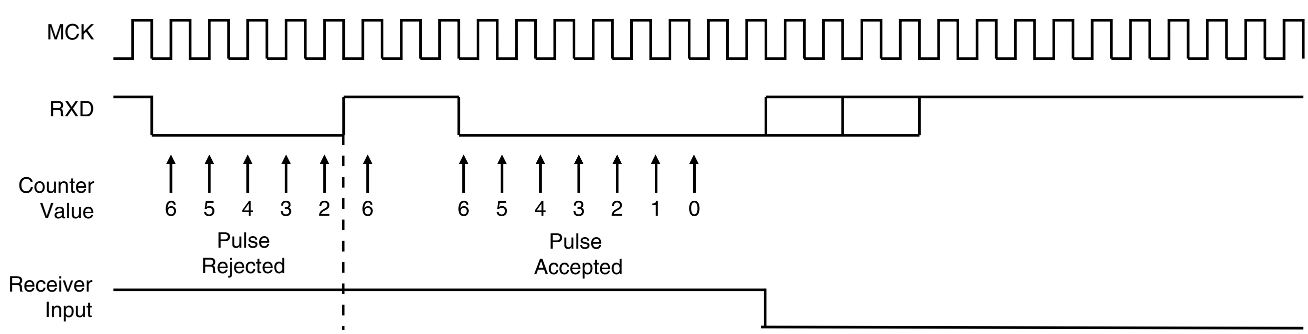
Peripheral Clock	Baud Rate	CD	Baud Rate Error	Pulse Time
40 000 000	57 600	43	0.93%	3.26
3 686 400	38 400	6	0.00%	4.88
20 000 000	38 400	33	1.38%	4.88
32 768 000	38 400	53	0.63%	4.88
40 000 000	38 400	65	0.16%	4.88
3 686 400	19 200	12	0.00%	9.77
20 000 000	19 200	65	0.16%	9.77
32 768 000	19 200	107	0.31%	9.77
40 000 000	19 200	130	0.16%	9.77
3 686 400	9 600	24	0.00%	19.53
20 000 000	9 600	130	0.16%	19.53
32 768 000	9 600	213	0.16%	19.53
40 000 000	9 600	260	0.16%	19.53
3 686 400	2 400	96	0.00%	78.13
20 000 000	2 400	521	0.03%	78.13
32 768 000	2 400	853	0.04%	78.13

36.6.5.3 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 36-35 illustrates the operations of the IrDA demodulator.

Figure 36-35. IrDA Demodulator Operations

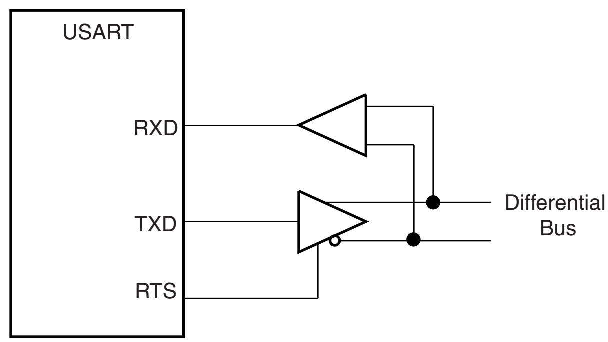


As the IrDA mode uses the same logic as the ISO7816, note that the FI_DI_RATIO field in US_FIDI must be set to a value higher than 0 in order to assure IrDA communications operate correctly.

36.6.6 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 36-36.

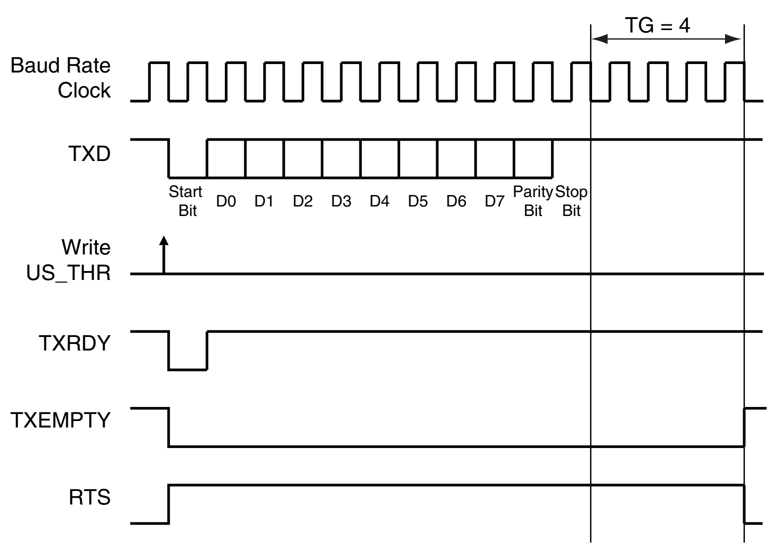
Figure 36-36. Typical Connection to a RS485 Bus



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 36-37 gives an example of the RTS waveform during a character transmission when the timeguard is enabled.

Figure 36-37. Example of RTS Drive with Timeguard



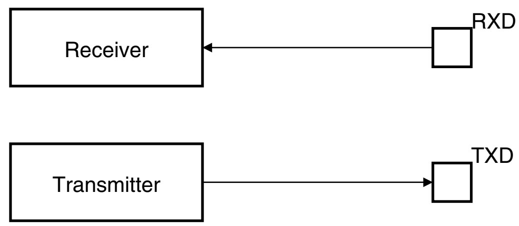
36.6.7 Test Modes

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.

36.6.7.1 Normal Mode

Normal mode connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

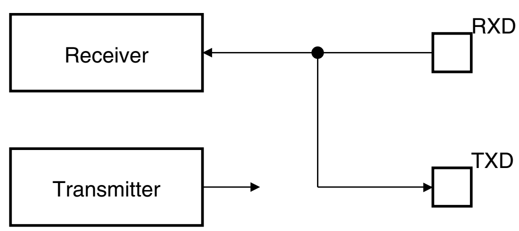
Figure 36-38. Normal Mode Configuration



36.6.7.2 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 36-39. 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.

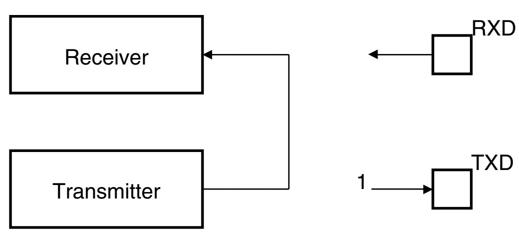
Figure 36-39. Automatic Echo Mode Configuration



36.6.7.3 Local Loopback Mode

Local loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in Figure 36-40. 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.

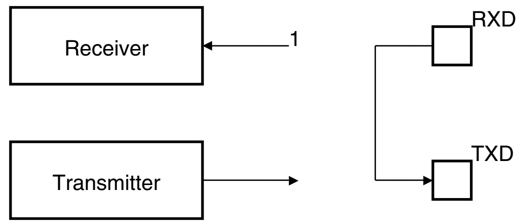
Figure 36-40. Local Loopback Mode Configuration



36.6.7.4 Remote Loopback Mode

Remote loopback mode directly connects the RXD pin to the TXD pin, as shown in Figure 36-41. The transmitter and the receiver are disabled and have no effect. This mode allows bit-by-bit retransmission.

Figure 36-41. Remote Loopback Mode Configuration



36.7 Universal Synchronous Asynchronous Receiver Transmitter (USART) User Interface

Table 36-13. Register Mapping

Offset	Register	Name	Access	Reset
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	–	–	–

36.7.1 USART Control Register

Name: US_CR

Addresses: 0xFFF8C000 (0), 0xFFF90000 (1), 0xFFF94000 (2)

Access: Write-only

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

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

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

- **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.

36.7.2 USART Mode Register

Name: US_MR

Addresses: 0xFFF8C004 (0), 0xFFF90004 (1), 0xFFF94004 (2)

Access: 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	1	0	0	IS07816 Protocol: T = 0
0	1	1	0	IS07816 Protocol: T = 1
1	0	0	0	IrDA
Others				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.

36.7.3 USART Interrupt Enable Register

Name: US_IER

Addresses: 0xFFF8C008 (0), 0xFFF90008 (1), 0xFFF94008 (2)

Access: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITER	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**
- **ITER: Iteration Interrupt Enable**
- **TXBUFE: Buffer Empty Interrupt Enable**
- **RXBUFF: Buffer Full Interrupt Enable**
- **NACK: Non Acknowledge Interrupt Enable**
- **CTSIC: Clear to Send Input Change Interrupt Enable**
- **MANE: Manchester Error Interrupt Enable**

36.7.4 USART Interrupt Disable Register

Name: US_IDR

Addresses: 0xFFF8C00C (0), 0xFFF9000C (1), 0xFFF9400C (2)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	MANE	CTSIC	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	NACK	RXBUFF	TXBUFE	ITER	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**
- **ITER: Iteration Interrupt Enable**
- **TXBUFE: Buffer Empty Interrupt Disable**
- **RXBUFF: Buffer Full Interrupt Disable**
- **NACK: Non Acknowledge Interrupt Disable**
- **CTSIC: Clear to Send Input Change Interrupt Disable**
- **MANE: Manchester Error Interrupt Disable**

36.7.5 USART Interrupt Mask Register

Name: US_IMR

Addresses: 0xFFF8C010 (0), 0xFFF90010 (1), 0xFFF94010 (2)

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITER	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**
- **ITER: Iteration Interrupt Enable**
- **TXBUFE: Buffer Empty Interrupt Mask**
- **RXBUFF: Buffer Full Interrupt Mask**
- **NACK: Non Acknowledge Interrupt Mask**
- **CTSIC: Clear to Send Input Change Interrupt Mask**
- **MANE: Manchester Error Interrupt Mask**

36.7.6 USART Channel Status Register

Name: US_CSR

Addresses: 0xFFF8C014 (0), 0xFFF90014 (1), 0xFFF94014 (2)

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITER	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.

• **ITER: Max number of Repetitions Reached**

0: Maximum number of repetitions has not been reached since the last RSTSTA.

1: Maximum number of repetitions has been reached since the last RSTSTA.

• **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.

• **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.

• **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.

36.7.7 USART Receive Holding Register

Name: US_RHR

Addresses: 0xFFF8C018 (0), 0xFFF90018 (1), 0xFFF94018 (2)

Access: Read-only

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

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

15	14	13	12	11	10	9	8
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.

36.7.8 USART Transmit Holding Register

Name: US_THR

Addresses: 0xFFF8C01C (0), 0xFFF9001C (1), 0xFFF9401C (2)

Access: Write-only

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

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

15	14	13	12	11	10	9	8
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.

36.7.9 USART Baud Rate Generator Register

Name: US_BRGR

Addresses: 0xFFF8C020 (0), 0xFFF90020 (1), 0xFFF94020 (2)

Access: 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 x 1/8.

36.7.10 USART Receiver Time-out Register

Name: US_RTOR

Addresses: 0xFFF8C024 (0), 0xFFF90024 (1), 0xFFF94024 (2)

Access: 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.

36.7.11 USART Transmitter Timeguard Register

Name: US_TTGR

Addresses: 0xFFF8C028 (0), 0xFFF90028 (1), 0xFFF94028 (2)

Access: 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.

36.7.12 USART FI DI RATIO Register

Name: US_FIDI

Addresses: 0xFFF8C040 (0), 0xFFF90040 (1), 0xFFF94040 (2)

Access: 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.

36.7.13 USART Number of Errors Register

Name: US_NER

Addresses: 0xFFF8C044 (0), 0xFFF90044 (1), 0xFFF94044 (2)

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
NB_ERRORS							

- **NB_ERRORS: Number of Errors**

Total number of errors that occurred during an ISO7816 transfer. This register automatically clears when read.

36.7.14 USART IrDA FILTER Register

Name: US_IF

Addresses: 0xFFF8C04C (0), 0xFFF9004C (1), 0xFFF9404C (2)

Access: 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.

36.7.15 USART Manchester Configuration Register

Name: US_MAN

Addresses: 0xFFF8C050 (0), 0xFFF90050 (1), 0xFFF94050 (2)

Access: Read-write

31	30	29	28	27	26	25	24
–	DRIFT	1	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			

This register can only be written if the WPEN bit is cleared in "USART Write Protect Mode Register" on page 103.

- **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.

37. Synchronous Serial Controller (SSC)

37.1 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.

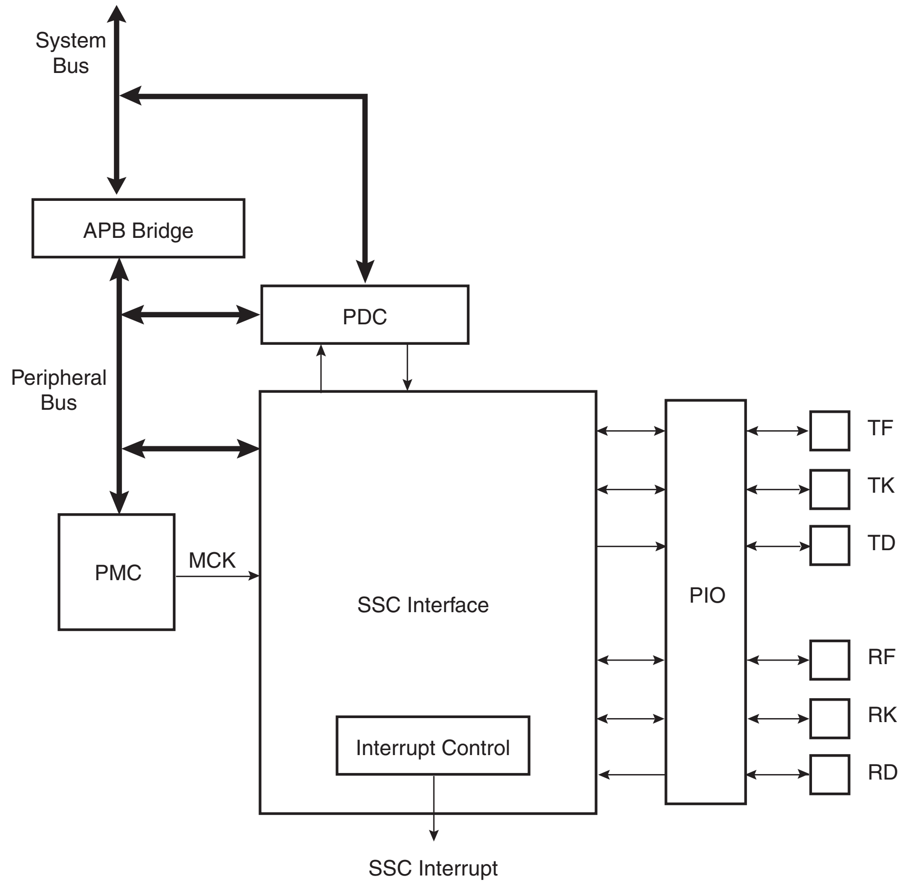
The SSC's high-level of programmability and its two dedicated PDC channels of up to 32 bits permit a continuous high bit rate data transfer without processor intervention.

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

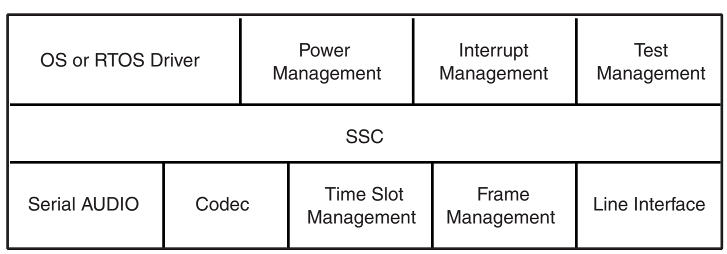
37.2 Block Diagram

Figure 37-1. Block Diagram



37.3 Application Block Diagram

Figure 37-2. Application Block Diagram



37.4 Pin Name List

Table 37-1. 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

37.5 Product Dependencies

37.5.1 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.

Table 37-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
SSC0	RD0	PB3	A
SSC0	RF0	PB5	A
SSC0	RK0	PB4	A
SSC0	TD0	PB2	A
SSC0	TF0	PB0	A
SSC0	TK0	PB1	A
SSC1	RD1	PB9	A
SSC1	RF1	PB11	A
SSC1	RK1	PB10	A
SSC1	TD1	PB8	A
SSC1	TF1	PB6	A
SSC1	TK1	PB7	A

37.5.2 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.

37.5.3 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.

Table 37-3. Peripheral IDs

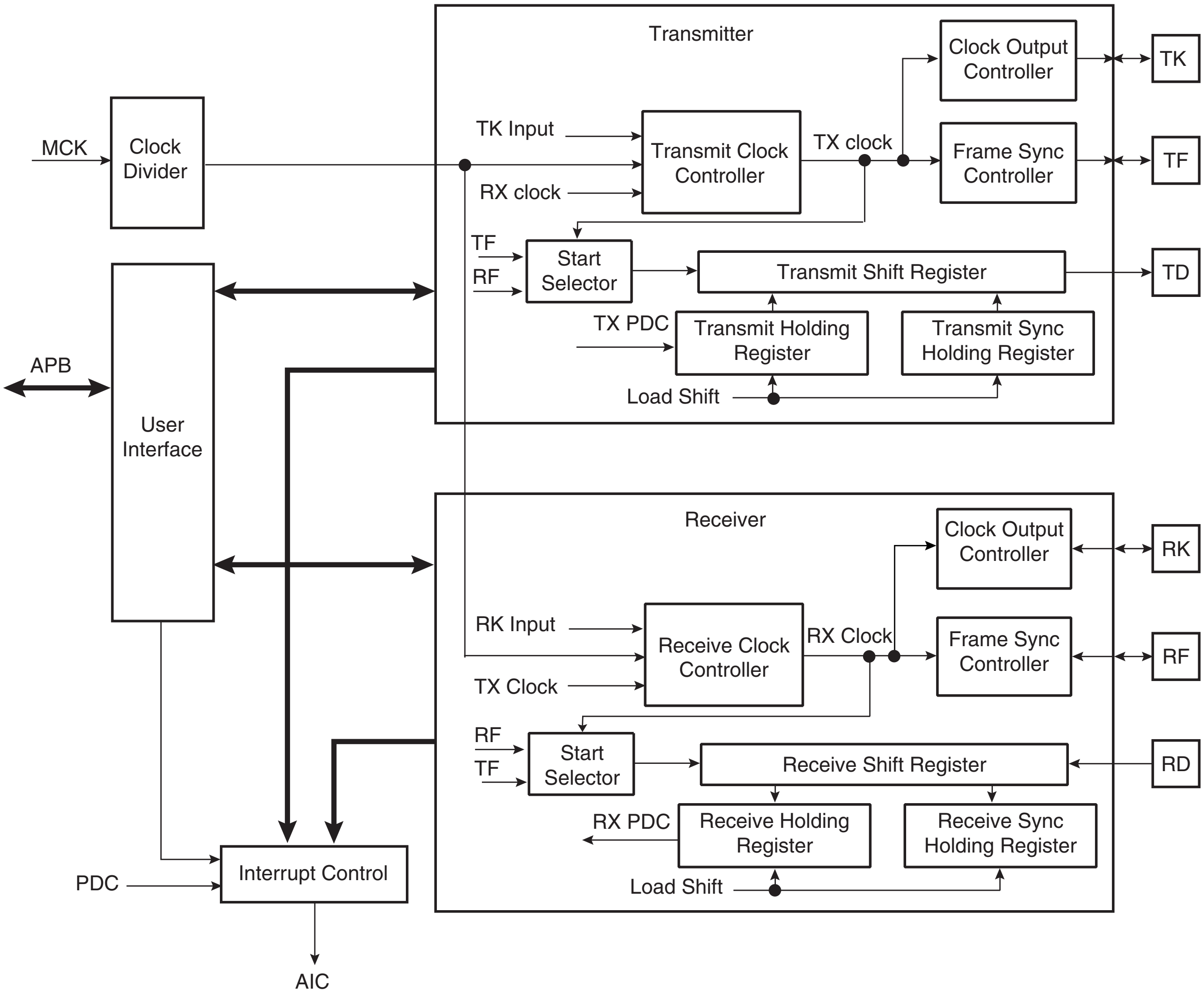
Instance	ID
SSC0	17
SSC1	18

37.6 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 37-3. SSC Functional Block Diagram



37.6.1 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:

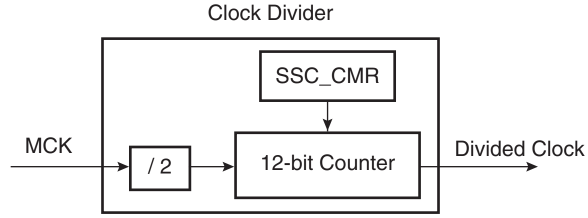
- an external clock received on the RK I/O pad
- the transmitter clock
- the internal clock divider

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.

37.6.1.1 Clock Divider

Figure 37-4. 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_CMR, 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 37-5. Divided Clock Generation

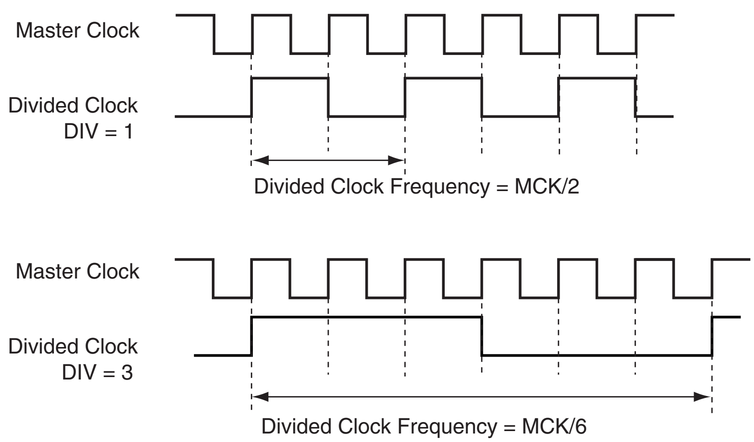


Table 37-4.

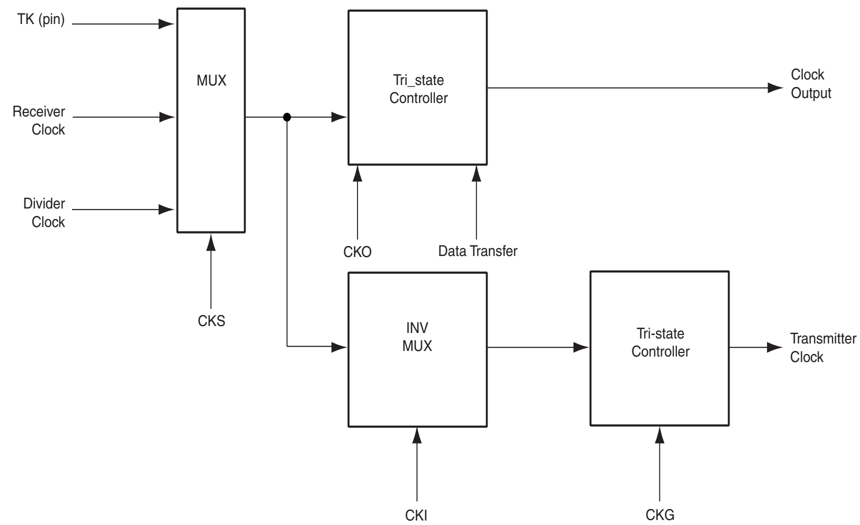
Maximum	Minimum
MCK / 2	MCK / 8190

37.6.1.2 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 37-6. Transmitter Clock Management

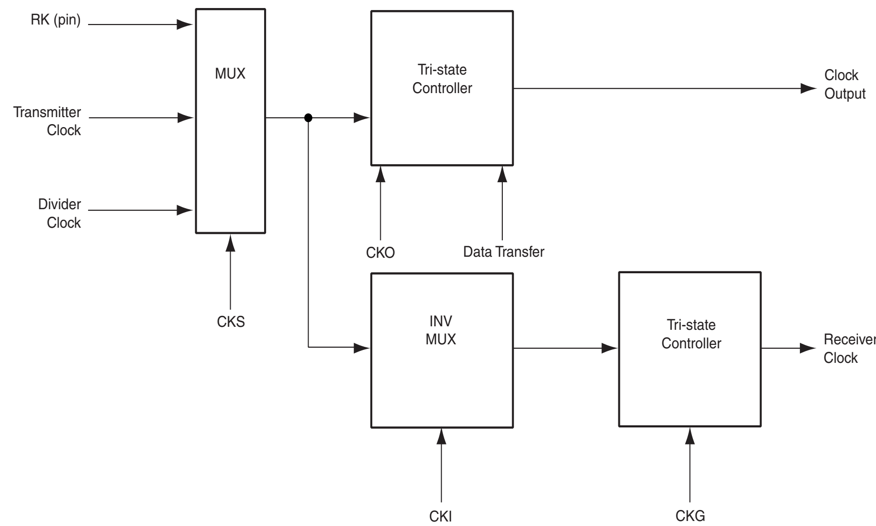


37.6.1.3 *Receiver Clock Management*

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 37-7. Receiver Clock Management



37.6.1.4 *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

37.6.2 Transmitter Operations

A transmitted frame is triggered by a start event and can be followed by synchronization data before data transmission.

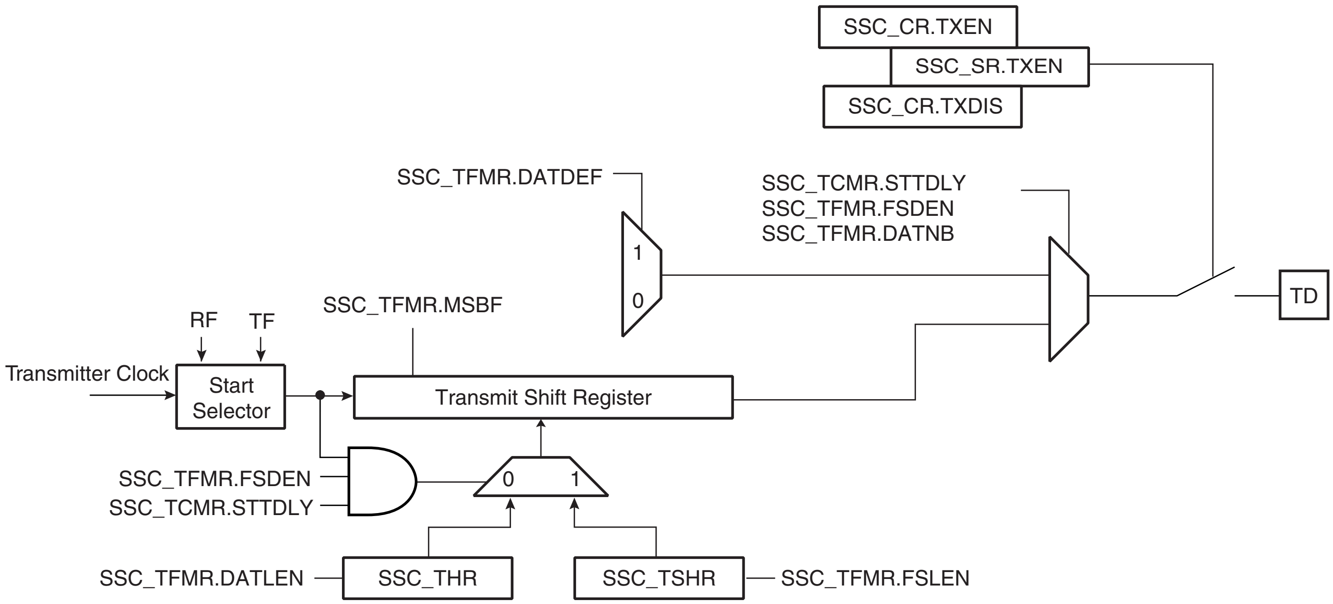
The start event is configured by setting the Transmit Clock Mode Register (SSC_TCMR). See “Start” on page 634.

The frame synchronization is configured setting the Transmit Frame Mode Register (SSC_TFMR). See “Frame Sync” on page 636.

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 37-8. Transmitter Block Diagram



37.6.3 Receiver Operations

A received frame is triggered by a start event and can be followed by synchronization data before data transmission.

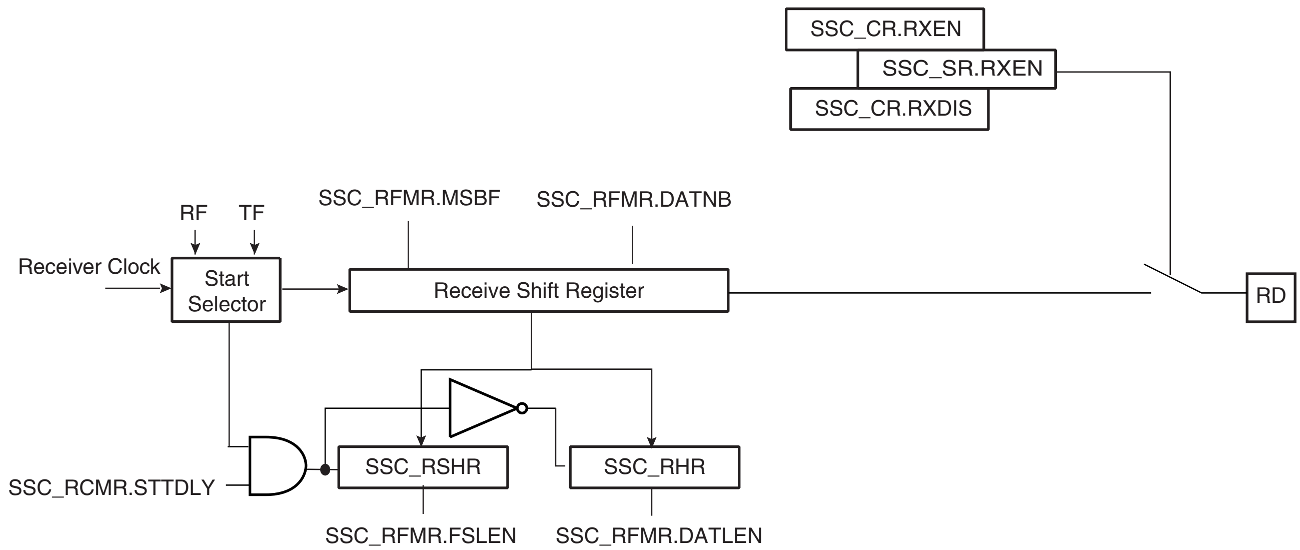
The start event is configured setting the Receive Clock Mode Register (SSC_RCMR). See "Start" on page 634.

The frame synchronization is configured setting the Receive Frame Mode Register (SSC_RFMR). See "Frame Sync" on page 636.

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 37-9. Receiver Block Diagram



37.6.4 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 37-10. Transmit Start Mode

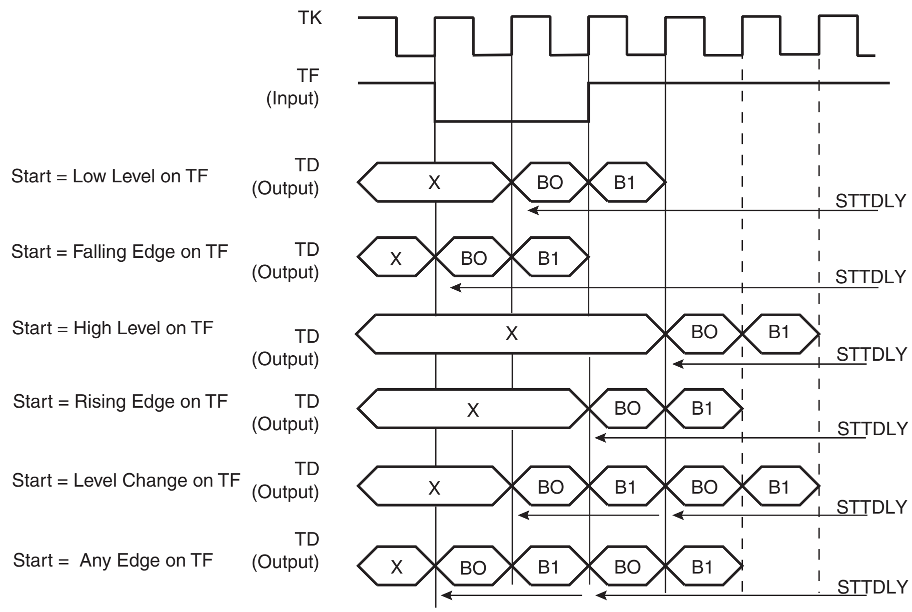
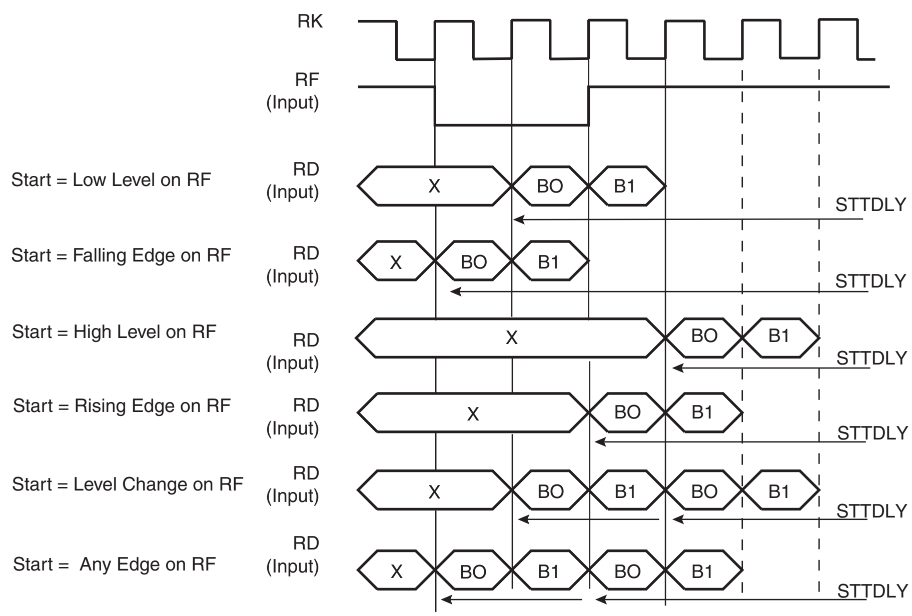


Figure 37-11. Receive Pulse/Edge Start Modes



37.6.5 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 16 bit time.

The periodicity of the Receive and Transmit Frame Sync pulse output can be programmed through the Period Divider Selection (PERIOD) field in SSC_RCMR and SSC_TCMR.

37.6.5.1 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.

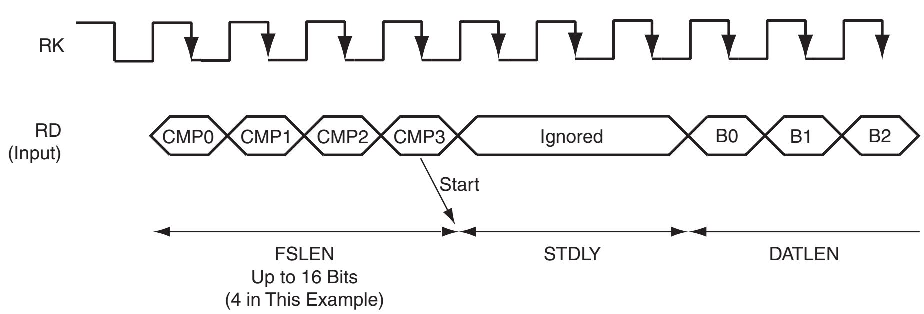
The Transmit Frame Sync Operation is performed by the transmitter only if the bit Frame Sync Data Enable (FSDEN) in SSC_TFMR is set. If the Frame Sync length is equal to or lower than the delay between the start event and the actual data transmission, the normal transmission has priority and the data contained in the Transmit Sync Holding Register is transferred in the Transmit Register, then shifted out.

37.6.5.2 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).

37.6.6 Receive Compare Modes

Figure 37-12. Receive Compare Modes



37.6.6.1 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.

37.6.7 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:

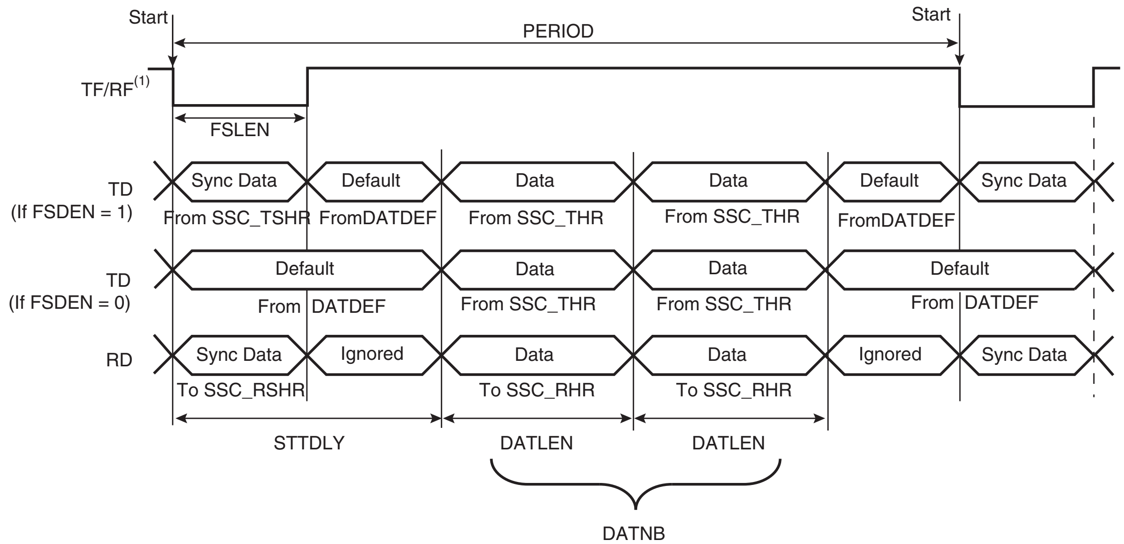
- the event that starts the data transfer (START)
- the delay in number of bit periods between the start event and the first data bit (STTDLY)
- the length of the data (DATLEN)
- the number of data to be transferred for each start event (DATNB).
- the length of synchronization transferred for each start 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 37-5. Data Frame Registers

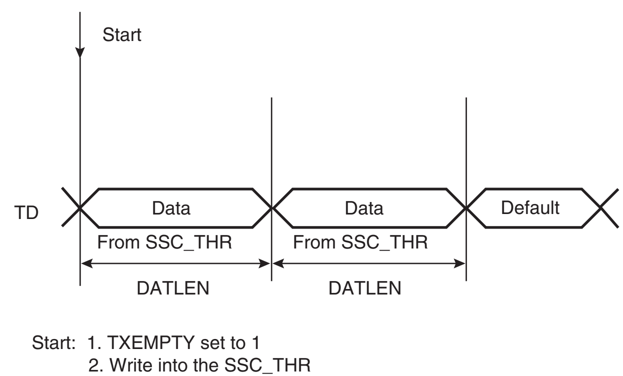
Transmitter	Receiver	Field	Length	Comment
SSC_TFMR	SSC_RFMR	DATLEN	Up to 32	Size of word
SSC_TFMR	SSC_RFMR	DATNB	Up to 16	Number of words transmitted in frame
SSC_TFMR	SSC_RFMR	MSBF		Most significant bit first
SSC_TFMR	SSC_RFMR	FSLEN	Up to 16	Size of Synchro data register
SSC_TFMR		DATDEF	0 or 1	Data 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 37-13. Transmit and Receive Frame Format in Edge/Pulse Start Modes



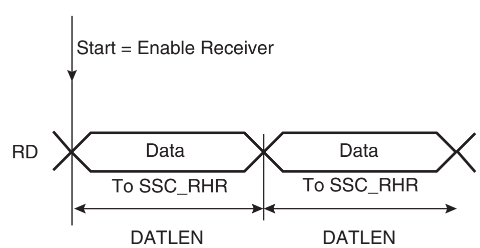
Note: 1. Example of input on falling edge of TF/RF.

Figure 37-14. Transmit Frame Format in Continuous Mode



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 37-15. Receive Frame Format in Continuous Mode



Note: 1. STTDLY is set to 0.

37.6.8 Loop Mode

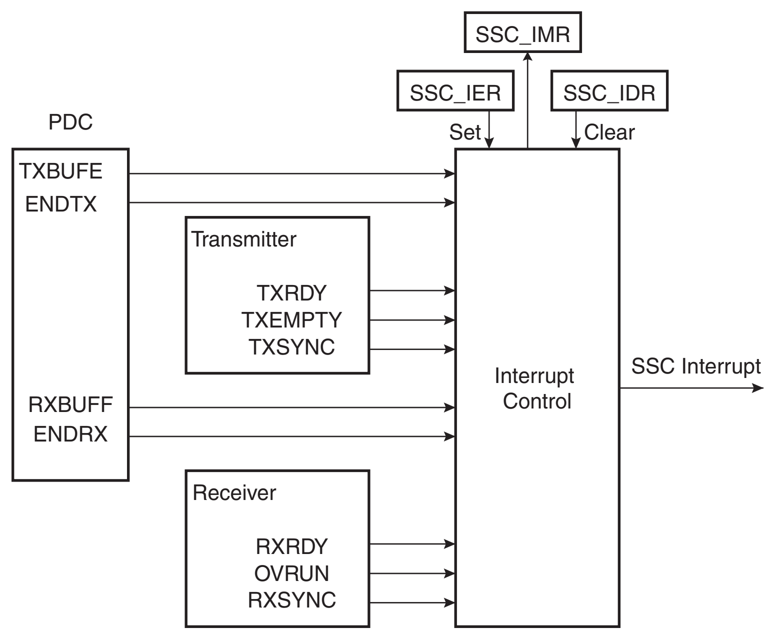
The receiver can be programmed to receive transmissions from the transmitter. This is done by setting the Loop Mode (LOOP) bit in SSC_RFMR. In this case, RD is connected to TD, RF is connected to TF and RK is connected to TK.

37.6.9 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 37-16. Interrupt Block Diagram



37.7 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 37-17. Audio Application Block Diagram

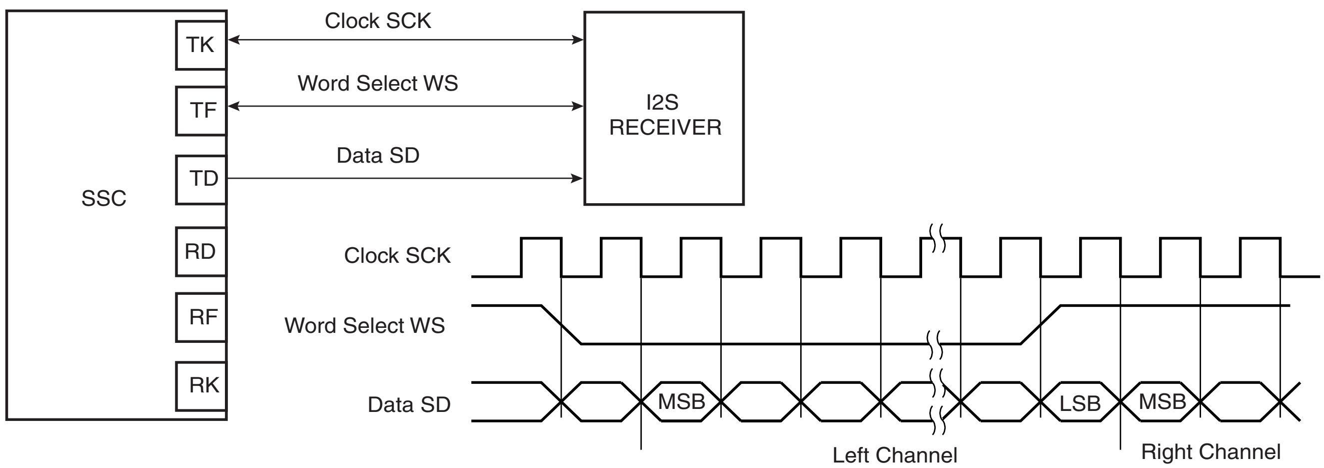


Figure 37-18. Codec Application Block Diagram

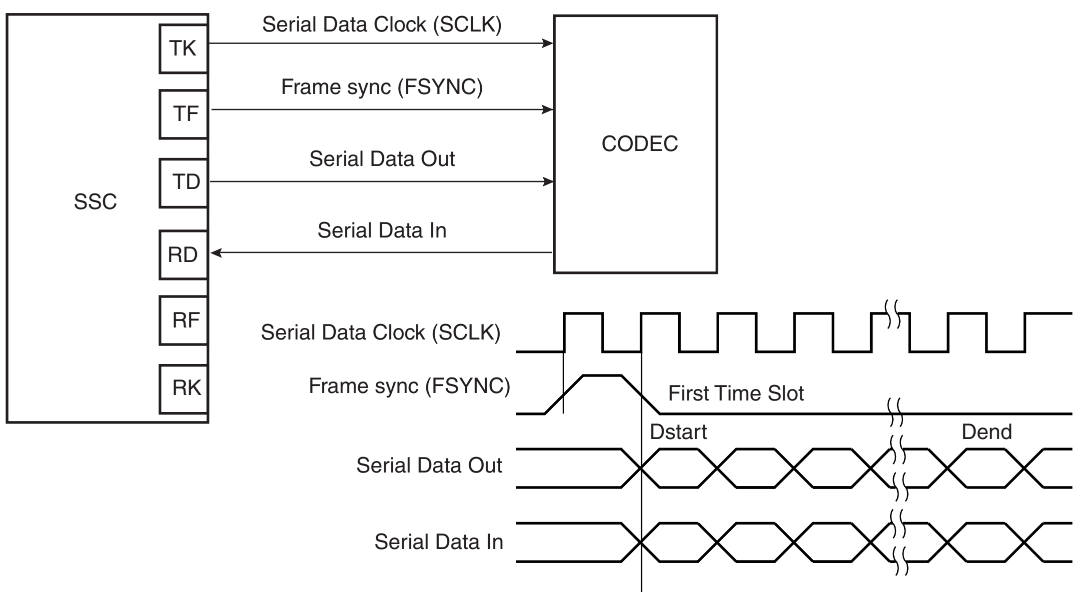
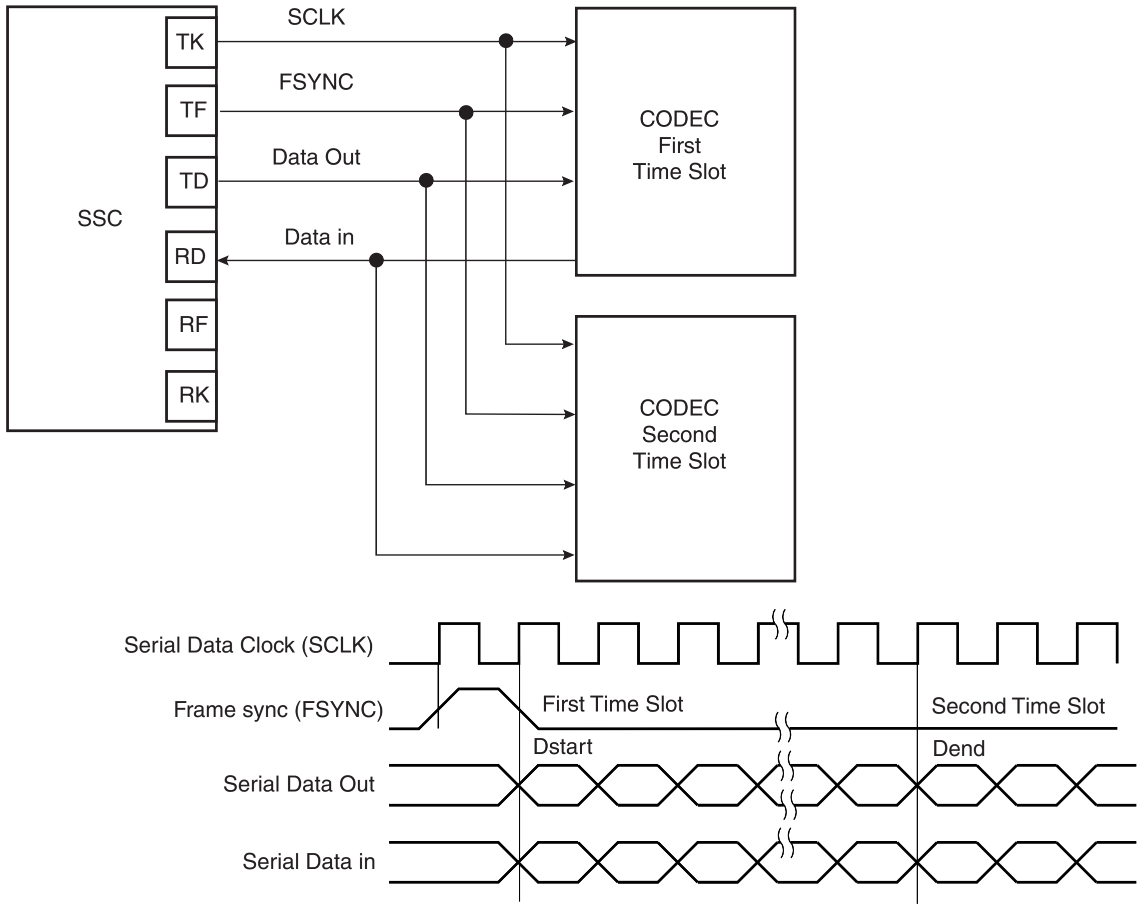


Figure 37-19. Time Slot Application Block Diagram



37.8 Synchronous Serial Controller (SSC) User Interface

Table 37-6. Register Mapping

Offset	Register	Name	Access	Reset
0x0	Control Register	SSC_CR	Write-only	–
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-only	0x0
0x24	Transmit Holding Register	SSC_THR	Write-only	–
0x28	Reserved	–	–	–
0x2C	Reserved	–	–	–
0x30	Receive Sync. Holding Register	SSC_RSHR	Read-only	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-only	0x000000CC
0x44	Interrupt Enable Register	SSC_IER	Write-only	–
0x48	Interrupt Disable Register	SSC_IDR	Write-only	–
0x4C	Interrupt Mask Register	SSC_IMR	Read-only	0x0
0x50-0xFC	Reserved	–	–	–
0x100- 0x124	Reserved for Peripheral Data Controller (PDC)	–	–	–

37.8.1 SSC Control Register

Name: SSC_CR

Addresses: 0xFFF98000 (0), 0xFFF9C000 (1)

Access: Write-only

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

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

15	14	13	12	11	10	9	8
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.

37.8.2 SSC Clock Mode Register

Name: SSC_CMR

Addresses: 0xFFF98004 (0), 0xFFF9C004 (1)

Access: 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 x 4095 = MCK/8190.

37.8.3 SSC Receive Clock Mode Register

Name: SSC_RCMR

Addresses: 0xFFF98010 (0), 0xFFF9C010 (1)

Access: 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
–	–	–	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.

37.8.4 SSC Receive Frame Mode Register

Name: SSC_RFMR

Addresses: 0xFFF98014 (0), 0xFFF9C014 (1)

Access: 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 647.

37.8.5 SSC Transmit Clock Mode Register

Name: SSC_TCMR

Addresses: 0xFFF98018 (0), 0xFFF9C018 (1)

Access: 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 x (PERIOD+1) Transmit Clock.

37.8.6 SSC Transmit Frame Mode Register

Name: SSC_TFMR

Addresses: 0xFFF9801C (0), 0xFFF9C01C (1)

Access: 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 period.

- **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 651.

37.8.7 SSC Receive Holding Register

Name: SSC_RHR

Addresses: 0xFFF98020 (0), 0xFFF9C020 (1)

Access: 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.

37.8.8 SSC Transmit Holding Register

Name: SSC_THR

Addresses: 0xFFF98024 (0), 0xFFF9C024 (1)

Access: 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.

37.8.9 SSC Receive Synchronization Holding Register

Name: SSC_RSHR

Addresses: 0xFFF98030 (0), 0xFFF9C030 (1)

Access: Read-only

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

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

15	14	13	12	11	10	9	8
RSDAT							

7	6	5	4	3	2	1	0
RSDAT							

- **RSDAT: Receive Synchronization Data**

37.8.10 SSC Transmit Synchronization Holding Register

Name: SSC_TSHR

Addresses: 0xFFF98034 (0), 0xFFF9C034 (1)

Access: 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**

37.8.11 SSC Receive Compare 0 Register

Name: SSC_RC0R

Addresses: 0xFFF98038 (0), 0xFFF9C038 (1)

Access: 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**

37.8.12 SSC Receive Compare 1 Register

Name: SSC_RC1R

Addresses: 0xFFF9803C (0), 0xFFF9C03C (1)

Access: 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**

37.8.13 SSC Status Register

Name: SSC_SR

Addresses: 0xFFF98040 (0), 0xFFF9C040 (1)

Access: 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.

37.8.14 SSC Interrupt Enable Register

Name: SSC_IER

Addresses: 0xFFF98044 (0), 0xFFF9C044 (1)

Access: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	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 = 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.

37.8.15 SSC Interrupt Disable Register

Name: SSC_IDR

Addresses: 0xFFF98048 (0), 0xFFF9C048 (1)

Access: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	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.

37.8.16 SSC Interrupt Mask Register

Name: SSC_IMR

Addresses: 0xFFF9804C (0), 0xFFF9C04C (1)

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0

7	6	5	4	3	2	1	0
RXBUFF	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.

38. AC97 Controller (AC97C)

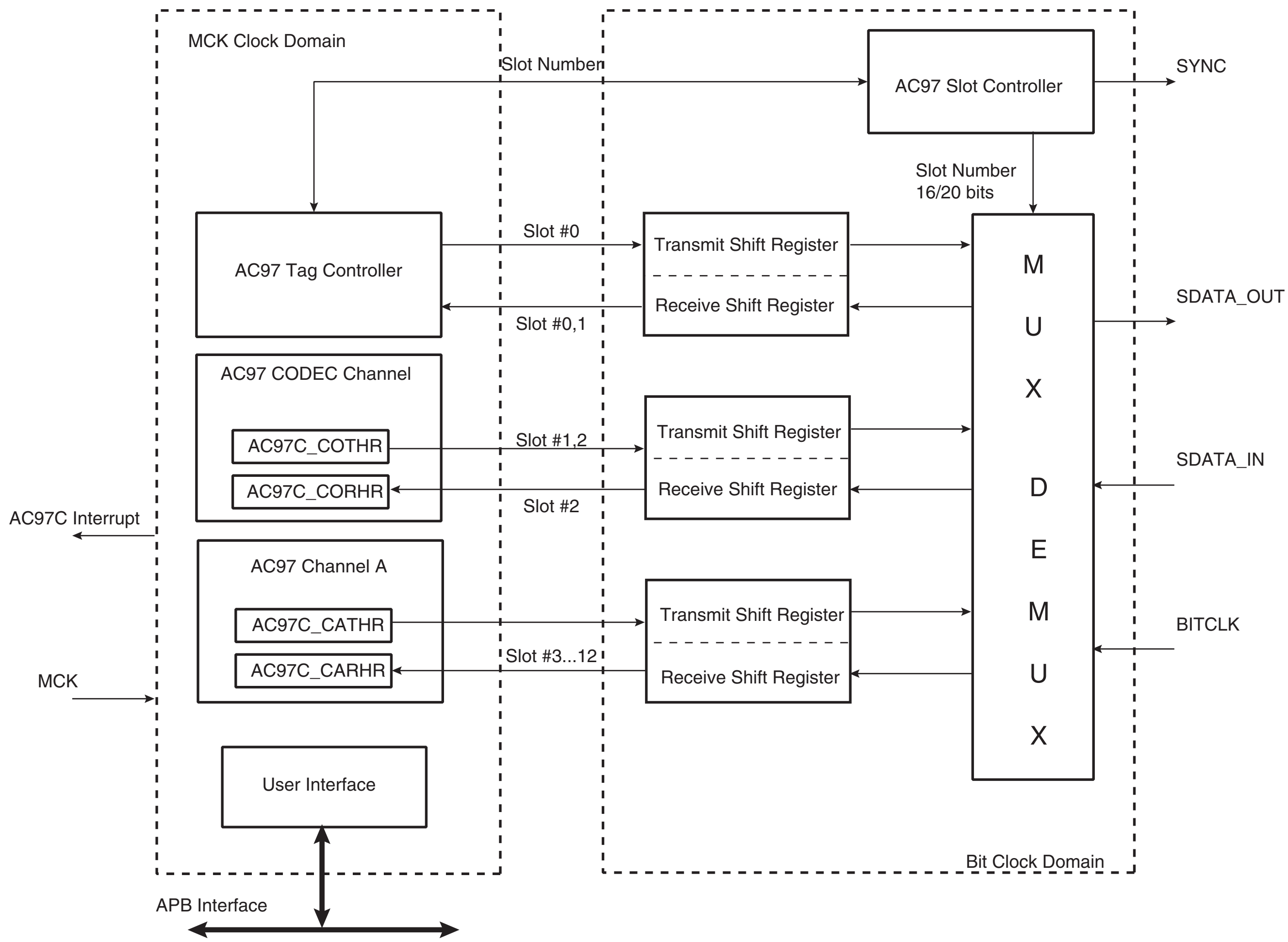
38.1 Overview

The AC97 Controller is the hardware implementation of the AC97 digital controller (DC'97) compliant with AC97 Component Specification 2.2. The AC97 Controller communicates with an audio codec (AC97) 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 AC97 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.

38.2 Block Diagram

Figure 38-1. Functional Block Diagram



38.3 Pin Name List

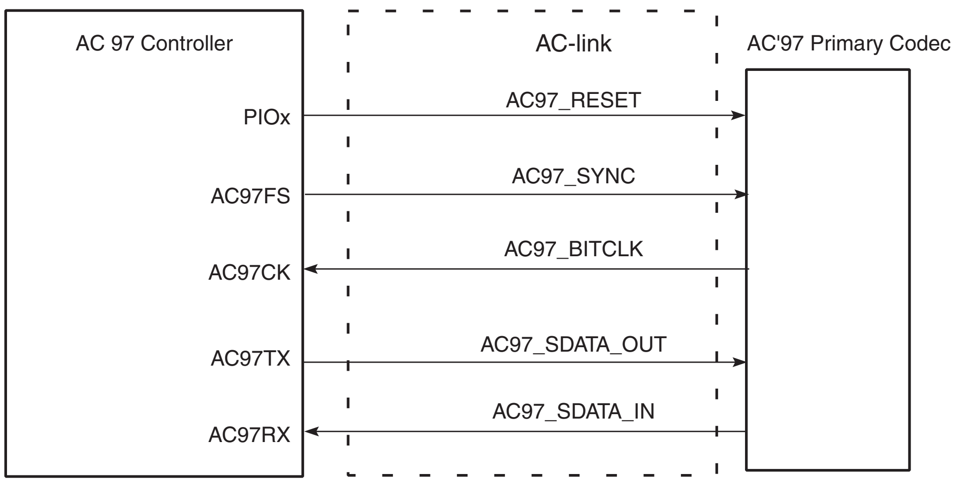
Table 38-1. I/O Lines Description

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 AC97 reset signal provided to the primary codec can be generated by a PIO.

38.4 Application Block Diagram

Figure 38-2. Application Block diagram



38.5 Product Dependencies

38.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines.

Before using the AC97 Controller receiver, the PIO controller must be configured in order for the AC97C receiver I/O lines to be in AC97 Controller peripheral mode.

Before using the AC97 Controller transmitter, the PIO controller must be configured in order for the AC97C transmitter I/O lines to be in AC97 Controller peripheral mode.

Table 38-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
AC97C	AC97CK	PA7	A
AC97C	AC97FS	PA6	A
AC97C	AC97RX	PA9	A
AC97C	AC97TX	PA8	A

38.5.2 Power Management

The AC97 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 AC97 Controller clock.

The AC97 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.

38.5.3 Interrupt

The AC97 Controller interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling interrupts requires programming the AIC before configuring the AC97C.

All AC97 Controller interrupts can be enabled/disabled by writing to the AC97 Controller Interrupt Enable/Disable Registers. Each pending and unmasked AC97 Controller interrupt will assert the interrupt line. The AC97 Controller interrupt service routine can get the interrupt source in two steps:

- Reading and ANDing AC97 Controller Interrupt Mask Register (AC97C_IMR) and AC97 Controller Status Register (AC97C_SR).
- Reading AC97 Controller Channel x Status Register (AC97C_CxSR).

Table 38-3. Peripheral IDs

Instance	ID
AC97C	19

38.6 Functional Description

38.6.1 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 38-3. Bidirectional AC-link Frame with Slot Assignment

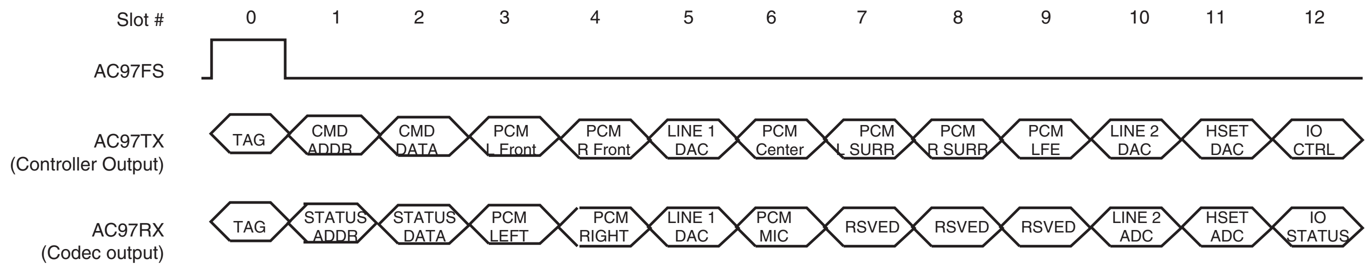


Table 38-4. AC-link Output Slots Transmitted from the AC97C Controller

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 38-5. AC-link Input Slots Transmitted from the AC97C 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
10	Modem Line 2 ADC
11	Modem Handset Input ADC
12	Modem IO Status

38.6.1.1 Slot Description

38.6.1.2 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 AC97 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 AC97 Controller according to the transmit request of each channel and to the SLOTREQ from the previous input frame, sent by the AC97 Codec, in Variable Sample Rate mode.

38.6.1.3 Codec Slot 1

The command/status slot is a 20-bit wide slot used to control features, and monitors status for AC97 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.

38.6.1.4 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.

38.6.1.5 Data Slots [3:12]

Slots [3:12] are 20-bit wide data slots, they usually carry audio PCM or/and modem I/O data.

38.6.2 AC97 Controller Channel Organization

The AC97 Controller features a Codec channel and 2 logical channels: Channel A.

The Codec channel controls AC97 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 transfer data to/from AC97 codec. All audio samples and modem data must transit by these 2 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 Channel A. The slot to channel assignment is configured by two registers:

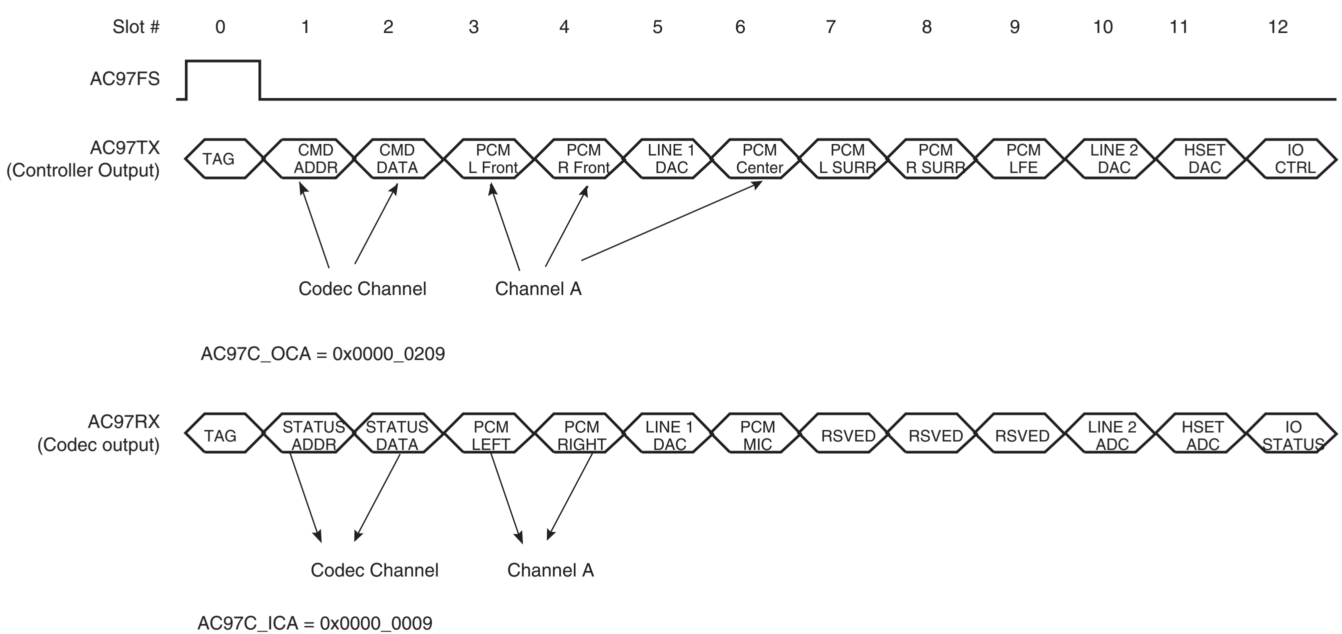
- AC97 Controller Input Channel Assignment Register (AC97C_ICA)
- AC97 Controller Output Channel Assignment Register (AC97C_OCA)

The AC97 Controller Input Channel Assignment Register (AC97C_ICA) configures the input slot to channel assignment. The AC97 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 AC97 Controller. Slots 0, 1,and 2 cannot be assigned to Channel A through the AC97C_OCA and AC97C_ICA Registers.

The width of sample data, that transit via the Channel varies and can take one of these values; 10, 16, 18 or 20 bits.

Figure 38-4. Logical Channel Assignment



38.6.2.1 AC97 Controller Setup

The following operations must be performed in order to bring the AC97 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 AC97 Controller Input Assignment Register (AC97C_ICA).
4. Configure the output channel assignment by controlling the AC97 Controller Input Assignment Register (AC97C_OCA).
5. Configure sample width for Channel A by writing the SIZE bit field in AC97C Channel x Mode Register (AC97C_CAMR). The application can write 10, 16, 18,or 20-bit wide PCM samples through the AC97 interface and they will be transferred into 20-bit wide slots.
6. Configure data Endianness for Channel A by writing CEM bit field in (AC97C_CAMR) register. Data on the AC-link are shifted MSB first. The application can write little- or big-endian data to the AC97 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 by writing CEN bit field in AC97C_CxMR register.

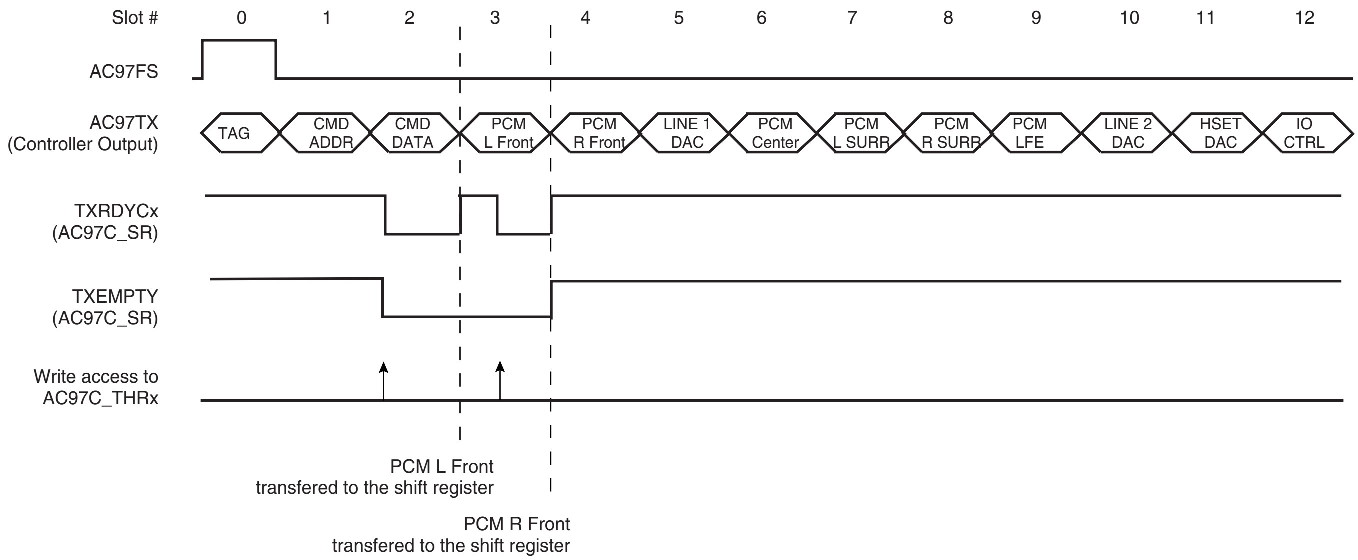
38.6.2.2 Transmit Operation

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 (AC97_CxSR). x being one of the 2 channels.
- Write data to the AC97 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 AC97 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 38-5. Audio Transfer (PCM L Front, PCM R Front) on Channel x



The TXEMPTY flag in the AC97 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 AC97 Controller is embedded in chips that target audio player devices. In such cases, the AC97 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.

38.6.2.3 AC97 Output Frame

The AC97 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 AC97 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.

38.6.2.4 Receive Operation

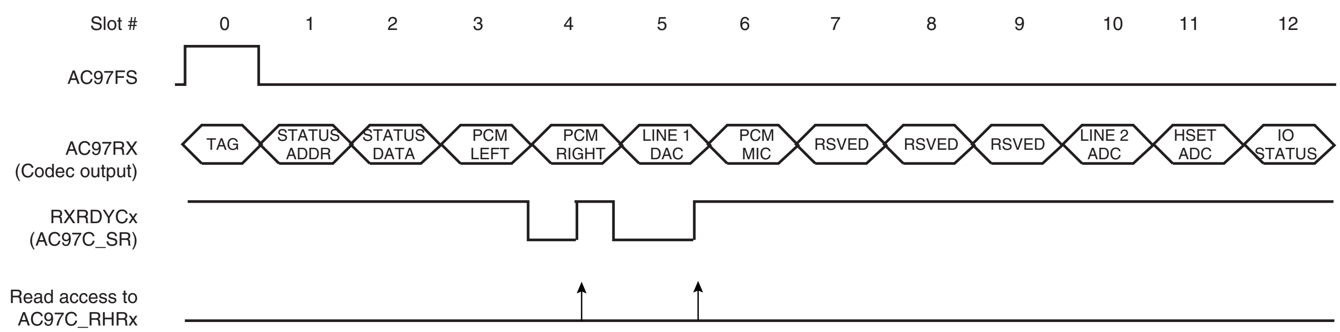
The AC97 Controller can also receive data from AC97 Codec. Data is received in the channel's shift register and then transferred to the AC97 Controller Channel x Read Holding Register. To read the newly received data, the application must perform the following steps:

- Poll RXRDY flag in AC97 Controller Channel x Status Register (AC97C_CxSR). x being one of the 2 channels.
- Read data from AC97 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 38-6. 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 AC97 Controller can also be used in sound recording devices in association with an AC97 Codec. The AC97 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.

38.6.2.5 AC97 Input Frame

The AC97 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 AC97 Controller Output Channel Assignment Register (AC97C_ICA) content. The AC97 Controller will not receive any data from any slot if AC97C_ICA is not assigned to a channel in input.

38.6.2.6 Configuring and Using Interrupts

Instead of polling flags in AC97 Controller Global Status Register (AC97C_SR) and in AC97 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 AC97 Controller Channel x Mode Register (AC97C_CxMR).
- Set the interruptible event and channel event in AC97 Controller Interrupt Enable Register (AC97C_IER).

The interrupt handler must read both AC97 Controller Global Status Register (AC97C_SR) and AC97 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 AC97 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 AC97 Controller Interrupt Disable Register (AC97C_IDR). The AC97 Controller Interrupt Mask Register (AC97C_IMR) shows which event can trigger an interrupt and which one cannot.

38.6.2.7 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.

38.6.2.8 To Transmit a Word Stored in Big Endian Format on AC-link

Word to be written in AC97 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]}.

38.6.2.9 To Transmit A Halfword Stored in Big Indian Format on AC-link

Halfword to be written in AC97 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 AC97 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]}.

38.6.2.10 To Transmit a10-bit Sample Stored in Big Endian Format on AC-link

Halfword to be written in AC97 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 AC97 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]}.

38.6.2.11 *To Receive Word transfers*

Data received on appropriate slot: data[19:0] = {Byte2[3:0], Byte1[7:0], Byte0[7:0]}.

Word stored in AC97 Controller Channel x Receive Holding Register (AC97C_CxRHR) (Received Data).

31 ... 24	23 ... 20	19 ... 16	15 ... 8	7 ... 0
–	–	Byte2[3:0]	Byte1[7:0]	Byte0[7:0]

Data is read from AC97 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).

31 ... 24	23 ... 16	15 ... 8	7 ... 0
Byte0[7:0]	Byte1[7:0]	{0x0, Byte2[3:0]}	0x00

38.6.2.12 *To Receive Halfword Transfers*

Data received on appropriate slot: data[19:0] = {0x0, Byte1[7:0], Byte0[7:0]}.

Halfword stored in AC97 Controller Channel x Receive Holding Register (AC97C_CxRHR) (Received Data).

31 ... 24	23 ... 16	15 ... 8	7 ... 0
–	–	Byte1[7:0]	Byte0[7:0]

Data is read from AC97 Controller Channel x Receive Holding Register (AC97C_CxRHR) when data size is equal to 16 bits and when big-endian mode is enabled.

31 ... 24	23 ... 16	15 ... 8	7 ... 0
–	–	Byte0[7:0]	Byte1[7:0]

38.6.2.13 *To Receive 10-bit Samples*

Data received on appropriate slot: data[19:0] = {0x000, Byte1[1:0], Byte0[7:0]}.Halfword stored in AC97 Controller Channel x Receive Holding Register (AC97C_CxRHR) (Received Data)

31 ... 24	23 ... 16	15 ... 10	9 ... 8	7 ... 0
–	–	–	Byte1 [1:0]	Byte0[7:0]

Data read from AC97 Controller Channel x Receive Holding Register (AC97C_CxRHR) when data size is equal to 10 bits and when big-endian mode is enabled.

31 ... 24	23 ... 16	15 ... 8	7 ... 3	1 ... 0
–	–	Byte0[7:0]	0x00	Byte1 [1:0]

38.6.3 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 AC97 standard approach calls for the addition of “on-demand” slot request flags. The AC97 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 AC97 Controller in slot 1/SLOTREQ in every audio input frame. Each time the

AC97 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 AC97 Controller Mode Register (AC97C_MR).
- Enable Variable Rate mode in the AC97 Codec by performing a transfer on the Codec channel.

Slot 1 of the input frame is automatically interpreted as SLOTREQ signaling bits. The AC97 Controller will automatically fill the active slots according to both SLOTREQ and AC97C_OCA register in the next transmitted frame.

38.6.4 Power Management

38.6.4.1 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 AC97 Controller should not attempt to play or capture audio data until it has awakened AC97 Codec.

To set the AC97 Codec in low power mode, the PR4 bit in the AC97 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 field in the AC97C_CAMR 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.

38.6.4.2 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.

38.6.4.3 Wake-up Triggered by the AC97 Controller

The AC97 Controller can wake up the AC97 Codec by issuing either a cold or a warm reset.

The AC97 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.

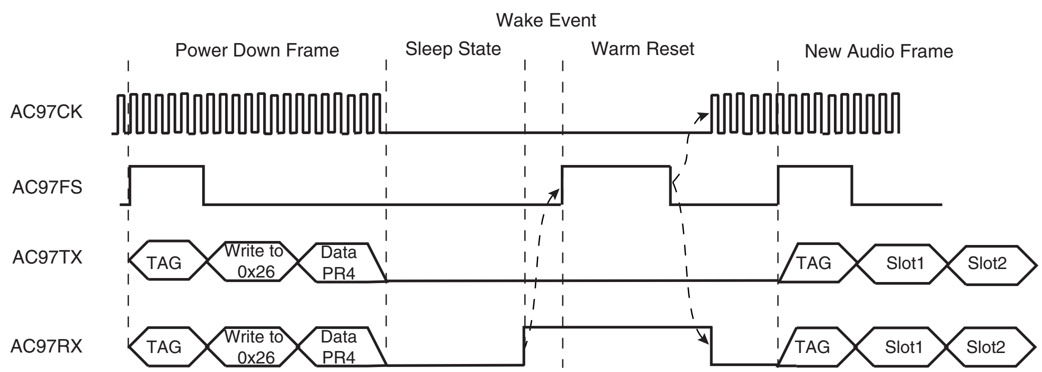
38.6.4.4 Wake-up Triggered by the AC97 Codec

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 worm reset. The AC97RX rising edge is asynchronously (regarding AC97FS) detected by the AC97 Controller. If WKUP bit is enabled in AC97C_IMR register, an interrupt is triggered that wakes up the AC97 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 38-7. AC97 Power-Down/Up Sequence



38.6.4.5 AC97 Codec Reset

There are three ways to reset an AC97 Codec.

38.6.4.6 Cold AC97 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 AC97 Controller
- Clear PIO line output controlling the AC97 RESET signal
- Wait for the minimum specified time
- Set PIO line output controlling the AC97 RESET signal

AC97CK, the clock provided by AC97 Codec, is detected by the controller.

38.6.4.7 Warm AC97 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 1us
- 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.

38.7 AC97 Controller (AC97C) User Interface

Table 38-6. Register Mapping

Offset	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
0x40	Codec Channel Receive Holding Register	AC97C_CORHR	Read	0x0
0x44	Codec Channel Transmit Holding Register	AC97C_COTHR	Write	–
0x48	Codec Status Register	AC97C_COSR	Read	0x0
0x4C	Codec Mode Register	AC97C_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 DMA Controller (PDC) registers related to channel transfers	–	–	–

38.7.1 AC97 Controller Mode Register

Name: AC97C_MR

Address: 0xFFFA0008

Access: 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: AC97 Controller Global Enable**

0: No effect. AC97 function as well as access to other AC97 Controller registers are disabled.

1: Activates the AC97 function.

38.7.2 AC97 Controller Input Channel Assignment Register

Name: AC97C_ICA

Address: 0xFFFA0010

Access: 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 time
0x1	Channel A data will be received during this slot time.

38.7.3 AC97 Controller Output Channel Assignment Register

Name: AC97C_OCA

Address: 0xFFFA0014

Access: 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 time
0x1	Channel A data will be transferred during this slot time.

38.7.4 AC97 Controller Codec Channel Receive Holding Register

Name: AC97C_CORHR

Address: 0xFFFA0040

Access: Read-only

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

- **SDATA: Status Data**

Data sent by the CODEC in the third AC97 input frame slot (Slot 2).

38.7.5 AC97 Controller Codec Channel Transmit Holding Register

Name: AC97C_COTHR

Address: 0xFFFA0044

Access: 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 AC97 frame slot

- **CADDR: CODEC control register index**

Data sent to the CODEC in the second AC97 frame slot.

- **CDATA: Command Data**

Data sent to the CODEC in the third AC97 frame slot (Slot 2).

38.7.6 AC97 Controller Channel A, Receive Holding Register

Name: AC97C_CARHR

Address: 0xFFFA0020

Access: 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.

38.7.7 AC97 Controller Channel A, Transmit Holding Register

Name: AC97C_CATHR

Address: 0xFFFA0024

Access: 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.

38.7.8 AC97 Controller Channel A Status Register

Name: AC97C_CASR

Address: 0xFFFA0028

Access: Read-only

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

- **ENDTX: End of Transmission for Channel A**

0: The register AC97C_CATCR has not reached 0 since the last write in AC97C_CATCR or AC97C_CANCR.

1: The register AC97C_CATCR has reached 0 since the last write in AC97C_CATCR or AC97C_CATNCR.

- **TXBUFE: Transmit Buffer Empty for Channel A**

0: AC97C_CATCR or AC97C_CATNCR have a value other than 0.

1: Both AC97C_CATCR and AC97C_CATNCR have a value of 0.

- **ENDRX: End of Reception for Channel A**

0: The register AC97C_CARCR has not reached 0 since the last write in AC97C_CARCR or AC97C_CARNCR.

1: The register AC97C_CARCR has reached 0 since the last write in AC97C_CARCR or AC97C_CARNCR.

- **RXBUFF: Receive Buffer Full for Channel A**

0: AC97C_CARCR or AC97C_CARNCR have a value other than 0.

1: Both AC97C_CARCR and AC97C_CARNCR have a value of 0.

38.7.9 AC97 Controller Codec Status Register

Name: AC97C_COSR

Address: 0xFFFA0048

Access: Read-only

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

38.7.10 AC97 Controller Channel A Mode Register

Name: AC97C_CAMR

Address: 0xFFFA002C

Access: 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

- **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**

- **ENDTX: End of Transmission for Channel A Interrupt Enable**

- **TXBUFE: Transmit Buffer Empty for Channel A 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.

- **ENDRX: End of Reception for Channel A 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.

- **RXBUFF: Receive Buffer Full for Channel A 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.

- **SIZE: Channel A Data Size**

SIZE Encoding

SIZE	Selected Data Size
0x0	20 bits
0x1	18 bits
0x2	16 bits
0x3	10 bits

Note: Each time slot in the data phase is 20 bit long. For example, if a 16-bit sample stream is being played to an AC 97 DAC, the first 16 bit positions are presented to the DAC MSB-justified. They are followed by the next four bit positions that the AC97 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)

- **CEM: Channel A Endian Mode**

0: Transferring Data through Channel A is straight forward (Little-Endian).

1: Transferring Data through Channel A from/to a memory is performed with from/to Big-Endian format translation.

- **CEN: Channel A Enable**

0: Data transfer is disabled on Channel A.

1: Data transfer is enabled on Channel A.

- **PDCEN: Peripheral Data Controller Channel Enable**

0: Channel A is not transferred through a Peripheral Data Controller Channel. Related PDC flags are ignored or not generated.

1: Channel A is transferred through a Peripheral Data Controller Channel. Related PDC flags are taken into account or generated.

38.7.11 AC97 Controller Codec Mode Register

Name: AC97C_COMR

Address: 0xFFFA004C

Access: 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**

38.7.12 AC97 Controller Status Register

Name: AC97C_SR

Address: 0xFFFA0050

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	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 AC97 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.

38.7.13 AC97 Codec Controller Interrupt Enable Register

Name: AC97C_IER

Address: 0xFFFA0054

Access: Write-only

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

- **SOF: Start Of Frame**

- **WKUP: Wake Up**

- **COEVT: Codec Event**

- **CAEVT: Channel A Event**

0: No Effect.

1: Enables the corresponding interrupt.

38.7.14 AC97 Controller Interrupt Disable Register

Name: AC97C_IDR

Address: 0xFFFA0058

Access: Write-only

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

- **SOF: Start Of Frame**

- **WKUP: Wake Up**

- **COEVT: Codec Event**

- **CAEVT: Channel A Event**

0: No Effect.

1: Disables the corresponding interrupt.

38.7.15 AC97 Controller Interrupt Mask Register

Name: AC97C_IMR

Address: 0xFFFA005C

Access: Read-only

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

- **SOF: Start Of Frame**

- **WKUP: Wake Up**

- **COEVT: Codec Event**

- **CAEVT: Channel A Event**

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

39. Timer Counter (TC)

39.1 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 39-1 gives the assignment of the device Timer Counter clock inputs common to Timer Counter 0 to 2.

Table 39-1. Timer Counter Clock Assignment

Name	Definition
TIMER_CLOCK1	MCK/2
TIMER_CLOCK2	MCK/8
TIMER_CLOCK3	MCK/32
TIMER_CLOCK4	MCK/128
TIMER_CLOCK5(1)	SLCK

Note: 1. When Slow Clock is selected for Master Clock (CSS = 0 in PMC Master CLock Register), TIMER_CLOCK5 input is Master Clock, i.e., Slow CLock modified by PRES and MDIV fields.

39.2 Block Diagram

Figure 39-1. Timer Counter Block Diagram

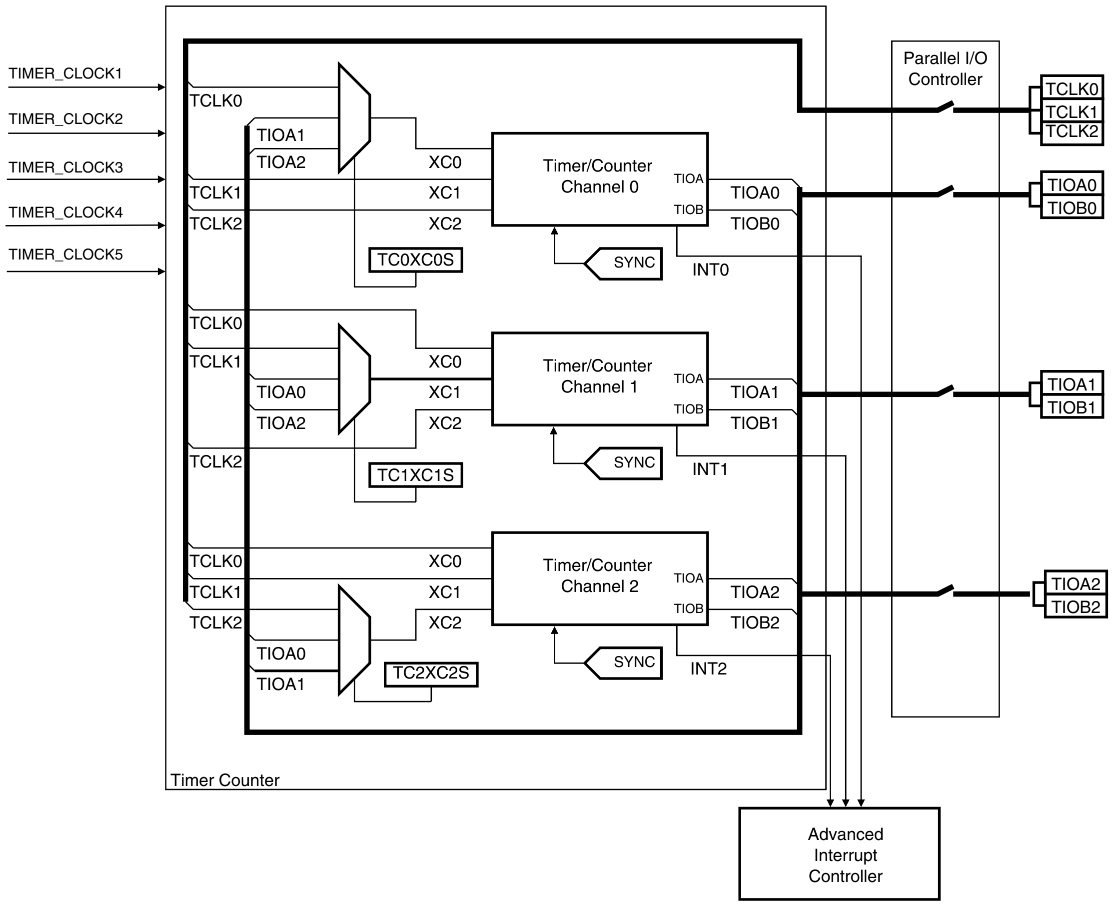


Table 39-2. Signal Name Description

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

39.3 Pin Name List

Table 39-3. 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

39.4 Product Dependencies

39.4.1 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.

Table 39-4. I/O Lines

Instance	Signal	I/O Line	Peripheral
TC0	TCLK0	PB28	B
TC0	TCLK1	PC28	B
TC0	TCLK2	PA14	A
TC0	TIOA0	PA29	A
TC0	TIOA1	PB6	B
TC0	TIOA2	PB21	B
TC0	TIOB0	PA30	A
TC0	TIOB1	PB7	B
TC0	TIOB2	PB22	B

39.4.2 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.

39.4.3 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.

39.5 Functional Description

39.5.1 TC Description

The three channels of the Timer Counter are independent and identical in operation. The registers for channel programming are listed in Table 39-5 on page 713.

39.5.2 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.

39.5.3 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 39-2 on page 701.

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 39-3 on page 701

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 39-2. Clock Chaining Selection

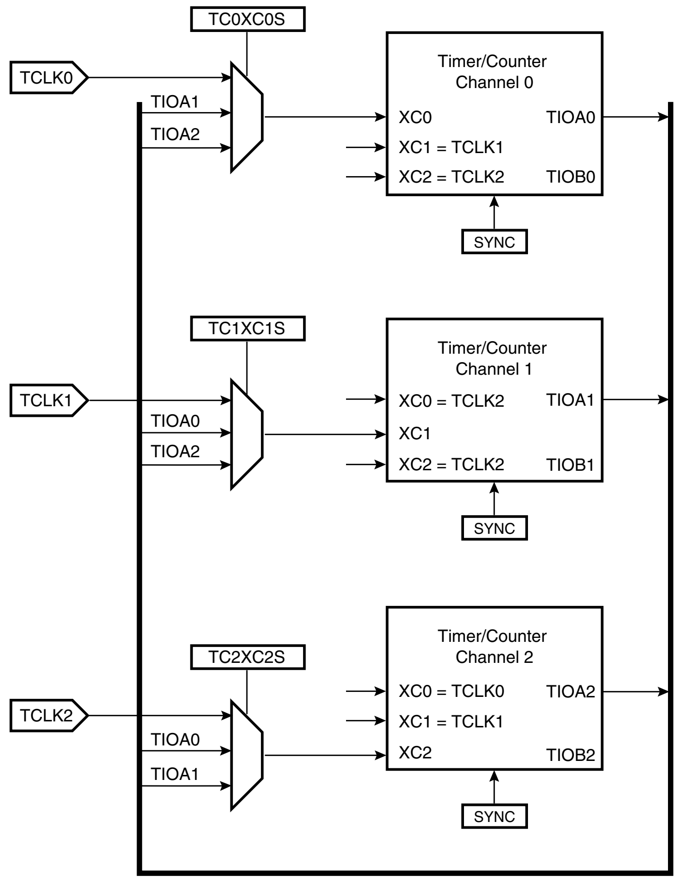
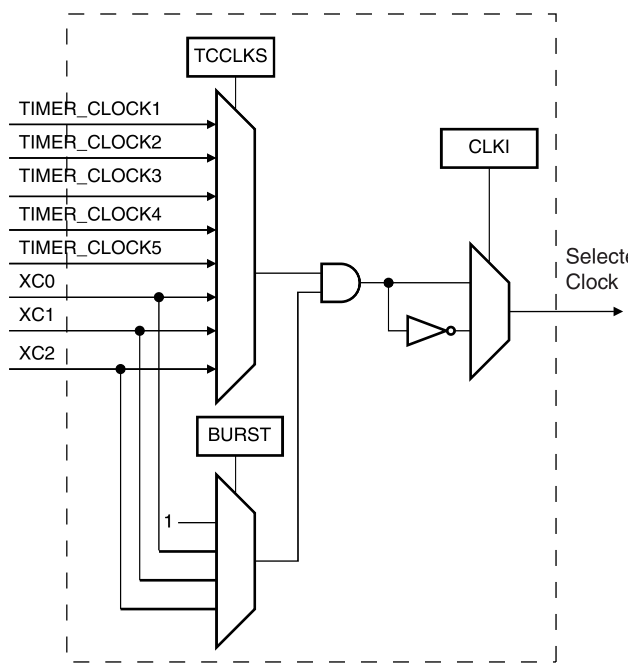


Figure 39-3. Clock Selection

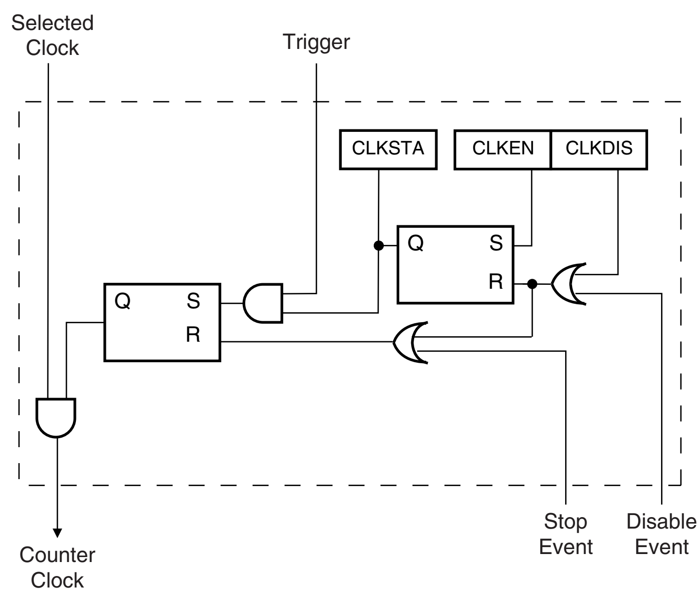


39.5.4 Clock Control

The clock of each counter can be controlled in two different ways: it can be enabled/disabled and started/stopped. See Figure 39-4.

- 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 39-4. Clock Control



39.5.5 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.

39.5.6 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.

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.

The following triggers are common to both modes:

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

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

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

39.5.7 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 39-5 shows the configuration of the TC channel when programmed in Capture Mode.

39.5.8 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.

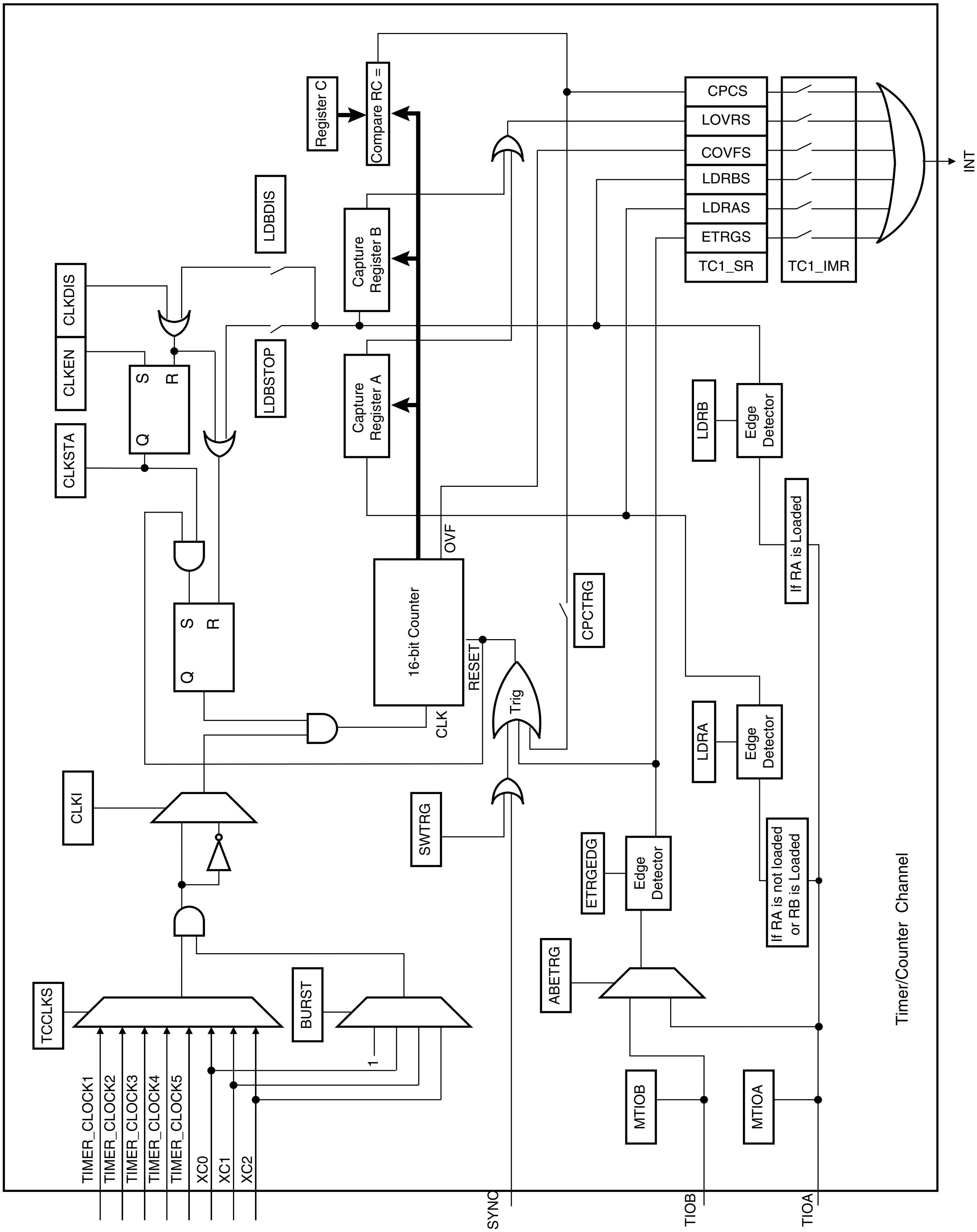
Loading RA or RB before the read of the last value loaded sets the Overrun Error Flag (LOVRS) in TC_SR (Status Register). In this case, the old value is overwritten.

39.5.9 Trigger Conditions

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

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

Figure 39-5. Capture Mode



39.5.10 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 39-6 shows the configuration of the TC channel when programmed in Waveform Operating Mode.

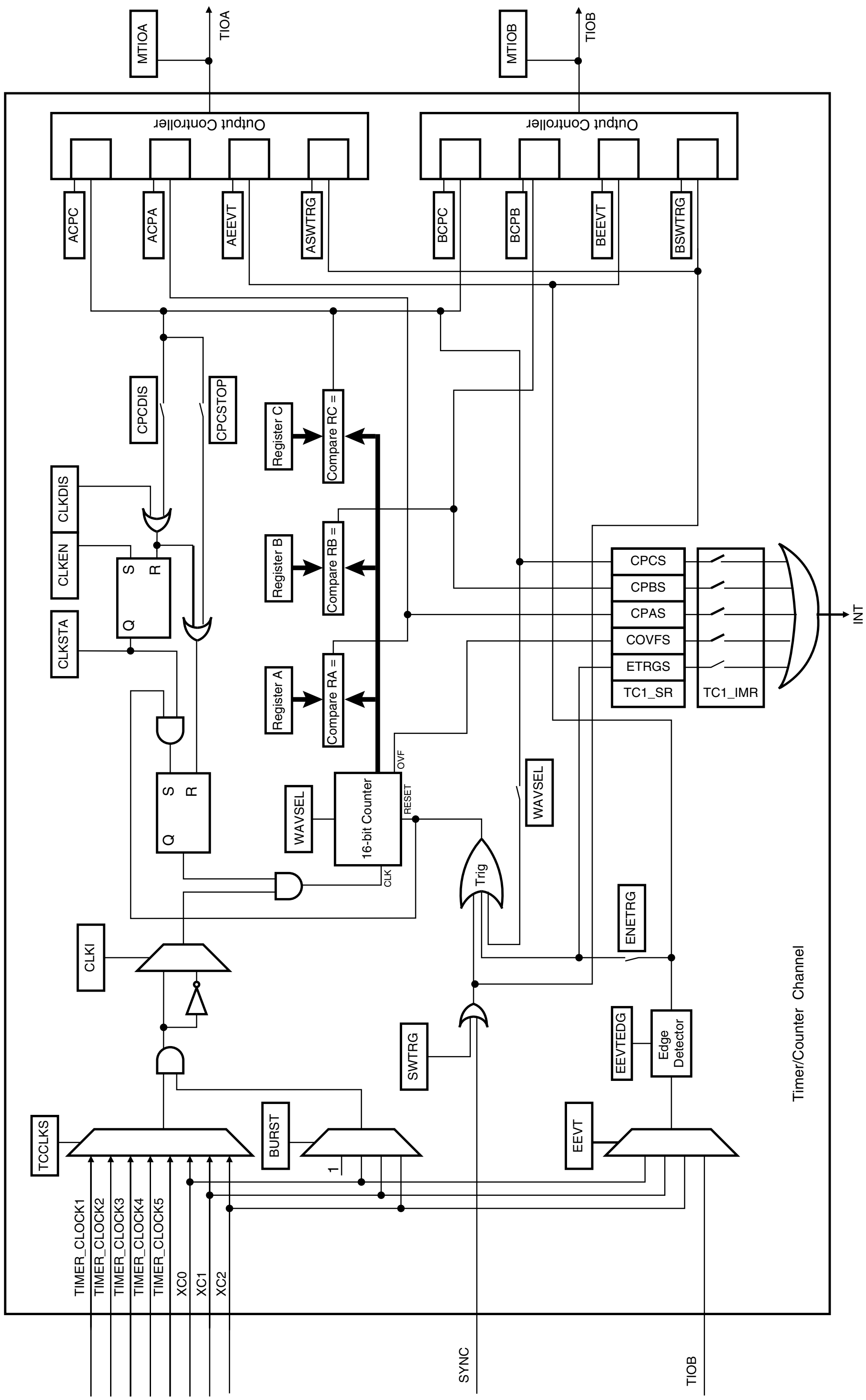
39.5.11 Waveform Selection

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 39-6. Waveform Mode



39.5.11.1 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 39-7.

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 39-8.

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 39-7. WAVSEL= 00 without trigger

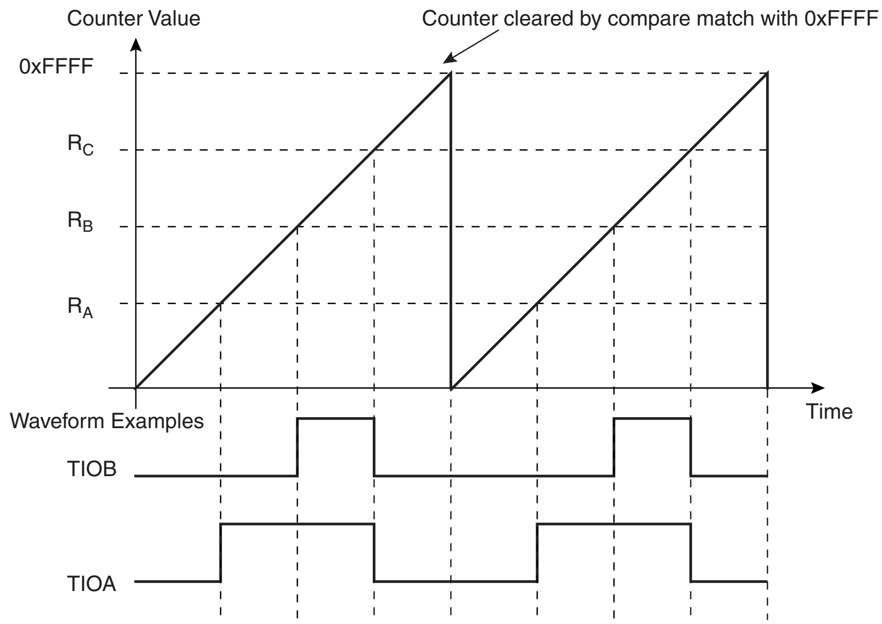
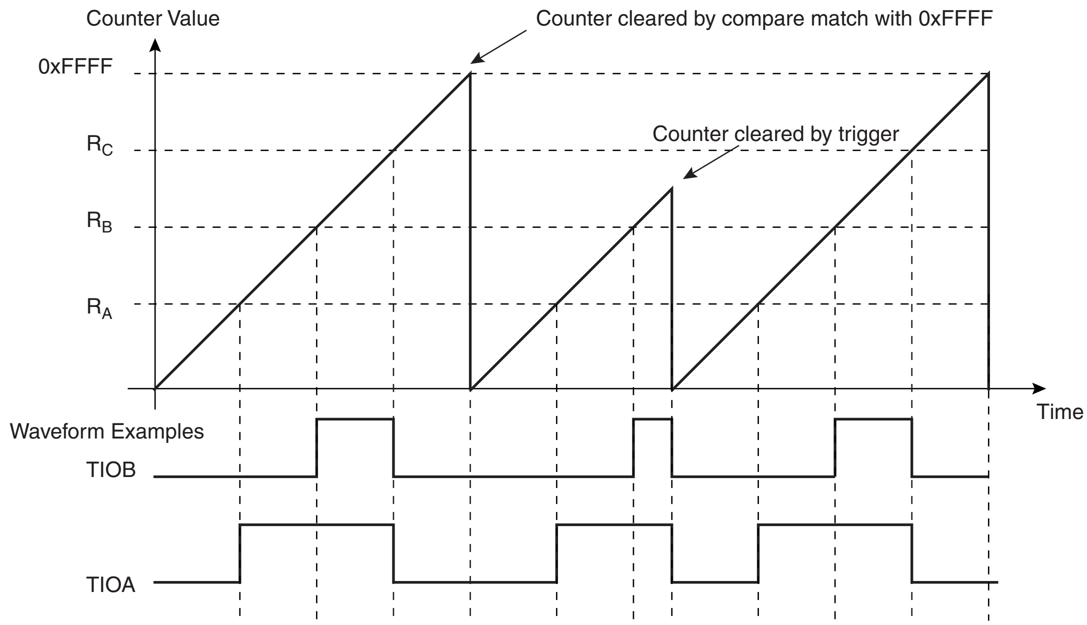


Figure 39-8. WAVSEL= 00 with trigger



39.5.11.2 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 39-9.

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 39-10.

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 39-9. WAVSEL = 10 Without Trigger

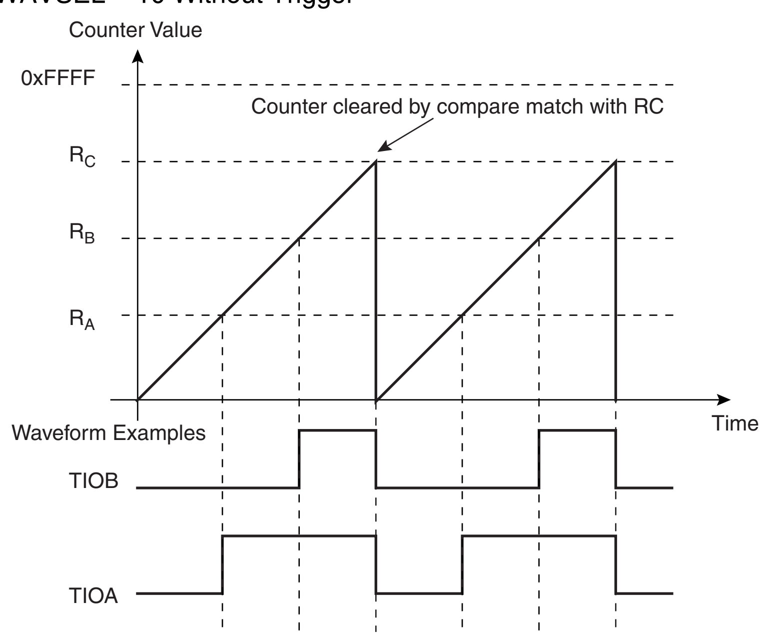
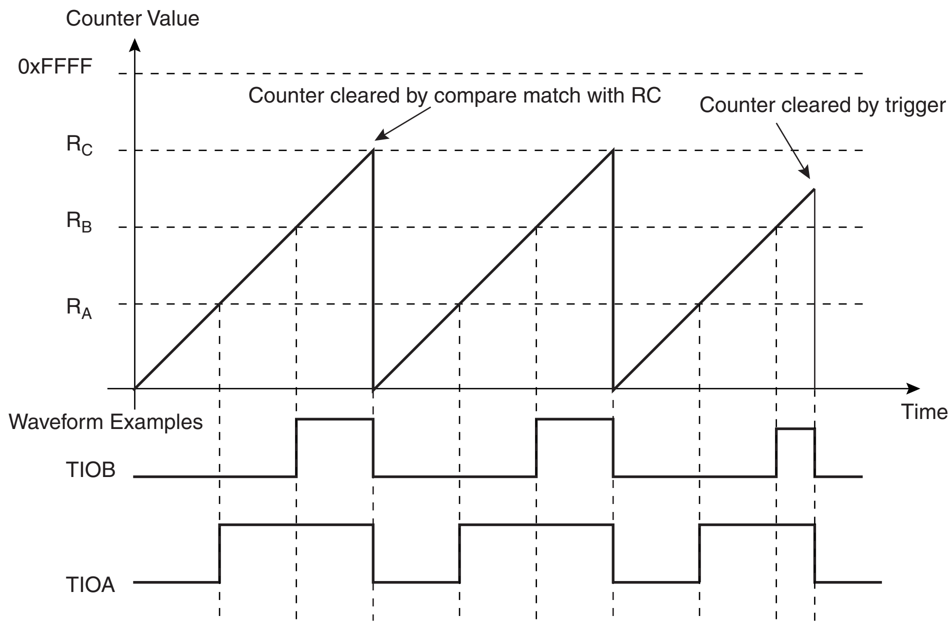


Figure 39-10. WAVSEL = 10 With Trigger



39.5.11.3 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 39-11.

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 39-12.

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 39-11. WAVSEL = 01 Without Trigger

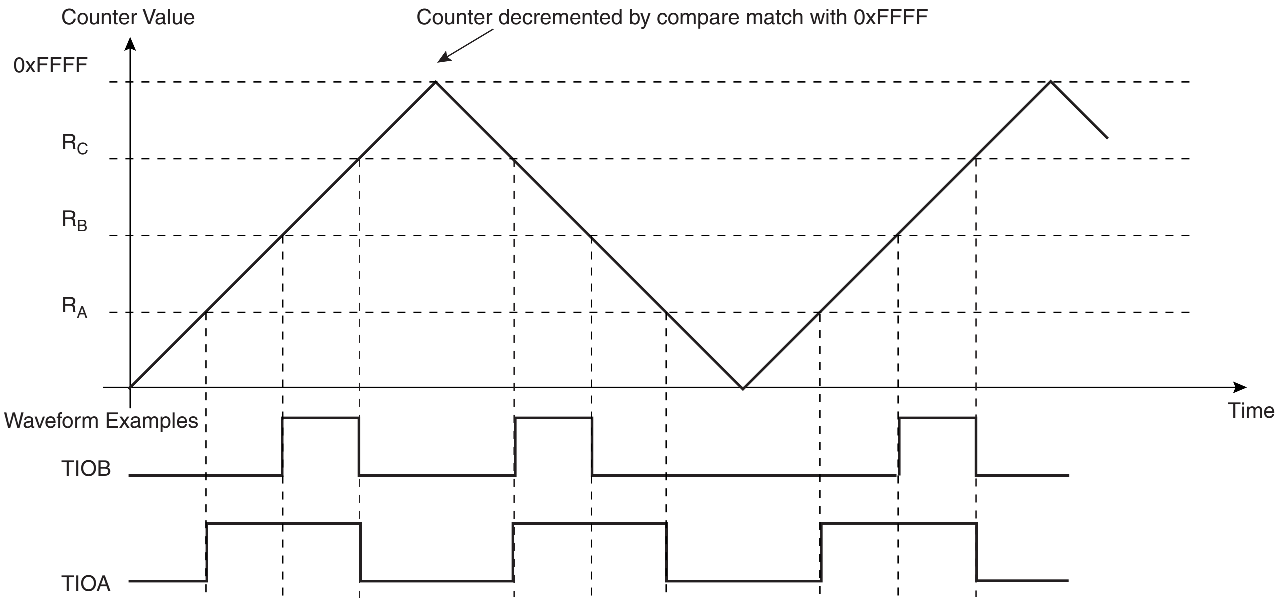
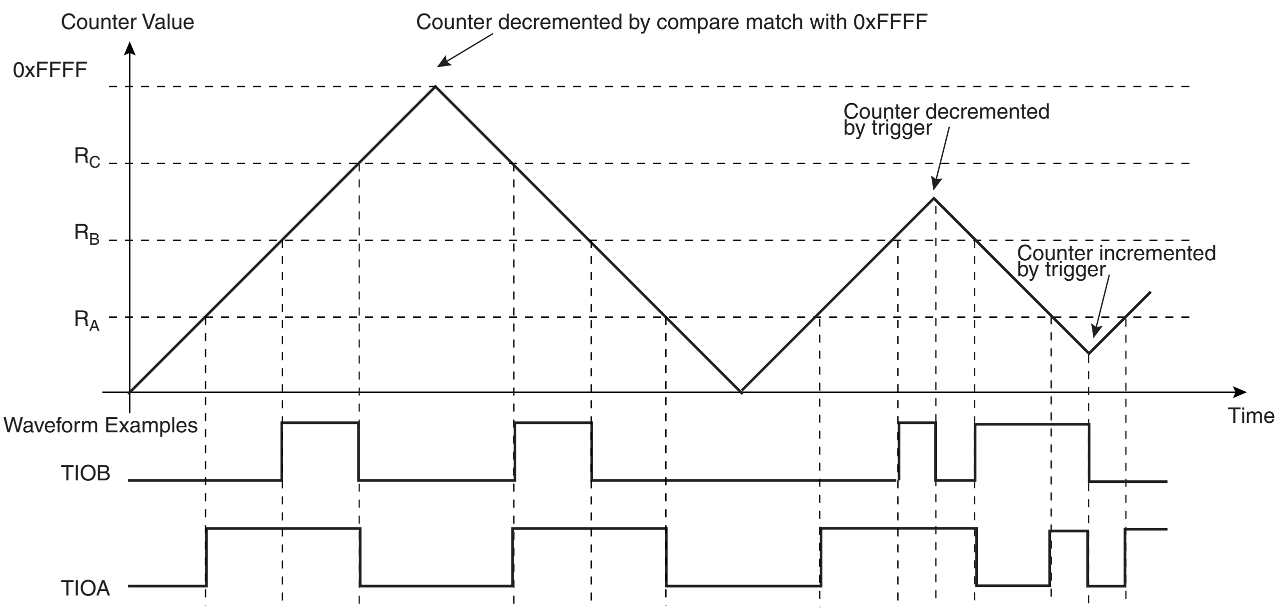


Figure 39-12. WAVSEL = 01 With Trigger



39.5.11.4 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 39-13.

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 39-14.

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

Figure 39-13. WAVSEL = 11 Without Trigger

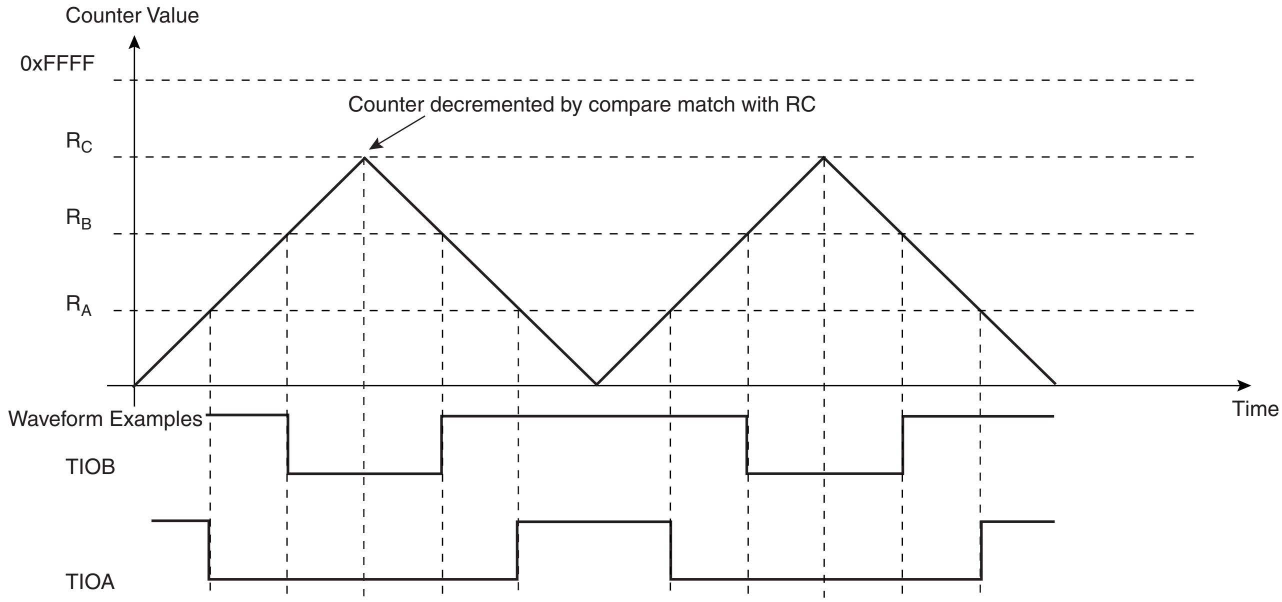
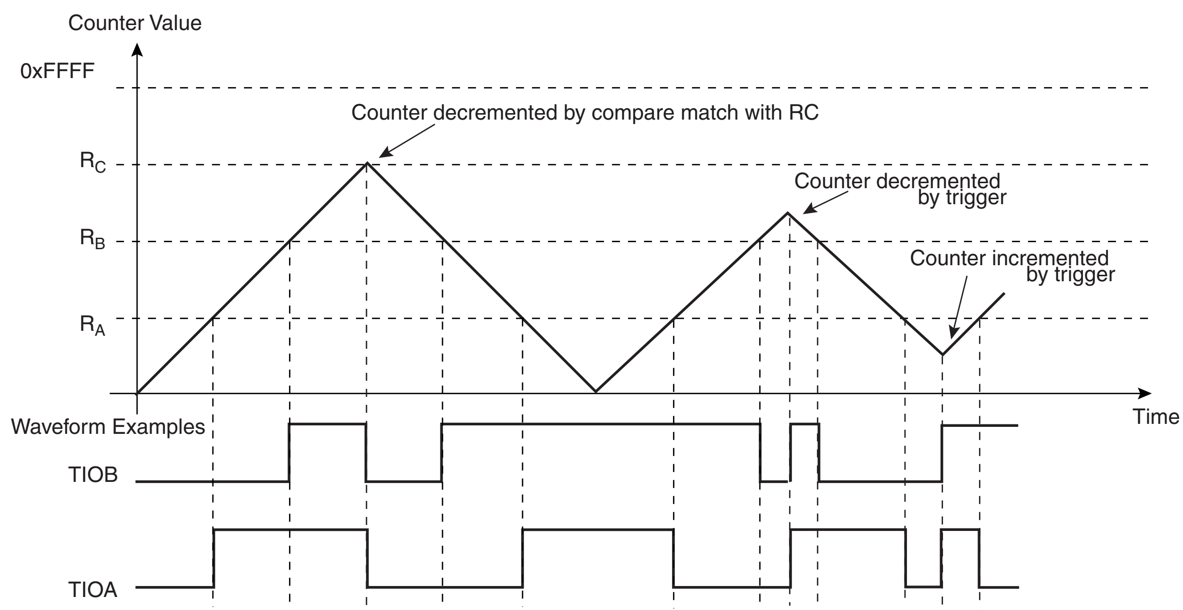


Figure 39-14. WAVSEL = 11 With Trigger



39.5.12 External Event/Trigger Conditions

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

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

If TIOB is defined as an external event signal (EEVT = 0), TIOB is no longer used as an output and the compare register B is not used to generate waveforms and subsequently no IRQs. In this case the TC channel can only generate a waveform on TIOA.

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

As in Capture Mode, the SYNC signal and the software trigger are also available as triggers. RC Compare can also be used as a trigger depending on the parameter WAVSEL.

39.5.13 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.

39.6 Timer Counter (TC) User Interface

Table 39-5. Register Mapping

Offset[1]	Register	Name	Access	Reset
0x00 + channel * 0x40 + 0x00	Channel Control Register	TC_CCR	Write-only	–
0x00 + channel * 0x40 + 0x04	Channel Mode Register	TC_CMR	Read-write	0
0x00 + channel * 0x40 + 0x08	Reserved			
0x00 + channel * 0x40 + 0x0C	Reserved			
0x00 + channel * 0x40 + 0x10	Counter Value	TC_CV	Read-only	0
0x00 + channel * 0x40 + 0x14	Register A	TC_RA	Read-write[2]	0
0x00 + channel * 0x40 + 0x18	Register B	TC_RB	Read-write[2]	0
0x00 + channel * 0x40 + 0x1C	Register C	TC_RC	Read-write	0
0x00 + channel * 0x40 + 0x20	Status Register	TC_SR	Read-only	0
0x00 + channel * 0x40 + 0x24	Interrupt Enable Register	TC_IER	Write-only	–
0x00 + channel * 0x40 + 0x28	Interrupt Disable Register	TC_IDR	Write-only	–
0x00 + channel * 0x40 + 0x2C	Interrupt Mask Register	TC_IMR	Read-only	0
0xC0	Block Control Register	TC_BCR	Write-only	–
0xC4	Block Mode Register	TC_BMR	Read-write	0
0xFC	Reserved	–	–	–

Notes: 1. Channel index ranges from 0 to 2.

2. Read-only if WAVE = 0

39.6.1 TC Block Control Register

Name: TC_BCR

Address: 0xFFF7C0C0

Access: Write-only

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

39.6.2 TC Block Mode Register

Name: TC_BMR

Address: 0xFFF7C0C4

Access: Read-write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	TC2XC2S		TC1XC1S		TC0XC0S	

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

TC0XC0S		Signal Connected to XC0
0	0	TCLK0
0	1	none
1	0	TIOA1
1	1	TIOA2

- **TC1XC1S: External Clock Signal 1 Selection**

TC1XC1S		Signal Connected to XC1
0	0	TCLK1
0	1	none
1	0	TIOA0
1	1	TIOA2

- **TC2XC2S: External Clock Signal 2 Selection**

TC2XC2S		Signal Connected to XC2
0	0	TCLK2
0	1	none
1	0	TIOA0
1	1	TIOA1

39.6.3 TC Channel Control Register

Name: TC_CCRx [x=0..2]

Addresses: 0xFFF7C000 (0)[0], 0xFFF7C040 (0)[1], 0xFFF7C080 (0)[2]

Access: Write-only

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

39.6.4 TC Channel Mode Register: Capture Mode

Name: TC_CMRx [x=0..2] (WAVE = 0)

Addresses: 0xFFF7C004 (0)[0], 0xFFF7C044 (0)[1], 0xFFF7C084 (0)[2]

Access: 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	CPCTRG	–	–	–	ABETRG	ETRGEDG	

7	6	5	4	3	2	1	0
LDBDIS	LDBSTOP	BURST		CLKI	TCCLKS		

- **TCCLKS: Clock Selection**

TCCLKS			Clock Selected
0	0	0	TIMER_CLOCK1
0	0	1	TIMER_CLOCK2
0	1	0	TIMER_CLOCK3
0	1	1	TIMER_CLOCK4
1	0	0	TIMER_CLOCK5
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI: Clock Invert**

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

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

- **BURST: Burst Signal Selection**

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

- **LDBSTOP: Counter Clock Stopped with RB Loading**

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

1 = Counter clock is stopped when RB loading occurs.

- **LDBDIS: Counter Clock Disable with RB Loading**

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

1 = Counter clock is disabled when RB loading occurs.

- **ETRGEDG: External Trigger Edge Selection**

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

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

0 = TIOB is used as an external trigger.

1 = TIOA is used as an external trigger.

- **CPCTRG: RC Compare Trigger Enable**

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

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

- **WAVE**

0 = Capture Mode is enabled.

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

- **LDRA: RA Loading Selection**

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

- **LDRB: RB Loading Selection**

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

39.6.5 TC Channel Mode Register: Waveform Mode

Name: TC_CMRx [x=0..2] (WAVE = 1)

Addresses: 0xFFF7C004 (0)[0], 0xFFF7C044 (0)[1], 0xFFF7C084 (0)[2]

Access: 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	WAVSEL		ENETRG	EEVT		EEVTEDG	

7	6	5	4	3	2	1	0
CPCDIS	CPCSTOP	BURST		CLKI	TCCLKS		

- **TCCLKS: Clock Selection**

TCCLKS			Clock Selected
0	0	0	TIMER_CLOCK1
0	0	1	TIMER_CLOCK2
0	1	0	TIMER_CLOCK3
0	1	1	TIMER_CLOCK4
1	0	0	TIMER_CLOCK5
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI: Clock Invert**

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

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

- **BURST: Burst Signal Selection**

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

- **CPCSTOP: Counter Clock Stopped with RC Compare**

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

1 = Counter clock is stopped when counter reaches RC.

- **CPCDIS: Counter Clock Disable with RC Compare**

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

1 = Counter clock is disabled when counter reaches RC.

- **EEVTEDG: External Event Edge Selection**

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

- **EEVT: External Event Selection**

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

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

- **ENETRG: External Event Trigger Enable**

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

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

- **WAVSEL: Waveform Selection**

WAVSEL		Effect
0	0	UP mode without automatic trigger on RC Compare
1	0	UP mode with automatic trigger on RC Compare
0	1	UPDOWN mode without automatic trigger on RC Compare
1	1	UPDOWN mode with automatic trigger on RC Compare

- **WAVE**

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

39.6.6 TC Counter Value Register

Name: TC_CVx [x=0..2]

Addresses: 0xFFF7C010 (0)[0], 0xFFF7C050 (0)[1], 0xFFF7C090 (0)[2]

Access: Read-only

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

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

15	14	13	12	11	10	9	8
CV							

7	6	5	4	3	2	1	0
CV							

- **CV: Counter Value**

CV contains the counter value in real time.

39.6.7 TC Register A

Name: TC_RAx [x=0..2]

Addresses: 0xFFF7C014 (0)[0], 0xFFF7C054 (0)[1], 0xFFF7C094 (0)[2]

Access: 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.

39.6.8 TC Register B

Name: TC_RBx [x=0..2]

Addresses: 0xFFF7C018 (0)[0], 0xFFF7C058 (0)[1], 0xFFF7C098 (0)[2]

Access: 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.

39.6.9 TC Register C

Name: TC_RCx [x=0..2]

Addresses: 0xFFF7C01C (0)[0], 0xFFF7C05C (0)[1], 0xFFF7C09C (0)[2]

Access: 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.

39.6.10 TC Status Register

Name: TC_SRx [x=0..2]

Addresses: 0xFFF7C020 (0)[0], 0xFFF7C060 (0)[1], 0xFFF7C0A0 (0)[2]

Access: 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.

39.6.11 TC Interrupt Enable Register

Name: TC_IERx [x=0..2]

Addresses: 0xFFF7C024 (0)[0], 0xFFF7C064 (0)[1], 0xFFF7C0A4 (0)[2]

Access: Write-only

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

39.6.12 TC Interrupt Disable Register

Name: TC_IDRx [x=0..2]

Addresses: 0xFFF7C028 (0)[0], 0xFFF7C068 (0)[1], 0xFFF7C0A8 (0)[2]

Access: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

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.

39.6.13 TC Interrupt Mask Register

Name: TC_IMRx [x=0..2]

Addresses: 0xFFF7C02C (0)[0], 0xFFF7C06C (0)[1], 0xFFF7C0AC (0)[2]

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
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.

40. Controller Area Network (CAN)

40.1 Overview

The CAN controller provides all the features required to implement the serial communication protocol CAN defined by Robert Bosch GmbH, the CAN specification as referred to by ISO/11898A (2.0 Part A and 2.0 Part B) for high speeds and ISO/11519-2 for low speeds. The CAN Controller is able to handle all types of frames (Data, Remote, Error and Overload) and achieves a bitrate of 1 Mbit/sec.

CAN controller accesses are made through configuration registers. 16 independent message objects (mailboxes) are implemented.

Any mailbox can be programmed as a reception buffer block (even non-consecutive buffers). For the reception of defined messages, one or several message objects can be masked without participating in the buffer feature. An interrupt is generated when the buffer is full. According to the mailbox configuration, the first message received can be locked in the CAN controller registers until the application acknowledges it, or this message can be discarded by new received messages.

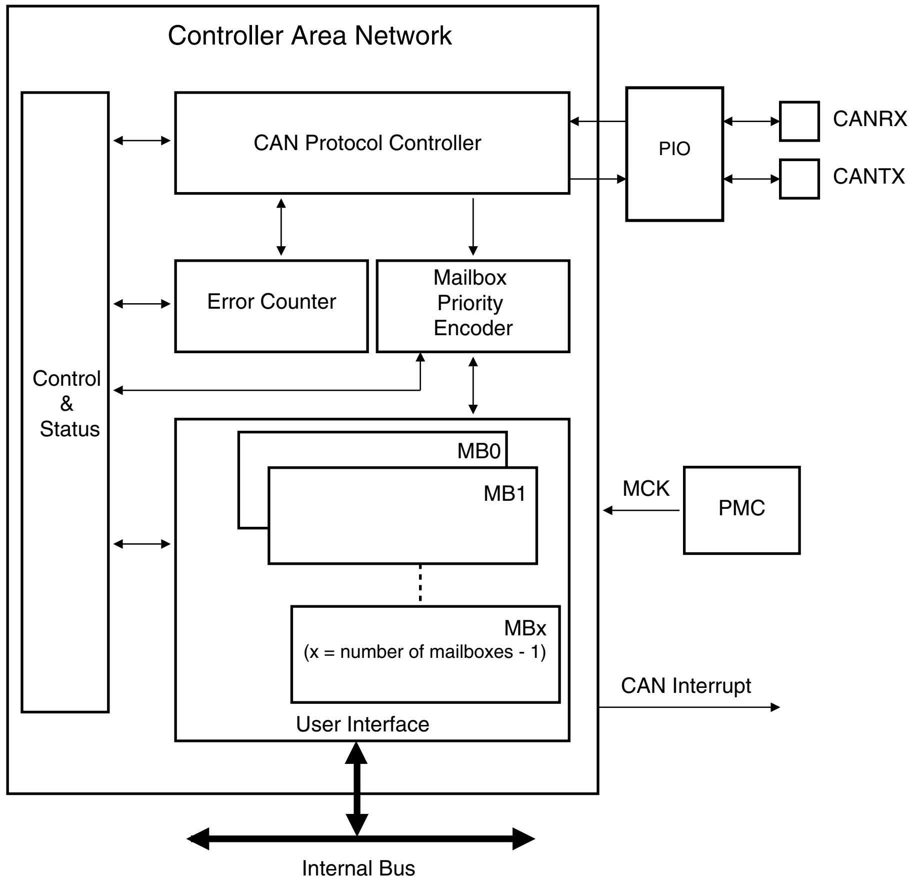
Any mailbox can be programmed for transmission. Several transmission mailboxes can be enabled in the same time. A priority can be defined for each mailbox independently.

An internal 16-bit timer is used to stamp each received and sent message. This timer starts counting as soon as the CAN controller is enabled. This counter can be reset by the application or automatically after a reception in the last mailbox in Time Triggered Mode.

The CAN controller offers optimized features to support the Time Triggered Communication (TTC) protocol.

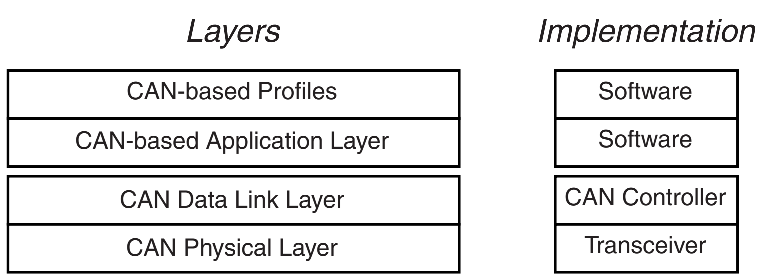
40.2 Block Diagram

Figure 40-1. CAN Block Diagram



40.3 Application Block Diagram

Figure 40-2. Application Block Diagram



40.4 I/O Lines Description

Table 40-1. I/O Lines Description

Name	Description	Type
CANRX	CAN Receive Serial Data	Input
CANTX	CAN Transmit Serial Data	Output

40.5 Product Dependencies

40.5.1 I/O Lines

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

Table 40-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
CAN	CANRX	PA13	A
CAN	CANTX	PA12	A

40.5.2 Power Management

The programmer must first enable the CAN clock in the Power Management Controller (PMC) before using the CAN.

A Low-power Mode is defined for the CAN controller: If the application does not require CAN operations, the CAN clock can be stopped when not needed and be restarted later. Before stopping the clock, the CAN Controller must be in Low-power Mode to complete the current transfer. After restarting the clock, the application must disable the Low-power Mode of the CAN controller.

40.5.3 Interrupt

The CAN interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the CAN interrupt requires the AIC to be programmed first. Note that it is not recommended to use the CAN interrupt line in edge-sensitive mode.

Table 40-3. Peripheral IDs

Instance	ID
CAN	13

40.6 CAN Controller Features

40.6.1 CAN Protocol Overview

The Controller Area Network (CAN) is a multi-master serial communication protocol that efficiently supports real-time control with a very high level of security with bit rates up to 1 Mbit/s.

The CAN protocol supports four different frame types:

- Data frames: They carry data from a transmitter node to the receiver nodes. The overall maximum data frame length is 108 bits for a standard frame and 128 bits for an extended frame.
- Remote frames: A destination node can request data from the source by sending a remote frame with an identifier that matches the identifier of the required data frame. The appropriate data source node then sends a data frame as a response to this node request.
- Error frames: An error frame is generated by any node that detects a bus error.
- Overload frames: They provide an extra delay between the preceding and the successive data frames or remote frames.

The Atmel CAN controller provides the CPU with full functionality of the CAN protocol V2.0 Part A and V2.0 Part B. It minimizes the CPU load in communication overhead. The Data Link Layer and part of the physical layer are automatically handled by the CAN controller itself.

The CPU reads or writes data or messages via the CAN controller mailboxes. An identifier is assigned to each mailbox. The CAN controller encapsulates or decodes data messages to build or to decode bus data frames. Remote frames, error frames and overload frames are automatically handled by the CAN controller under supervision of the software application.

40.6.2 Mailbox Organization

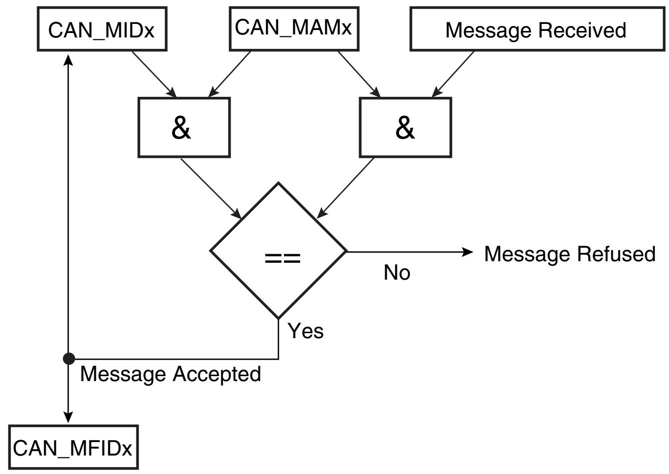
The CAN module has 16 buffers, also called channels or mailboxes. An identifier that corresponds to the CAN identifier is defined for each active mailbox. Message identifiers can match the standard frame identifier or the extended frame identifier. This identifier is defined for the first time during the CAN initialization, but can be dynamically reconfigured later so that the mailbox can handle a new message family. Several mailboxes can be configured with the same ID.

Each mailbox can be configured in receive or in transmit mode independently. The mailbox object type is defined in the MOT field of the CAN_MMRx register.

40.6.2.1 Message Acceptance Procedure

If the MIDE field in the CAN_MIDx register is set, the mailbox can handle the extended format identifier; otherwise, the mailbox handles the standard format identifier. Once a new message is received, its ID is masked with the CAN_MAMx value and compared with the CAN_MIDx value. If accepted, the message ID is copied to the CAN_MIDx register.

Figure 40-3. Message Acceptance Procedure



If a mailbox is dedicated to receiving several messages (a family of messages) with different IDs, the acceptance mask defined in the CAN_MAMx register must mask the variable part of the ID family. Once a message is received, the application must decode the masked bits in the CAN_MIDx. To speed up the decoding, masked bits are grouped in the family ID register (CAN_MFIDx).

For example, if the following message IDs are handled by the same mailbox:

```
ID0 101000100100010010000100 0 11 00b
ID1 101000100100010010000100 0 11 01b
ID2 101000100100010010000100 0 11 10b
ID3 101000100100010010000100 0 11 11b
ID4 101000100100010010000100 1 11 00b
ID5 101000100100010010000100 1 11 01b
ID6 101000100100010010000100 1 11 10b
ID7 101000100100010010000100 1 11 11b
```

The CAN_MIDx and CAN_MAMx of Mailbox x must be initialized to the corresponding values:

```
CAN_MIDx = 001 101000100100010010000100 x 11 xxb
CAN_MAMx = 001 111111111111111111111111 0 11 00b
```

If Mailbox x receives a message with ID6, then CAN_MIDx and CAN_MFIDx are set:

```
CAN_MIDx = 001 101000100100010010000100 1 11 10b
CAN_MFIDx = 00000000000000000000000000000110b
```

If the application associates a handler for each message ID, it may define an array of pointers to functions:

```
void (*pHandler[8])(void);
```

When a message is received, the corresponding handler can be invoked using CAN_MFIDx register and there is no need to check masked bits:

```
unsigned int MFID0_register;
MFID0_register = Get_CAN_MFID0_Register();
// Get_CAN_MFID0_Register() returns the value of the CAN_MFID0 register
pHandler[MFID0_register]();
```

40.6.2.2 Receive Mailbox

When the CAN module receives a message, it looks for the first available mailbox with the lowest number and compares the received message ID with the mailbox ID. If such a mailbox is found, then the message is stored in its data registers. Depending on the configuration, the mailbox is disabled as long as the message has not been acknowledged by the application (Receive only), or, if new messages with the same ID are received, then they overwrite the previous ones (Receive with overwrite).

It is also possible to configure a mailbox in Consumer Mode. In this mode, after each transfer request, a remote frame is automatically sent. The first answer received is stored in the corresponding mailbox data registers.

Several mailboxes can be chained to receive a buffer. They must be configured with the same ID in Receive Mode, except for the last one, which can be configured in Receive with Overwrite Mode. The last mailbox can be used to detect a buffer overflow.

Mailbox Object Type	Description
Receive	The first message received is stored in mailbox data registers. Data remain available until the next transfer request.
Receive with overwrite	The last message received is stored in mailbox data register. The next message always overwrites the previous one. The application has to check whether a new message has not overwritten the current one while reading the data registers.
Consumer	A remote frame is sent by the mailbox. The answer received is stored in mailbox data register. This extends Receive mailbox features. Data remain available until the next transfer request.

40.6.2.3 Transmit Mailbox

When transmitting a message, the message length and data are written to the transmit mailbox with the correct identifier. For each transmit mailbox, a priority is assigned. The controller automatically sends the message with the highest priority first (set with the field PRIOR in CAN_MMRx register).

It is also possible to configure a mailbox in Producer Mode. In this mode, when a remote frame is received, the mailbox data are sent automatically. By enabling this mode, a producer can be done using only one mailbox instead of two: one to detect the remote frame and one to send the answer.

Mailbox Object Type	Description
Transmit	The message stored in the mailbox data registers will try to win the bus arbitration immediately or later according to or not the Time Management Unit configuration (see Section 40.6.3). The application is notified that the message has been sent or aborted.
Producer	The message prepared in the mailbox data registers will be sent after receiving the next remote frame. This extends transmit mailbox features.

40.6.3 Time Management Unit

The CAN Controller integrates a free-running 16-bit internal timer. The counter is driven by the bit clock of the CAN bus line. It is enabled when the CAN controller is enabled (CANEN set in the CAN_MR register). It is automatically cleared in the following cases:

- after a reset
- when the CAN controller is in Low-power Mode is enabled (LPM bit set in the CAN_MR and SLEEP bit set in the CAN_SR)
- after a reset of the CAN controller (CANEN bit in the CAN_MR register)
- in Time-triggered Mode, when a message is accepted by the last mailbox (rising edge of the MRDY signal in the $CAN\_MSR_{last\_mailbox\_number}$ register).

The application can also reset the internal timer by setting TIMRST in the CAN_TCR register. The current value of the internal timer is always accessible by reading the CAN_TIM register.

When the timer rolls-over from FFFFh to 0000h, TOVF (Timer Overflow) signal in the CAN_SR register is set. TOVF bit in the CAN_SR register is cleared by reading the CAN_SR register. Depending on the corresponding interrupt mask in the CAN_IMR register, an interrupt is generated while TOVF is set.

In a CAN network, some CAN devices may have a larger counter. In this case, the application can also decide to freeze the internal counter when the timer reaches FFFFh and to wait for a restart condition from another device. This feature is enabled by setting TIMFRZ in the CAN_MR register. The CAN_TIM register is frozen to the FFFFh value. A clear condition described above restarts the timer. A timer overflow (TOVF) interrupt is triggered.

To monitor the CAN bus activity, the CAN_TIM register is copied to the CAN _TIMESTP register after each start of frame or end of frame and a TSTP interrupt is triggered. If TEOF bit in the CAN_MR register is set, the value is captured at each End Of Frame, else it is captured at each Start Of Frame. Depending on the corresponding mask in the CAN_IMR register, an interrupt is generated while TSTP is set in the CAN_SR. TSTP bit is cleared by reading the CAN_SR register.

The time management unit can operate in one of the two following modes:

- Timestamping mode: The value of the internal timer is captured at each Start Of Frame or each End Of Frame
- Time Triggered mode: A mailbox transfer operation is triggered when the internal timer reaches the mailbox trigger.

Timestamping Mode is enabled by clearing TTM field in the CAN_MR register. Time Triggered Mode is enabled by setting TTM field in the CAN_MR register.

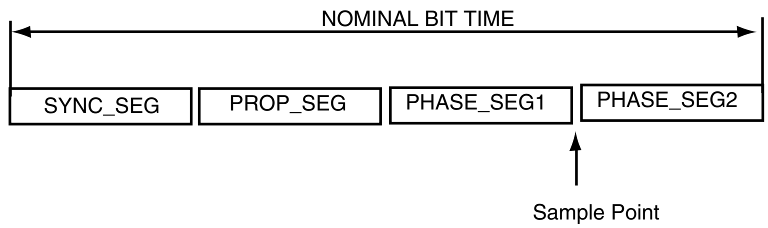
40.6.4 CAN 2.0 Standard Features

40.6.4.1 CAN Bit Timing Configuration

All controllers on a CAN bus must have the same bit rate and bit length. At different clock frequencies of the individual controllers, the bit rate has to be adjusted by the time segments.

The CAN protocol specification partitions the nominal bit time into four different segments:

Figure 40-4. Partition of the CAN Bit Time



- TIME QUANTUM

The TIME QUANTUM (TQ) is a fixed unit of time derived from the MCK period. The total number of TIME QUANTA in a bit time is programmable from 8 to 25.

SYNC SEG: SYNChronization Segment.

This part of the bit time is used to synchronize the various nodes on the bus. An edge is expected to lie within this segment. It is 1 TQ long.

- PROP SEG: PROPagation Segment.

This part of the bit time is used to compensate for the physical delay times within the network. It is twice the sum of the signal's propagation time on the bus line, the input comparator delay, and the output driver delay. It is programmable to be 1,2,..., 8 TQ long.

This parameter is defined in the PROPAG field of the ”CAN Baudrate Register”.

- PHASE SEG1, PHASE SEG2: PHASE Segment 1 and 2.

The Phase-Buffer-Segments are used to compensate for edge phase errors. These segments can be lengthened (PHASE SEG1) or shortened (PHASE SEG2) by resynchronization.

Phase Segment 1 is programmable to be 1,2,..., 8 TQ long.

Phase Segment 2 length has to be at least as long as the Information Processing Time (IPT) and may not be more than the length of Phase Segment 1.

These parameters are defined in the PHASE1 and PHASE2 fields of the ”CAN Baudrate Register”.

- INFORMATION PROCESSING TIME:

The Information Processing Time (IPT) is the time required for the logic to determine the bit level of a sampled bit. The IPT begins at the sample point, is measured in TQ and **is fixed at 2 TQ for the Atmel CAN**. Since Phase Segment 2 also begins at the sample point and is the last segment in the bit time, PHASE SEG2 shall not be less than the IPT.

- SAMPLE POINT:

The SAMPLE POINT is the point in time at which the bus level is read and interpreted as the value of that respective bit. Its location is at the end of PHASE_SEG1.

- SJW: ReSynchronization Jump Width.

The ReSynchronization Jump Width defines the limit to the amount of lengthening or shortening of the Phase Segments.

SJW is programmable to be the minimum of PHASE SEG1 and 4 TQ.

If the SMP field in the CAN_BR register is set, then the incoming bit stream is sampled three times with a period of half a CAN clock period, centered on sample point.

In the CAN controller, the length of a bit on the CAN bus is determined by the parameters (BRP, PROPAG, PHASE1 and PHASE2).

$$t_{BIT} = t_{CSC} + t_{PRS} + t_{PHS1} + t_{PHS2}$$

The time quantum is calculated as follows:

$$t_{CSC} = (BRP + 1) / MCK$$

Note: The BRP field must be within the range [1, 0x7F], i.e., BRP = 0 is not authorized.

$$t_{PRS} = t_{CSC} \times (PROPAG + 1)$$

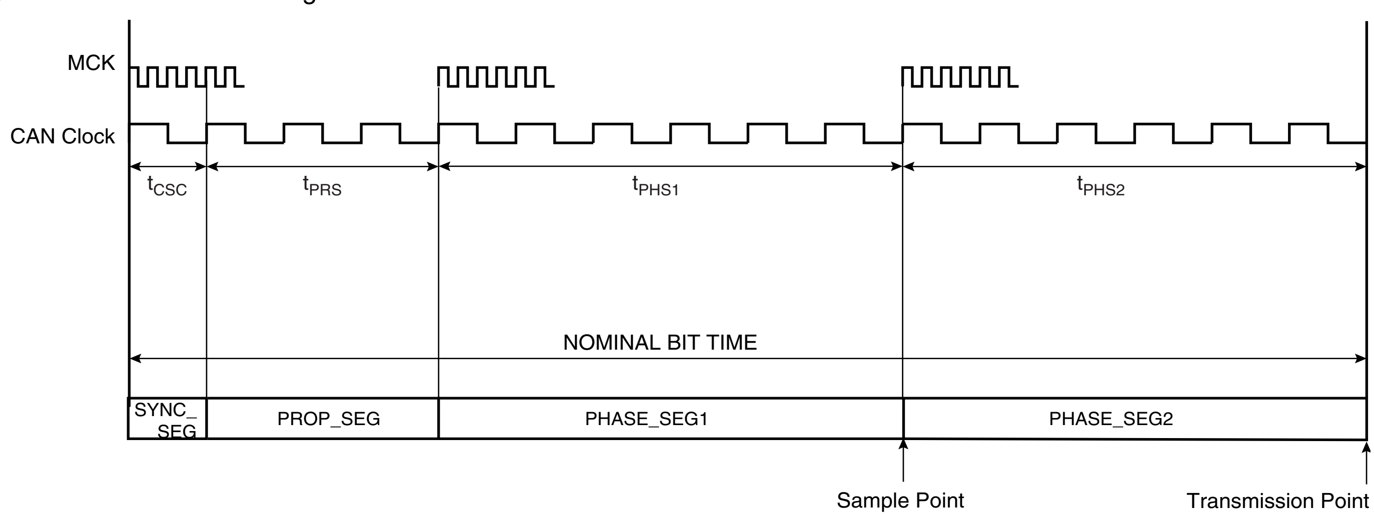
$$t_{PHS1} = t_{CSC} \times (PHASE1 + 1)$$

$$t_{PHS2} = t_{CSC} \times (PHASE2 + 1)$$

To compensate for phase shifts between clock oscillators of different controllers on the bus, the CAN controller must resynchronize on any relevant signal edge of the current transmission. The resynchronization shortens or lengthens the bit time so that the position of the sample point is shifted with regard to the detected edge. The resynchronization jump width (SJW) defines the maximum of time by which a bit period may be shortened or lengthened by resynchronization.

$$t_{SJW} = t_{CSC} \times (SJW + 1)$$

Figure 40-5. CAN Bit Timing



Example of bit timing determination for CAN baudrate of 500 Kbit/s:

```
MCK = 48MHz
CAN baudrate= 500kbit/s => bit time= 2us
Delay of the bus driver: 50 ns
Delay of the receiver: 30ns
Delay of the bus line (20m): 110ns

The total number of time quanta in a bit time must be comprised between 8
and 25. If we fix the bit time to 16 time quanta:
Tcsc = 1 time quanta = bit time / 16 = 125 ns
=> BRP = (Tcsc x MCK) - 1 = 5

The propagation segment time is equal to twice the sum of the signal's
propagation time on the bus line, the receiver delay and the output driver
delay:
Tprs = 2 * (50+30+110) ns = 380 ns = 3 Tcsc
=> PROPAG = Tprs/Tcsc - 1 = 2
The remaining time for the two phase segments is:
Tphs1 + Tphs2 = bit time - Tcsc - Tprs = (16 - 1 - 3)Tcsc
Tphs1 + Tphs2 = 12 Tcsc

Because this number is even, we choose Tphs2 = Tphs1 (else we would choose
Tphs2 = Tphs1 + Tcsc)
Tphs1 = Tphs2 = (12/2) Tcsc = 6 Tcsc
=> PHASE1 = PHASE2 = Tphs1/Tcsc - 1 = 5

The resynchronization jump width must be comprised between 1 Tcsc and the
minimum of 4 Tcsc and Tphs1. We choose its maximum value:
Tsjw = Min(4 Tcsc,Tphs1) = 4 Tcsc
=> SJW = Tsjw/Tcsc - 1 = 3

Finally: CAN_BR = 0x00053255
```

40.6.4.2 CAN Bus Synchronization

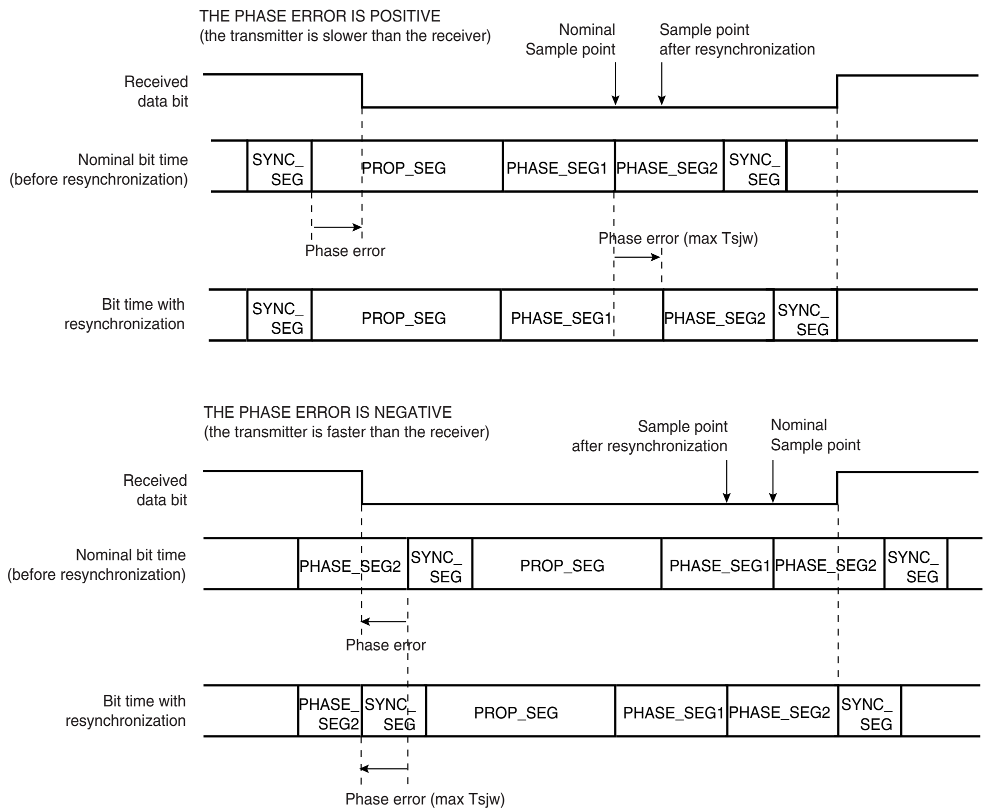
Two types of synchronization are distinguished: “hard synchronization” at the start of a frame and “resynchronization” inside a frame. After a hard synchronization, the bit time is restarted with the end of the SYNC_SEG segment, regardless of the phase error. Resynchronization causes a reduction or increase in the bit time so that the position of the sample point is shifted with respect to the detected edge.

The effect of resynchronization is the same as that of hard synchronization when the magnitude of the phase error of the edge causing the resynchronization is less than or equal to the programmed value of the resynchronization jump width (t_{SJW}).

When the magnitude of the phase error is larger than the resynchronization jump width and

- the phase error is positive, then PHASE_SEG1 is lengthened by an amount equal to the resynchronization jump width.
- the phase error is negative, then PHASE_SEG2 is shortened by an amount equal to the resynchronization jump width.

Figure 40-6. CAN Resynchronization



40.6.4.3 Autobaud Mode

The autobaud feature is enabled by setting the ABM field in the CAN_MR register. In this mode, the CAN controller is only listening to the line without acknowledging the received messages. It can not send any message. The errors flags are updated. The bit timing can be adjusted until no error occurs (good configuration found). In this mode, the error counters are frozen. To go back to the standard mode, the ABM bit must be cleared in the CAN_MR register.

40.6.4.4 Error Detection

There are five different error types that are not mutually exclusive. Each error concerns only specific fields of the CAN data frame (refer to the Bosch CAN specification for their correspondence):

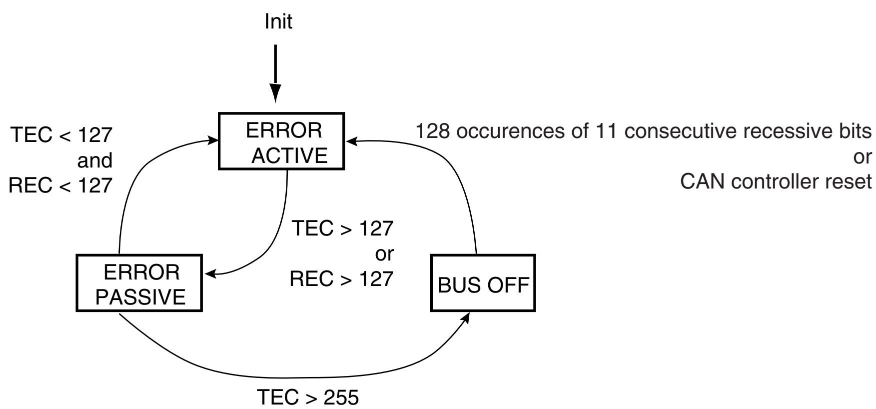
- CRC error (CERR bit in the CAN_SR register): With the CRC, the transmitter calculates a checksum for the CRC bit sequence from the Start of Frame bit until the end of the Data Field. This CRC sequence is transmitted in the CRC field of the Data or Remote Frame.
- Bit-stuffing error (SERR bit in the CAN_SR register): If a node detects a sixth consecutive equal bit level during the bit-stuffing area of a frame, it generates an Error Frame starting with the next bit-time.
- Bit error (BERR bit in CAN_SR register): A bit error occurs if a transmitter sends a dominant bit but detects a recessive bit on the bus line, or if it sends a recessive bit but detects a dominant bit on the bus line. An error frame is generated and starts with the next bit time.
- Form Error (FERR bit in the CAN_SR register): If a transmitter detects a dominant bit in one of the fix-formatted segments CRC Delimiter, ACK Delimiter or End of Frame, a form error has occurred and an error frame is generated.

- Acknowledgment error (AERR bit in the CAN_SR register): The transmitter checks the Acknowledge Slot, which is transmitted by the transmitting node as a recessive bit, contains a dominant bit. If this is the case, at least one other node has received the frame correctly. If not, an Acknowledge Error has occurred and the transmitter will start in the next bit-time an Error Frame transmission.

40.6.4.5 Fault Confinement

To distinguish between temporary and permanent failures, every CAN controller has two error counters: REC (Receive Error Counter) and TEC (Transmit Error Counter). The two counters are incremented upon detected errors and are decremented upon correct transmissions or receptions, respectively. Depending on the counter values, the state of the node changes: the initial state of the CAN controller is Error Active, meaning that the controller can send Error Active flags. The controller changes to the Error Passive state if there is an accumulation of errors. If the CAN controller fails or if there is an extreme accumulation of errors, there is a state transition to Bus Off.

Figure 40-7. Line Error Mode



An error active unit takes part in bus communication and sends an active error frame when the CAN controller detects an error.

An error passive unit cannot send an active error frame. It takes part in bus communication, but when an error is detected, a passive error frame is sent. Also, after a transmission, an error passive unit waits before initiating further transmission.

A bus off unit is not allowed to have any influence on the bus.

For fault confinement, two errors counters (TEC and REC) are implemented. These counters are accessible via the CAN_ECR register. The state of the CAN controller is automatically updated according to these counter values. If the CAN controller is in Error Active state, then the ERRA bit is set in the CAN_SR register. The corresponding interrupt is pending while the interrupt is not masked in the CAN_IMR register. If the CAN controller is in Error Passive Mode, then the ERRP bit is set in the CAN_SR register and an interrupt remains pending while the ERRP bit is set in the CAN_IMR register. If the CAN is in Bus Off Mode, then the BOFF bit is set in the CAN_SR register. As for ERRP and ERRA, an interrupt is pending while the BOFF bit is set in the CAN_IMR register.

When one of the error counters values exceeds 96, an increased error rate is indicated to the controller through the WARN bit in CAN_SR register, but the node remains error active. The corresponding interrupt is pending while the interrupt is set in the CAN_IMR register.

Refer to the Bosch CAN specification v2.0 for details on fault confinement.

40.6.4.6 Error Interrupt Handler

WARN, BOFF, ERRA and ERRP (CAN_SR) represent the current status of the CAN bus and are not latched. They reflect the current TEC and REC (CAN_ECR) values as described in Section 40.6.4.5 "Fault Confinement" on page 742.

Based on that, if these bits are used as an interrupt, the user can enter into an interrupt and not see the corresponding status register if the TEC and REC counter have changed their state. When entering Bus Off Mode, the only way to exit from this state is 128 occurrences of 11 consecutive recessive bits or a CAN controller reset.

In Error Active Mode, the user reads:

- ERRA =1
- ERRP = 0
- BOFF = 0

In Error Passive Mode, the user reads:

- ERRA = 0
- ERRP =1
- BOFF = 0

In Bus Off Mode, the user reads:

- ERRA = 0
- ERRP =1
- BOFF =1

The CAN interrupt handler should do the following:

- Only enable one error mode interrupt at a time.
- Look at and check the REC and TEC values in the interrupt handler to determine the current state.

40.6.4.7 Overload

The overload frame is provided to request a delay of the next data or remote frame by the receiver node ("Request overload frame") or to signal certain error conditions ("Reactive overload frame") related to the intermission field respectively.

Reactive overload frames are transmitted after detection of the following error conditions:

- Detection of a dominant bit during the first two bits of the intermission field
- Detection of a dominant bit in the last bit of EOF by a receiver, or detection of a dominant bit by a receiver or a transmitter at the last bit of an error or overload frame delimiter

The CAN controller can generate a request overload frame automatically after each message sent to one of the CAN controller mailboxes. This feature is enabled by setting the OVL bit in the CAN_MR register.

Reactive overload frames are automatically handled by the CAN controller even if the OVL bit in the CAN_MR register is not set. An overload flag is generated in the same way as an error flag, but error counters do not increment.

40.6.5 Low-power Mode

In Low-power Mode, the CAN controller cannot send or receive messages. All mailboxes are inactive.

In Low-power Mode, the SLEEP signal in the CAN_SR register is set; otherwise, the WAKEUP signal in the CAN_SR register is set. These two fields are exclusive except after a CAN controller reset (WAKEUP and SLEEP are stuck at 0 after a reset). After power-up reset, the Low-power Mode is disabled and the WAKEUP bit is set in the CAN_SR register only after detection of 11 consecutive recessive bits on the bus.

40.6.5.1 Enabling Low-power Mode

A software application can enable Low-power Mode by setting the LPM bit in the CAN_MR global register. The CAN controller enters Low-power Mode once all pending transmit messages are sent.

When the CAN controller enters Low-power Mode, the SLEEP signal in the CAN_SR register is set. Depending on the corresponding mask in the CAN_IMR register, an interrupt is generated while SLEEP is set.

The SLEEP signal in the CAN_SR register is automatically cleared once WAKEUP is set. The WAKEUP signal is automatically cleared once SLEEP is set.

Reception is disabled while the SLEEP signal is set to one in the CAN_SR register. It is important to note that those messages with higher priority than the last message transmitted can be received between the LPM command and entry in Low-power Mode.

Once in Low-power Mode, the CAN controller clock can be switched off by programming the chip's Power Management Controller (PMC). The CAN controller drains only the static current.

Error counters are disabled while the SLEEP signal is set to one.

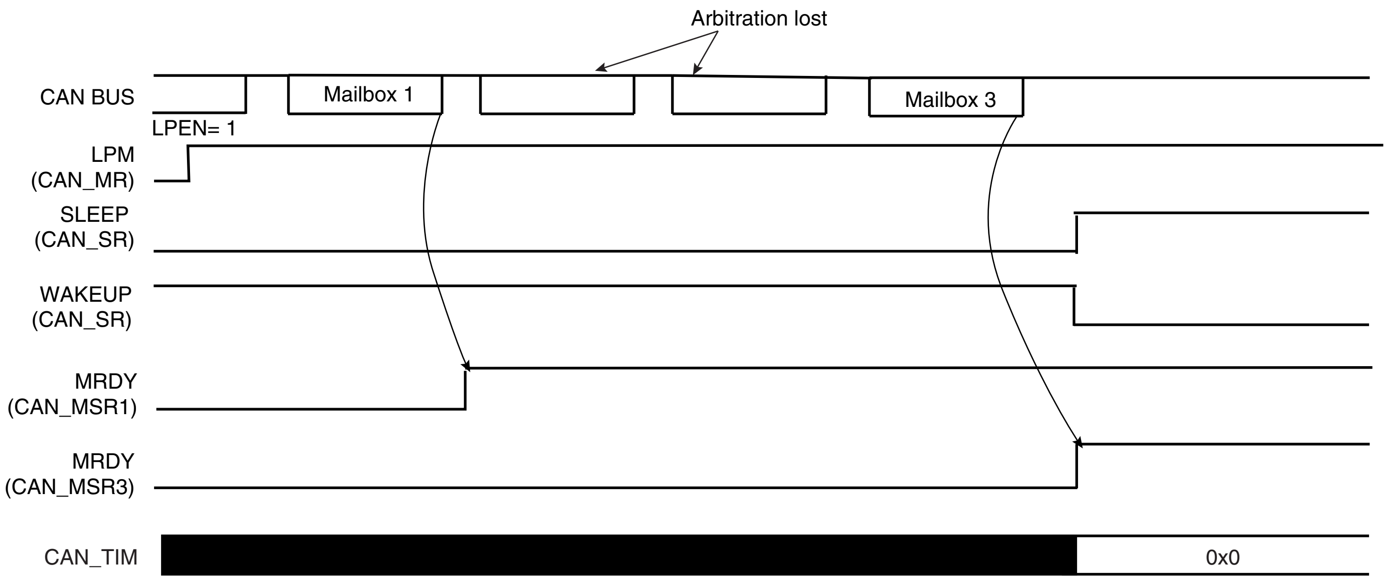
Thus, to enter Low-power Mode, the software application must:

– Set LPM field in the CAN_MR register

– Wait for SLEEP signal rising

Now the CAN Controller clock can be disabled. This is done by programming the Power Management Controller (PMC).

Figure 40-8. Enabling Low-power Mode



40.6.5.2 Disabling Low-power Mode

The CAN controller can be awake after detecting a CAN bus activity. Bus activity detection is done by an external module that may be embedded in the chip. When it is notified of a CAN bus activity, the software application disables Low-power Mode by programming the CAN controller.

To disable Low-power Mode, the software application must:

– Enable the CAN Controller clock. This is done by programming the Power Management Controller (PMC).

– Clear the LPM field in the CAN_MR register

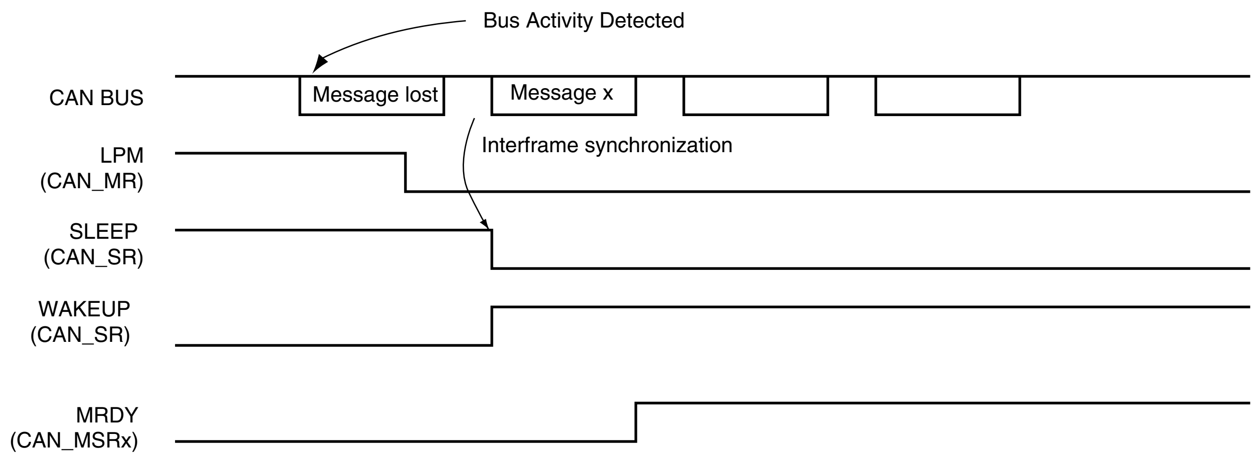
The CAN controller synchronizes itself with the bus activity by checking for eleven consecutive "recessive" bits. Once synchronized, the WAKEUP signal in the CAN_SR register is set.

Depending on the corresponding mask in the CAN_IMR register, an interrupt is generated while WAKEUP is set. The SLEEP signal in the CAN_SR register is automatically cleared once WAKEUP is set. WAKEUP signal is automatically cleared once SLEEP is set.

If no message is being sent on the bus, then the CAN controller is able to send a message eleven bit times after disabling Low-power Mode.

If there is bus activity when Low-power mode is disabled, the CAN controller is synchronized with the bus activity in the next interframe. The previous message is lost (see Figure 40-9).

Figure 40-9. Disabling Low-power Mode



40.7 Functional Description

40.7.1 CAN Controller Initialization

After power-up reset, the CAN controller is disabled. The CAN controller clock must be activated by the Power Management Controller (PMC) and the CAN controller interrupt line must be enabled by the interrupt controller (AIC).

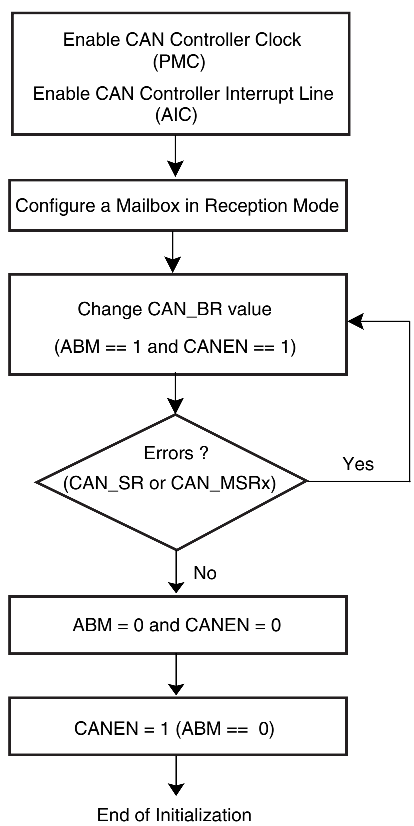
The CAN controller must be initialized with the CAN network parameters. The CAN_BR register defines the sampling point in the bit time period. CAN_BR must be set before the CAN controller is enabled by setting the CANEN field in the CAN_MR register.

The CAN controller is enabled by setting the CANEN flag in the CAN_MR register. At this stage, the internal CAN controller state machine is reset, error counters are reset to 0, error flags are reset to 0.

Once the CAN controller is enabled, bus synchronization is done automatically by scanning eleven recessive bits. The WAKEUP bit in the CAN_SR register is automatically set to 1 when the CAN controller is synchronized (WAKEUP and SLEEP are stuck at 0 after a reset).

The CAN controller can start listening to the network in Autobaud Mode. In this case, the error counters are locked and a mailbox may be configured in Receive Mode. By scanning error flags, the CAN_BR register values synchronized with the network. Once no error has been detected, the application disables the Autobaud Mode, clearing the ABM field in the CAN_MR register.

Figure 40-10. Possible Initialization Procedure



40.7.2 CAN Controller Interrupt Handling

There are two different types of interrupts. One type of interrupt is a message-object related interrupt, the other is a system interrupt that handles errors or system-related interrupt sources.

All interrupt sources can be masked by writing the corresponding field in the CAN_IDR register. They can be unmasked by writing to the CAN_IER register. After a power-up reset, all interrupt sources are disabled (masked). The current mask status can be checked by reading the CAN_IMR register.

The CAN_SR register gives all interrupt source states.

The following events may initiate one of the two interrupts:

- Message object interrupt
  - Data registers in the mailbox object are available to the application. In Receive Mode, a new message was received. In Transmit Mode, a message was transmitted successfully.
  - A sent transmission was aborted.
- System interrupts
  - Bus off interrupt: The CAN module enters the bus off state.
  - Error passive interrupt: The CAN module enters Error Passive Mode.
  - Error Active Mode: The CAN module is neither in Error Passive Mode nor in Bus Off mode.
  - Warn Limit interrupt: The CAN module is in Error-active Mode, but at least one of its error counter value exceeds 96.
  - Wake-up interrupt: This interrupt is generated after a wake-up and a bus synchronization.
  - Sleep interrupt: This interrupt is generated after a Low-power Mode enable once all pending messages in transmission have been sent.
  - Internal timer counter overflow interrupt: This interrupt is generated when the internal timer rolls over.
  - Timestamp interrupt: This interrupt is generated after the reception or the transmission of a start of frame or an end of frame. The value of the internal counter is copied in the CAN_TIMESTP register.

All interrupts are cleared by clearing the interrupt source except for the internal timer counter overflow interrupt and the timestamp interrupt. These interrupts are cleared by reading the CAN_SR register.

40.7.3 CAN Controller Message Handling

40.7.3.1 Receive Handling

Two modes are available to configure a mailbox to receive messages. In ***Receive Mode***, the first message received is stored in the mailbox data register. In ***Receive with Overwrite Mode***, the last message received is stored in the mailbox.

40.7.3.2 Simple Receive Mailbox

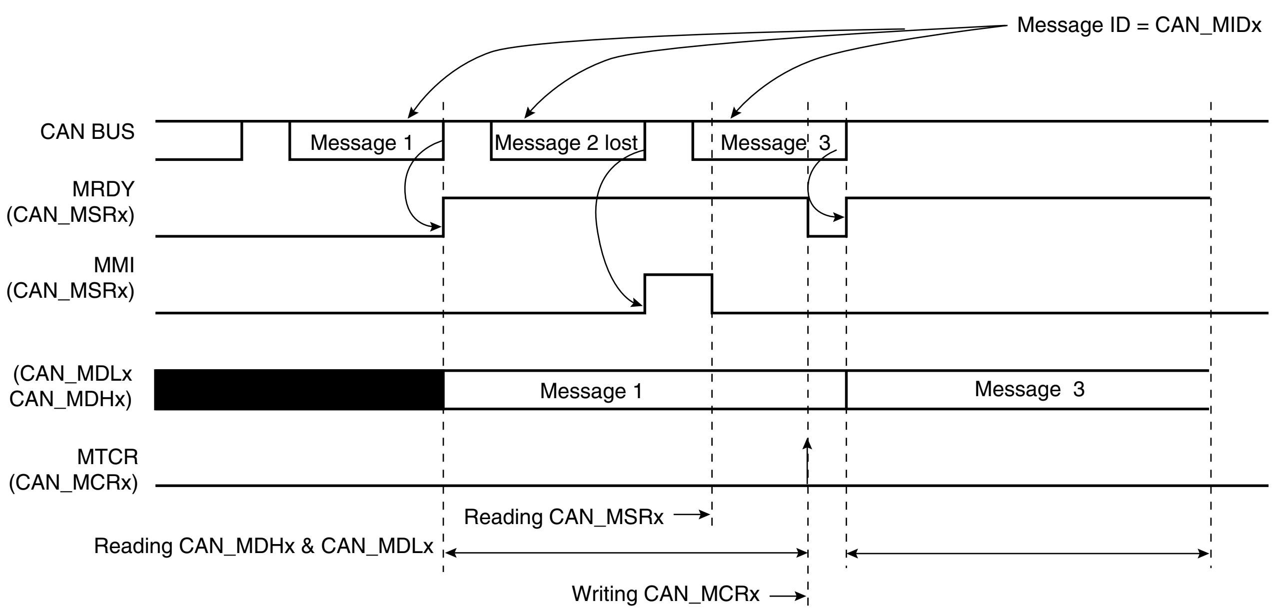
A mailbox is in Receive Mode once the MOT field in the CAN_MMRx register has been configured. Message ID and Message Acceptance Mask must be set before the Receive Mode is enabled.

After Receive Mode is enabled, the MRDY flag in the CAN_MSR register is automatically cleared until the first message is received. When the first message has been accepted by the mailbox, the MRDY flag is set. An interrupt is pending for the mailbox while the MRDY flag is set. This interrupt can be masked depending on the mailbox flag in the CAN_IMR global register.

Message data are stored in the mailbox data register until the software application notifies that data processing has ended. This is done by asking for a new transfer command, setting the MTCR flag in the CAN_MCRx register. This automatically clears the MRDY signal.

The MMI flag in the CAN_MSRx register notifies the software that a message has been lost by the mailbox. This flag is set when messages are received while MRDY is set in the CAN_MSRx register. This flag is cleared by reading the CAN_MSRs register. A receive mailbox prevents from overwriting the first message by new ones while MRDY flag is set in the CAN_MSRx register. See Figure 40-11.

Figure 40-11. Receive Mailbox



Note: In the case of ARM architecture, CAN_MSRx, CAN_MDLx, CAN_MDHx can be read using an optimized ldm assembler instruction.

40.7.3.3 *Receive with Overwrite Mailbox*

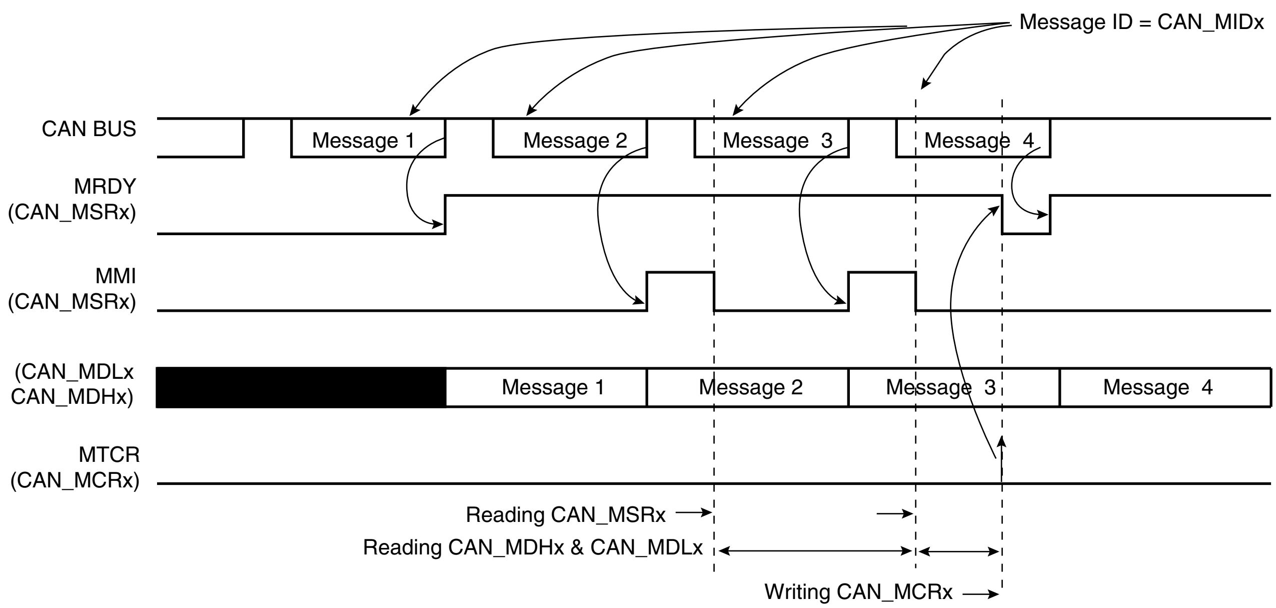
A mailbox is in Receive with Overwrite Mode once the MOT field in the CAN_MMRx register has been configured. Message ID and Message Acceptance masks must be set before Receive Mode is enabled.

After Receive Mode is enabled, the MRDY flag in the CAN_MSR register is automatically cleared until the first message is received. When the first message has been accepted by the mailbox, the MRDY flag is set. An interrupt is pending for the mailbox while the MRDY flag is set. This interrupt is masked depending on the mailbox flag in the CAN_IMR global register.

If a new message is received while the MRDY flag is set, this new message is stored in the mailbox data register, overwriting the previous message. The MMI flag in the CAN_MSRx register notifies the software that a message has been dropped by the mailbox. This flag is cleared when reading the CAN_MSRx register.

The CAN controller may store a new message in the CAN data registers while the application reads them. To check that CAN_MDHx and CAN_MDLx do not belong to different messages, the application must check the MMI field in the CAN_MSRx register before and after reading CAN_MDHx and CAN_MDLx. If the MMI flag is set again after the data registers have been read, the software application has to re-read CAN_MDHx and CAN_MDLx (see Figure 40-12).

Figure 40-12. Receive with Overwrite Mailbox

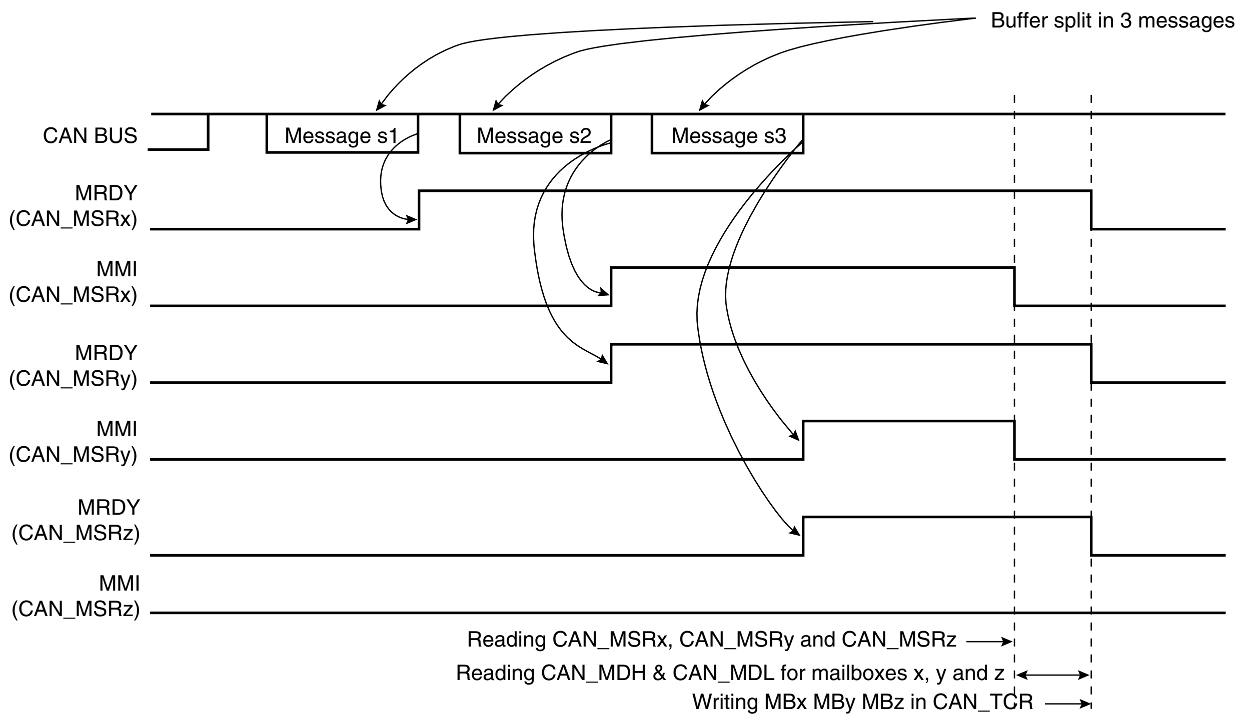


40.7.3.4 Chaining Mailboxes

Several mailboxes may be used to receive a buffer split into several messages with the same ID. In this case, the mailbox with the lowest number is serviced first. In the receive and receive with overwrite modes, the field PRIOR in the CAN_MMRx register has no effect. If Mailbox 0 and Mailbox 5 accept messages with the same ID, the first message is received by Mailbox 0 and the second message is received by Mailbox 5. Mailbox 0 must be configured in Receive Mode (i.e., the first message received is considered) and Mailbox 5 must be configured in Receive with Overwrite Mode. Mailbox 0 cannot be configured in Receive with Overwrite Mode; otherwise, all messages are accepted by this mailbox and Mailbox 5 is never serviced.

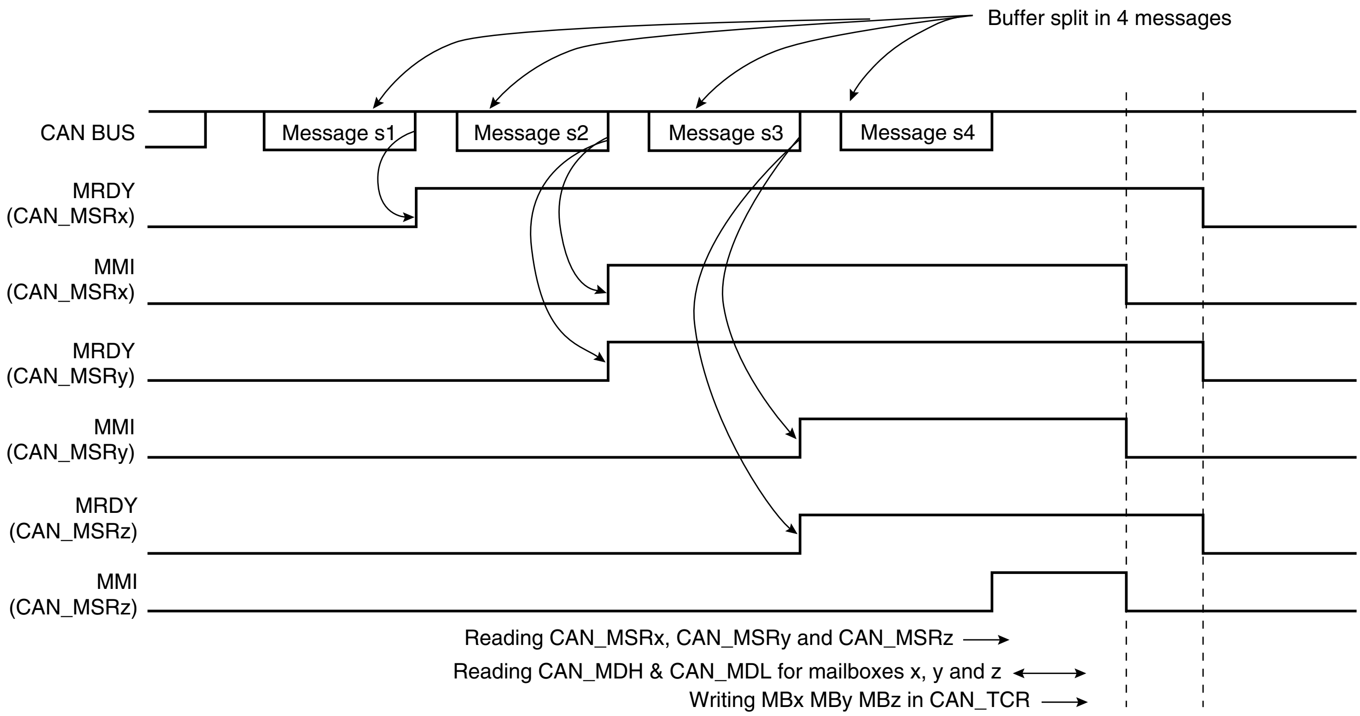
If several mailboxes are chained to receive a buffer split into several messages, all mailboxes except the last one (with the highest number) must be configured in Receive Mode. The first message received is handled by the first mailbox, the second one is refused by the first mailbox and accepted by the second mailbox, the last message is accepted by the last mailbox and refused by previous ones (see Figure 40-13).

Figure 40-13. Chaining Three Mailboxes to Receive a Buffer Split into Three Messages



If the number of mailboxes is not sufficient (the MMI flag of the last mailbox raises), the user must read each data received on the last mailbox in order to retrieve all the messages of the buffer split (see Figure 40-14).

Figure 40-14. Chaining Three Mailboxes to Receive a Buffer Split into Four Messages



40.7.3.5 *Transmission Handling*

A mailbox is in Transmit Mode once the MOT field in the CAN_MMRx register has been configured. Message ID and Message Acceptance mask must be set before Receive Mode is enabled.

After Transmit Mode is enabled, the MRDY flag in the CAN_MSR register is automatically set until the first command is sent. When the MRDY flag is set, the software application can prepare a message to be sent by writing to the CAN_MDx registers. The message is sent once the software asks for a transfer command setting the MTCR bit and the message data length in the CAN_MCRx register.

The MRDY flag remains at zero as long as the message has not been sent or aborted. It is important to note that no access to the mailbox data register is allowed while the MRDY flag is cleared. An interrupt is pending for the mailbox while the MRDY flag is set. This interrupt can be masked depending on the mailbox flag in the CAN_IMR global register.

It is also possible to send a remote frame setting the MRTR bit instead of setting the MDLC field. The answer to the remote frame is handled by another reception mailbox. In this case, the device acts as a consumer but with the help of two mailboxes. It is possible to handle the remote frame emission and the answer reception using only one mailbox configured in Consumer Mode. Refer to the section "Remote Frame Handling" on page 753.

Several messages can try to win the bus arbitration in the same time. The message with the highest priority is sent first. Several transfer request commands can be generated at the same time by setting MBx bits in the CAN_TCR register. The priority is set in the PRIOR field of the CAN_MMRx register. Priority 0 is the highest priority, priority 15 is the lowest priority. Thus it is possible to use a part of the message ID to set the PRIOR field. If two mailboxes have the same priority, the message of the mailbox with the lowest number is sent first. Thus if mailbox 0 and mailbox 5 have the same priority and have a message to send at the same time, then the message of the mailbox 0 is sent first.

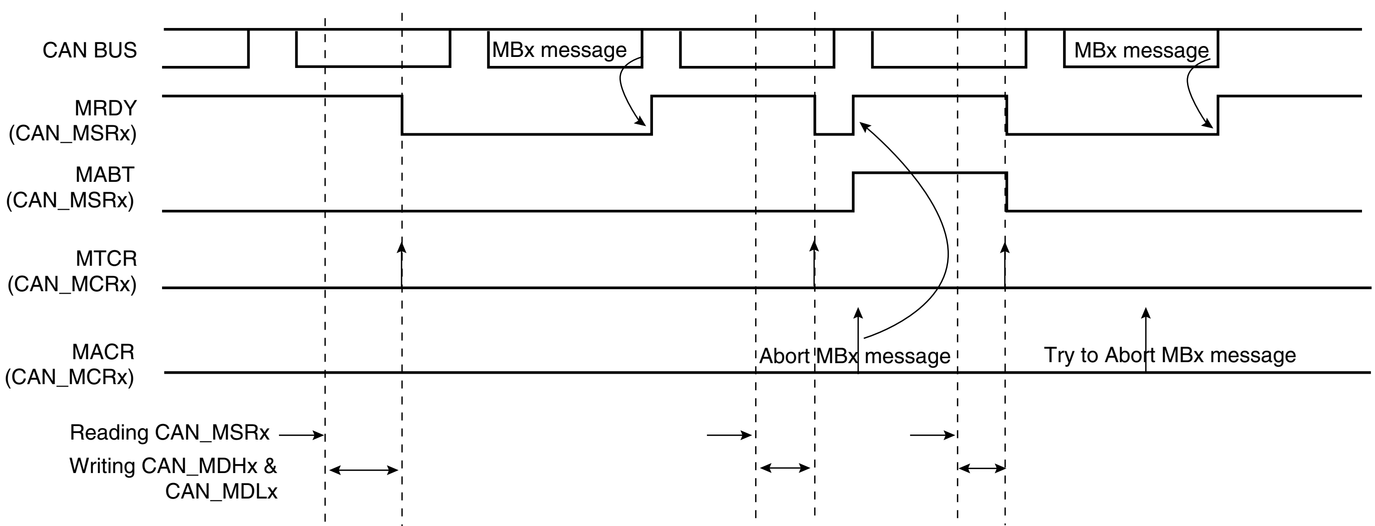
Setting the MACR bit in the CAN_MCRx register aborts the transmission. Transmission for several mailboxes can be aborted by writing MBx fields in the CAN_MACR register. If the message is being sent when the abort command is set, then the application is notified by the MRDY bit set and not the MABT in the CAN_MSRx register. Otherwise, if the message has not been sent, then the MRDY and the MABT are set in the CAN_MSR register.

When the bus arbitration is lost by a mailbox message, the CAN controller tries to win the next bus arbitration with the same message if this one still has the highest priority. Messages to be sent are re-tried automatically until they win the bus arbitration. This feature can be disabled by setting the bit DRPT in the CAN_MR register. In this case if the message was not sent the first time it was transmitted to the CAN transceiver, it is automatically aborted. The MABT flag is set in the CAN_MSRx register until the next transfer command.

Figure 40-15 shows three MBx message attempts being made (MRDY of MBx set to 0).

The first MBx message is sent, the second is aborted and the last one is trying to be aborted but too late because it has already been transmitted to the CAN transceiver.

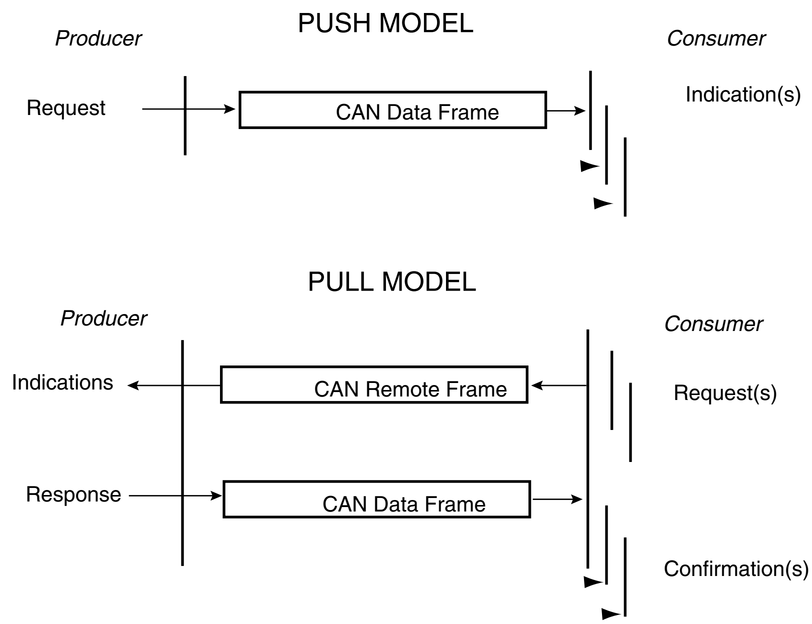
Figure 40-15. Transmitting Messages



40.7.3.6 Remote Frame Handling

Producer/consumer model is an efficient means of handling broadcasted messages. The push model allows a producer to broadcast messages; the pull model allows a customer to ask for messages.

Figure 40-16. Producer / Consumer Model



In Pull Mode, a consumer transmits a remote frame to the producer. When the producer receives a remote frame, it sends the answer accepted by one or many consumers. Using transmit and receive mailboxes, a consumer must dedicate two mailboxes, one in Transmit Mode to send remote frames, and at least one in Receive Mode to capture the producer's answer. The same structure is applicable to a producer: one reception mailbox is required to get the remote frame and one transmit mailbox to answer.

Mailboxes can be configured in Producer or Consumer Mode. A lonely mailbox can handle the remote frame and the answer. With 16 mailboxes, the CAN controller can handle 16 independent producers/consumers.

40.7.3.7 Producer Configuration

A mailbox is in Producer Mode once the MOT field in the CAN_MMRx register has been configured. Message ID and Message Acceptance masks must be set before Receive Mode is enabled.

After Producer Mode is enabled, the MRDY flag in the CAN_MSR register is automatically set until the first transfer command. The software application prepares data to be sent by writing to the CAN_MDHx and the CAN_MDLx registers, then by setting the MTCR bit in the CAN_MCRx register. Data is sent after the reception of a remote frame as soon as it wins the bus arbitration.

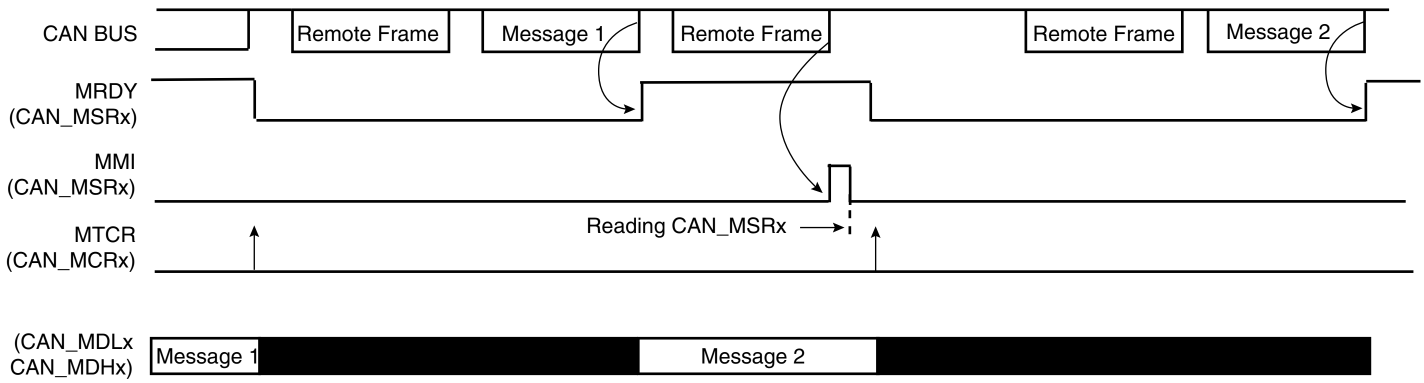
The MRDY flag remains at zero as long as the message has not been sent or aborted. No access to the mailbox data register can be done while MRDY flag is cleared. An interrupt is pending for the mailbox while the MRDY flag is set. This interrupt can be masked according to the mailbox flag in the CAN_IMR global register.

If a remote frame is received while no data are ready to be sent (signal MRDY set in the CAN_MSRx register), then the MMI signal is set in the CAN_MSRx register. This bit is cleared by reading the CAN_MSRx register.

The MRTR field in the CAN_MSRx register has no meaning. This field is used only when using Receive and Receive with Overwrite modes.

After a remote frame has been received, the mailbox functions like a transmit mailbox. The message with the highest priority is sent first. The transmitted message may be aborted by setting the MACR bit in the CAN_MCR register. Please refer to the section "Transmission Handling" on page 752.

Figure 40-17. Producer Handling



40.7.3.8 Consumer Configuration

A mailbox is in Consumer Mode once the MOT field in the CAN_MMRx register has been configured. Message ID and Message Acceptance masks must be set before Receive Mode is enabled.

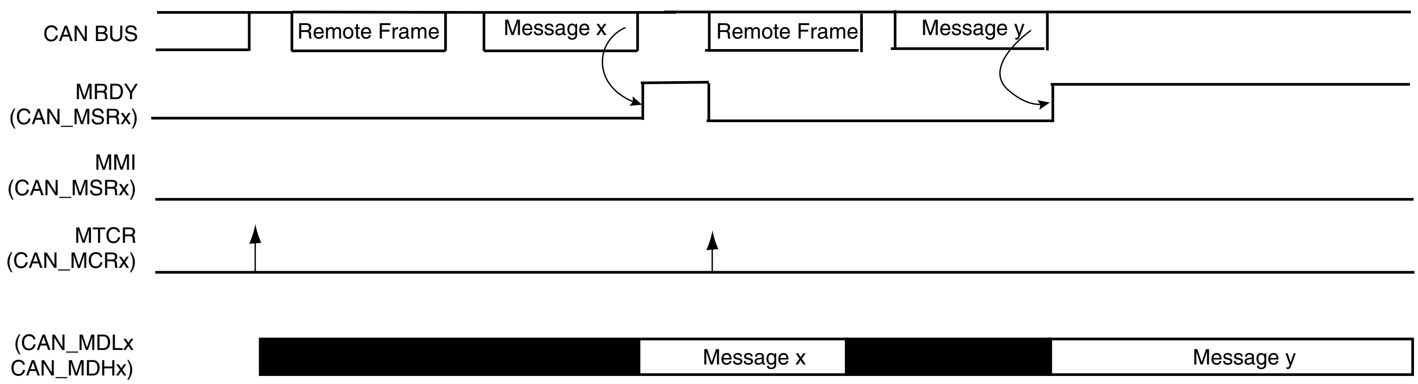
After Consumer Mode is enabled, the MRDY flag in the CAN_MSR register is automatically cleared until the first transfer request command. The software application sends a remote frame by setting the MTCR bit in the CAN_MCRx register or the MBx bit in the global CAN_TCR register. The application is notified of the answer by the MRDY flag set in the CAN_MSRx register. The application can read the data contents in the CAN_MDHx and CAN_MDLx registers. An interrupt is pending for the mailbox while the MRDY flag is set. This interrupt can be masked according to the mailbox flag in the CAN_IMR global register.

The MRTR bit in the CAN_MCRx register has no effect. This field is used only when using Transmit Mode.

After a remote frame has been sent, the consumer mailbox functions as a reception mailbox. The first message received is stored in the mailbox data registers. If other messages intended for this mailbox have been sent while the MRDY flag is set in the CAN_MSRx register, they will be lost. The application is notified by reading the MMI field in the CAN_MSRx register. The read operation automatically clears the MMI flag.

If several messages are answered by the Producer, the CAN controller may have one mailbox in consumer configuration, zero or several mailboxes in Receive Mode and one mailbox in Receive with Overwrite Mode. In this case, the consumer mailbox must have a lower number than the Receive with Overwrite mailbox. The transfer command can be triggered for all mailboxes at the same time by setting several MBx fields in the CAN_TCR register.

Figure 40-18. Consumer Handling



40.7.4 CAN Controller Timing Modes

Using the free running 16-bit internal timer, the CAN controller can be set in one of the two following timing modes:

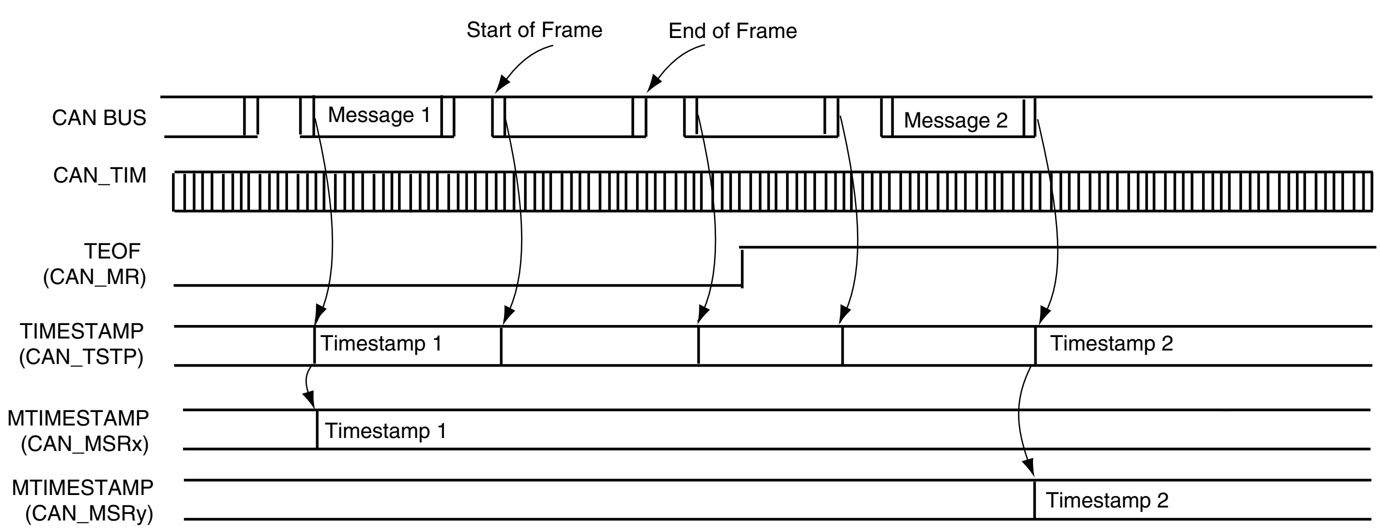
- Timestamping Mode: The value of the internal timer is captured at each Start Of Frame or each End Of Frame.
- Time Triggered Mode: The mailbox transfer operation is triggered when the internal timer reaches the mailbox trigger.

Timestamping Mode is enabled by clearing the TTM bit in the CAN_MR register. Time Triggered Mode is enabled by setting the TTM bit in the CAN_MR register.

40.7.4.1 Timestamping Mode

Each mailbox has its own timestamp value. Each time a message is sent or received by a mailbox, the 16-bit value MTIMESTAMP of the CAN_TIMESTP register is transferred to the LSB bits of the CAN_MSRx register. The value read in the CAN_MSRx register corresponds to the internal timer value at the Start Of Frame or the End Of Frame of the message handled by the mailbox.

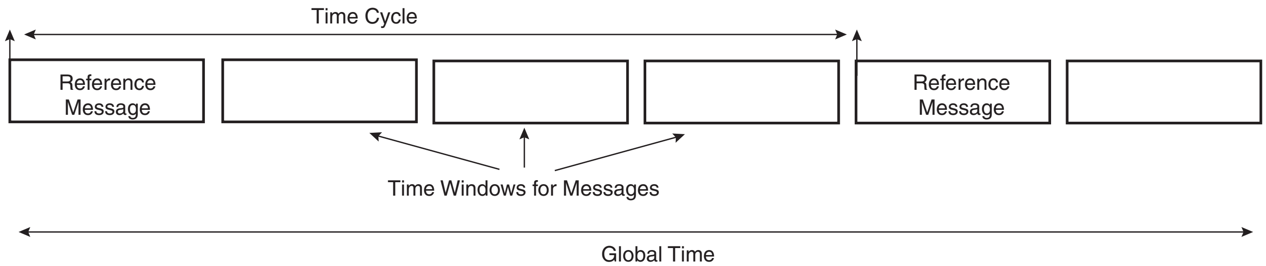
Figure 40-19. Mailbox Timestamp



40.7.4.2 Time Triggered Mode

In Time Triggered Mode, basic cycles can be split into several time windows. A basic cycle starts with a reference message. Each time a window is defined from the reference message, a transmit operation should occur within a pre-defined time window. A mailbox must not win the arbitration in a previous time window, and it must not be retried if the arbitration is lost in the time window.

Figure 40-20. Time Triggered Principle



Time Trigger Mode is enabled by setting the TTM field in the CAN_MR register. In Time Triggered Mode, as in Timestamp Mode, the CAN_TIMESTP field captures the values of the internal counter, but the MTIMESTAMP fields in the CAN_MSRx registers are not active and are read at 0.

40.7.4.3 Synchronization by a Reference Message

In Time Triggered Mode, the internal timer counter is automatically reset when a new message is received in the last mailbox. This reset occurs after the reception of the End Of Frame on the rising edge of the MRDY signal in the CAN_MSRx register. This allows synchronization of the internal timer counter with the reception of a reference message and the start a new time window.

40.7.4.4 Transmitting within a Time Window

A time mark is defined for each mailbox. It is defined in the 16-bit MTIMEMARK field of the CAN_MMRx register. At each internal timer clock cycle, the value of the CAN_TIM is compared with each mailbox time mark. When the internal timer counter reaches the MTIMEMARK value, an internal timer event for the mailbox is generated for the mailbox.

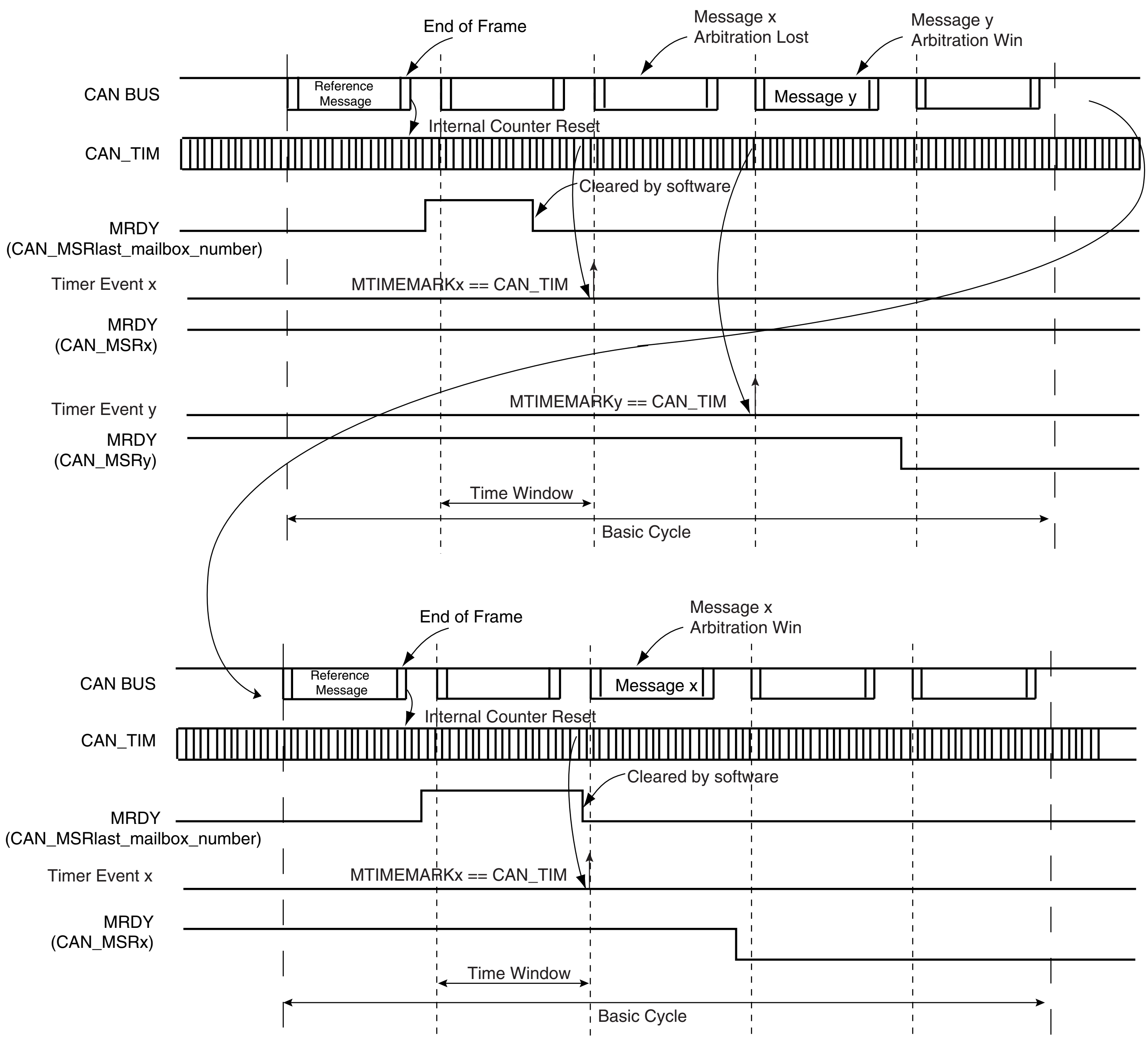
In Time Triggered Mode, transmit operations are delayed until the internal timer event for the mailbox. The application prepares a message to be sent by setting the MTCR in the CAN_MCRx register. The message is not sent until the CAN_TIM value is less than the MTIMEMARK value defined in the CAN_MMRx register.

If the transmit operation is failed, i.e., the message loses the bus arbitration and the next transmit attempt is delayed until the next internal time trigger event. This prevents overlapping the next time window, but the message is still pending and is retried in the next time window when CAN_TIM value equals the MTIMEMARK value. It is also possible to prevent a retry by setting the DRPT field in the CAN_MR register.

40.7.4.5 Freezing the Internal Timer Counter

The internal counter can be frozen by setting TIMFRZ in the CAN_MR register. This prevents an unexpected roll-over when the counter reaches FFFFh. When this occurs, it automatically freezes until a new reset is issued, either due to a message received in the last mailbox or any other reset counter operations. The TOVF bit in the CAN_SR register is set when the counter is frozen. The TOVF bit in the CAN_SR register is cleared by reading the CAN_SR register. Depending on the corresponding interrupt mask in the CAN_IMR register, an interrupt is generated when TOVF is set.

Figure 40-21. Time Triggered Operations



40.8 Controller Area Network (CAN) User Interface

Table 40-4. Register Mapping

Offset	Register	Name	Access	Reset
0x0000	Mode Register	CAN_MR	Read-write	0x0
0x0004	Interrupt Enable Register	CAN_IER	Write-only	-
0x0008	Interrupt Disable Register	CAN_IDR	Write-only	-
0x000C	Interrupt Mask Register	CAN_IMR	Read-only	0x0
0x0010	Status Register	CAN_SR	Read-only	0x0
0x0014	Baudrate Register	CAN_BR	Read-write	0x0
0x0018	Timer Register	CAN_TIM	Read-only	0x0
0x001C	Timestamp Register	CAN_TIMESTP	Read-only	0x0
0x0020	Error Counter Register	CAN_ECR	Read-only	0x0
0x0024	Transfer Command Register	CAN_TCR	Write-only	-
0x0028	Abort Command Register	CAN_ACR	Write-only	-
0x0100 - 0x01FC	Reserved	–	–	–
0x0200 + mb_num * 0x20 + 0x00	Mailbox Mode Register(2)	CAN_MMR	Read-write	0x0
0x0200 + mb_num * 0x20 + 0x04	Mailbox Acceptance Mask Register	CAN_MAM	Read-write	0x0
0x0200 + mb_num * 0x20 + 0x08	Mailbox ID Register	CAN_MID	Read-write	0x0
0x0200 + mb_num * 0x20 + 0x0C	Mailbox Family ID Register	CAN_MFID	Read-only	0x0
0x0200 + mb_num * 0x20 + 0x10	Mailbox Status Register	CAN_MSR	Read-only	0x0
0x0200 + mb_num * 0x20 + 0x14	Mailbox Data Low Register	CAN_MDL	Read-write	0x0
0x0200 + mb_num * 0x20 + 0x18	Mailbox Data High Register	CAN_MDH	Read-write	0x0
0x0200 + mb_num * 0x20 + 0x1C	Mailbox Control Register	CAN_MCR	Write-only	-

2. Mailbox number ranges from 0 to 15.

40.8.1 CAN Mode Register

Name: CAN_MR

Address: 0xFFFAC000

Access: 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
DRPT	TIMFRZ	TTM	TEOF	OVL	ABM	LPM	CANEN

- **CANEN: CAN Controller Enable**

0 = The CAN Controller is disabled.

1 = The CAN Controller is enabled.

- **LPM: Disable/Enable Low Power Mode**

w Power Mode.

1 = Enable Low Power M

CAN controller enters Low Power Mode once all pending messages have been transmitted.

- **ABM: Disable/Enable Autobaud/Listen mode**

0 = Disable Autobaud/listen mode.

1 = Enable Autobaud/listen mode.

- **OVL: Disable/Enable Overload Frame**

0 = No overload frame is generated.

1 = An overload frame is generated after each successful reception for mailboxes configured in Receive with/without overwrite Mode, Producer and Consumer.

- **TEOF: Timestamp messages at each end of Frame**

0 = The value of CAN_TIM is captured in the CAN_TIMESTP register at each Start Of Frame.

1 = The value of CAN_TIM is captured in the CAN_TIMESTP register at each End Of Frame.

- **TTM: Disable/Enable Time Triggered Mode**

0 = Time Triggered Mode is disabled.

1 = Time Triggered Mode is enabled.

- **TIMFRZ: Enable Timer Freeze**

0 = The internal timer continues to be incremented after it reached 0xFFFF.

1 = The internal timer stops incrementing after reaching 0xFFFF. It is restarted after a timer reset. See "Freezing the Internal Timer Counter" on page 757.

- **DRPT: Disable Repeat**

0 = When a transmit mailbox loses the bus arbitration, the transfer request remains pending.

1 = When a transmit mailbox lose the bus arbitration, the transfer request is automatically aborted. It automatically raises the MABT and MRDT flags in the corresponding CAN_MSRx.

40.8.2 CAN Interrupt Enable Register

Name: CAN_IER

Address: 0xFFFAC004

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	BERR	FERR	AERR	SERR	CERR

23	22	21	20	19	18	17	16
TSTP	TOVF	WAKEUP	SLEEP	BOFF	ERRP	WARN	ERRA

15	14	13	12	11	10	9	8
MB15	MB14	MB13	MB12	MB11	MB10	MB9	MB8

7	6	5	4	3	2	1	0
MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0

- **MBx: Mailbox x Interrupt Enable**

0 = No effect.

1 = Enable Mailbox x interrupt.

- **ERRA: Error Active Mode Interrupt Enable**

0 = No effect.

1 = Enable ERRA interrupt.

- **WARN: Warning Limit Interrupt Enable**

0 = No effect.

1 = Enable WARN interrupt.

- **ERRP: Error Passive Mode Interrupt Enable**

0 = No effect.

1 = Enable ERRP interrupt.

- **BOFF: Bus Off Mode Interrupt Enable**

0 = No effect.

1 = Enable BOFF interrupt.

- **SLEEP: Sleep Interrupt Enable**

0 = No effect.

1 = Enable SLEEP interrupt.

- **WAKEUP: Wakeup Interrupt Enable**

0 = No effect.

1 = Enable SLEEP interrupt.

- **TOVF: Timer Overflow Interrupt Enable**

0 = No effect.

1 = Enable TOVF interrupt.

- **TSTP: TimeStamp Interrupt Enable**

0 = No effect.

1 = Enable TSTP interrupt.

- **CERR: CRC Error Interrupt Enable**

0 = No effect.

1 = Enable CRC Error interrupt.

- **SERR: Stuffing Error Interrupt Enable**

0 = No effect.

1 = Enable Stuffing Error interrupt.

- **AERR: Acknowledgment Error Interrupt Enable**

0 = No effect.

1 = Enable Acknowledgment Error interrupt.

- **FERR: Form Error Interrupt Enable**

0 = No effect.

1 = Enable Form Error interrupt.

- **BERR: Bit Error Interrupt Enable**

0 = No effect.

1 = Enable Bit Error interrupt.

40.8.3 CAN Interrupt Disable Register

Name: CAN_IDR

Address: 0xFFFAC008

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	BERR	FERR	AERR	SERR	CERR

23	22	21	20	19	18	17	16
TSTP	TOVF	WAKEUP	SLEEP	BOFF	ERRP	WARN	ERRA

15	14	13	12	11	10	9	8
MB15	MB14	MB13	MB12	MB11	MB10	MB9	MB8

7	6	5	4	3	2	1	0
MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0

- **MBx: Mailbox x Interrupt Disable**

0 = No effect.

1 = Disable Mailbox x interrupt.

- **ERRA: Error Active Mode Interrupt Disable**

0 = No effect.

1 = Disable ERRA interrupt.

- **WARN: Warning Limit Interrupt Disable**

0 = No effect.

1 = Disable WARN interrupt.

- **ERRP: Error Passive Mode Interrupt Disable**

0 = No effect.

1 = Disable ERRP interrupt.

- **BOFF: Bus Off Mode Interrupt Disable**

0 = No effect.

1 = Disable BOFF interrupt.

- **SLEEP: Sleep Interrupt Disable**

0 = No effect.

1 = Disable SLEEP interrupt.

- **WAKEUP: Wakeup Interrupt Disable**

0 = No effect.

1 = Disable WAKEUP interrupt.

- **TOVF: Timer Overflow Interrupt**

0 = No effect.

1 = Disable TOVF interrupt.

- **TSTP: TimeStamp Interrupt Disable**

0 = No effect.

1 = Disable TSTP interrupt.

- **CERR: CRC Error Interrupt Disable**

0 = No effect.

1 = Disable CRC Error interrupt.

- **SERR: Stuffing Error Interrupt Disable**

0 = No effect.

1 = Disable Stuffing Error interrupt.

- **AERR: Acknowledgment Error Interrupt Disable**

0 = No effect.

1 = Disable Acknowledgment Error interrupt.

- **FERR: Form Error Interrupt Disable**

0 = No effect.

1 = Disable Form Error interrupt.

- **BERR: Bit Error Interrupt Disable**

0 = No effect.

1 = Disable Bit Error interrupt.

40.8.4 CAN Interrupt Mask Register

Name: CAN_IMR

Address: 0xFFFAC00C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	BERR	FERR	AERR	SERR	CERR

23	22	21	20	19	18	17	16
TSTP	TOVF	WAKEUP	SLEEP	BOFF	ERRP	WARN	ERRA

15	14	13	12	11	10	9	8
MB15	MB14	MB13	MB12	MB11	MB10	MB9	MB8

7	6	5	4	3	2	1	0
MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0

- **MBx: Mailbox x Interrupt Mask**

0 = Mailbox x interrupt is disabled.

1 = Mailbox x interrupt is enabled.

- **ERRA: Error Active Mode Interrupt Mask**

0 = ERRA interrupt is disabled.

1 = ERRA interrupt is enabled.

- **WARN: Warning Limit Interrupt Mask**

0 = Warning Limit interrupt is disabled.

1 = Warning Limit interrupt is enabled.

- **ERRP: Error Passive Mode Interrupt Mask**

0 = ERRP interrupt is disabled.

1 = ERRP interrupt is enabled.

- **BOFF: Bus Off Mode Interrupt Mask**

0 = BOFF interrupt is disabled.

1 = BOFF interrupt is enabled.

- **SLEEP: Sleep Interrupt Mask**

0 = SLEEP interrupt is disabled.

1 = SLEEP interrupt is enabled.

- **WAKEUP: Wakeup Interrupt Mask**

0 = WAKEUP interrupt is disabled.

1 = WAKEUP interrupt is enabled.

- **TOVF: Timer Overflow Interrupt Mask**

0 = TOVF interrupt is disabled.

1 = TOVF interrupt is enabled.

- **TSTP: Timestamp Interrupt Mask**

0 = TSTP interrupt is disabled.

1 = TSTP interrupt is enabled.

- **CERR: CRC Error Interrupt Mask**

0 = CRC Error interrupt is disabled.

1 = CRC Error interrupt is enabled.

- **SERR: Stuffing Error Interrupt Mask**

0 = Bit Stuffing Error interrupt is disabled.

1 = Bit Stuffing Error interrupt is enabled.

- **AERR: Acknowledgment Error Interrupt Mask**

0 = Acknowledgment Error interrupt is disabled.

1 = Acknowledgment Error interrupt is enabled.

- **FERR: Form Error Interrupt Mask**

0 = Form Error interrupt is disabled.

1 = Form Error interrupt is enabled.

- **BERR: Bit Error Interrupt Mask**

0 = Bit Error interrupt is disabled.

1 = Bit Error interrupt is enabled.

40.8.5 CAN Status Register

Name: CAN_SR

Address: 0xFFFAC010

Access: Read-only

31	30	29	28	27	26	25	24
OVLSY	TBSY	RBSY	BERR	FERR	AERR	SERR	CERR

23	22	21	20	19	18	17	16
TSTP	TOVF	WAKEUP	SLEEP	BOFF	ERRP	WARN	ERRA

15	14	13	12	11	10	9	8
MB15	MB14	MB13	MB12	MB11	MB10	MB9	MB8

7	6	5	4	3	2	1	0
MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0

- **MBx: Mailbox x Event**

0 = No event occurred on Mailbox x.

1 = An event occurred on Mailbox x.

An event corresponds to MRDY, MABT fields in the CAN_MSRx register.

- **ERRA: Error Active Mode**

0 = CAN controller is not in Error Active Mode.

1 = CAN controller is in Error Active Mode.

This flag is set depending on TEC and REC counter values. It is set when node is neither in Error Passive Mode nor in Bus Off Mode.

This flag is automatically reset when above condition is not satisfied. Refer to Section 40.6.4.6 “Error Interrupt Handler” on page 743 for more information.

- **WARN: Warning Limit**

0 = CAN controller Warning Limit is not reached.

1 = CAN controller Warning Limit is reached.

This flag is set depending on TEC and REC counter values. It is set when at least one of the counter values exceeds 96.

This flag is automatically reset when above condition is not satisfied. Refer to Section 40.6.4.6 “Error Interrupt Handler” on page 743 for more information.

- **ERRP: Error Passive Mode**

0 = CAN controller is not in Error Passive Mode.

1 = CAN controller is in Error Passive Mode.

This flag is set depending on TEC and REC counters values.

A node is error passive when TEC counter is greater or equal to 128 (decimal) or when the REC counter is greater or equal to 128 (decimal).

This flag is automatically reset when above condition is not satisfied. Refer to Section 40.6.4.6 "Error Interrupt Handler" on page 743 for more information.

- **BOFF: Bus Off Mode**

0 = CAN controller is not in Bus Off Mode.

1 = CAN controller is in Bus Off Mode.

This flag is set depending on TEC counter value. A node is bus off when TEC counter is greater or equal to 256 (decimal).

This flag is automatically reset when above condition is not satisfied. Refer to Section 40.6.4.6 "Error Interrupt Handler" on page 743 for more information.

- **SLEEP: CAN controller in Low power Mode**

0 = CAN controller is not in low power mode.

1 = CAN controller is in low power mode.

This flag is automatically reset when Low power mode is disabled

- **WAKEUP: CAN controller is not in Low power Mode**

0 = CAN controller is in low power mode.

1 = CAN controller is not in low power mode.

When a WAKEUP event occurs, the CAN controller is synchronized with the bus activity. Messages can be transmitted or received. The CAN controller clock must be available when a WAKEUP event occurs. This flag is automatically reset when the CAN Controller enters Low Power mode.

- **TOVF: Timer Overflow**

0 = The timer has not rolled-over FFFFh to 0000h.

1 = The timer rolls-over FFFFh to 0000h.

This flag is automatically cleared by reading CAN_SR register.

- **TSTP Timestamp**

0 = No bus activity has been detected.

1 = A start of frame or an end of frame has been detected (according to the TEOF field in the CAN_MR register).

This flag is automatically cleared by reading the CAN_SR register.

- **CERR: Mailbox CRC Error**

0 = No CRC error occurred during a previous transfer.

1 = A CRC error occurred during a previous transfer.

A CRC error has been detected during last reception.

This flag is automatically cleared by reading CAN_SR register.

- **SERR: Mailbox Stuffing Error**

0 = No stuffing error occurred during a previous transfer.

1 = A stuffing error occurred during a previous transfer.

A form error results from the detection of more than five consecutive bits with the same polarity.

This flag is automatically cleared by reading CAN_SR register.

- **AERR: Acknowledgment Error**

0 = No acknowledgment error occurred during a previous transfer.

1 = An acknowledgment error occurred during a previous transfer.

An acknowledgment error is detected when no detection of the dominant bit in the acknowledge slot occurs.

This flag is automatically cleared by reading CAN_SR register.

- **FERR: Form Error**

0 = No form error occurred during a previous transfer

1 = A form error occurred during a previous transfer

A form error results from violations on one or more of the fixed form of the following bit fields:

   – CRC delimiter
   – ACK delimiter
   – End of frame
   – Error delimiter
   – Overload delimiter

This flag is automatically cleared by reading CAN_SR register.

- **BERR: Bit Error**

0 = No bit error occurred during a previous transfer.

1 = A bit error occurred during a previous transfer.

A bit error is set when the bit value monitored on the line is different from the bit value sent.

This flag is automatically cleared by reading CAN_SR register.

- **RBSY: Receiver busy**

0 = CAN receiver is not receiving a frame.

1 = CAN receiver is receiving a frame.

Receiver busy. This status bit is set by hardware while CAN receiver is acquiring or monitoring a frame (remote, data, overload or error frame). It is automatically reset when CAN is not receiving.

- **TBSY: Transmitter busy**

0 = CAN transmitter is not transmitting a frame.

1 = CAN transmitter is transmitting a frame.

Transmitter busy. This status bit is set by hardware while CAN transmitter is generating a frame (remote, data, overload or error frame). It is automatically reset when CAN is not transmitting.

- **OVLSY: Overload busy**

0 = CAN transmitter is not transmitting an overload frame.

1 = CAN transmitter is transmitting a overload frame.

It is automatically reset when the bus is not transmitting an overload frame.

40.8.6 CAN Baudrate Register

Name: CAN_BR

Address: 0xFFFAC014

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	SMP
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	BRP						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	SJW		–	PROPAG		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	PHASE1			–	PHASE2		

Any modification on one of the fields of the CANBR register must be done while CAN module is disabled.

To compute the different Bit Timings, please refer to the Section 40.6.4.1 “CAN Bit Timing Configuration” on page 738.

- **PHASE2: Phase 2 segment**

This phase is used to compensate the edge phase error.

$$t_{PHS2} = t_{CSC} \times (PHASE2 + 1)$$

Warning: PHASE2 value must be different from 0.

- **PHASE1: Phase 1 segment**

This phase is used to compensate for edge phase error.

$$t_{PHS1} = t_{CSC} \times (PHASE1 + 1)$$

- **PROPAG: Programming time segment**

This part of the bit time is used to compensate for the physical delay times within the network.

$$t_{PRS} = t_{CSC} \times (PROPAG + 1)$$

- **SJW: Re-synchronization jump width**

To compensate for phase shifts between clock oscillators of different controllers on bus. The controller must re-synchronize on any relevant signal edge of the current transmission. The synchronization jump width defines the maximum of clock cycles a bit period may be shortened or lengthened by re-synchronization.

$$t_{SJW} = t_{CSC} \times (SJW + 1)$$

- **BRP: Baudrate Prescaler.**

This field allows user to program the period of the CAN system clock to determine the individual bit timing.

$$t_{CSC} = (BRP + 1) / MCK$$

The BRP field must be within the range [1, 0x7F], i.e., BRP = 0 is not authorized.

- **SMP: Sampling Mode**

0 = The incoming bit stream is sampled once at sample point.

1 = The incoming bit stream is sampled three times with a period of a MCK clock period, centered on sample point.

SMP Sampling Mode is automatically disabled if BRP = 0.

40.8.7 CAN Timer Register

Name: CAN_TIM

Address: 0xFFFAC018

Access: Read-only

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

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

15	14	13	12	11	10	9	8
TIMER15	TIMER14	TIMER13	TIMER12	TIMER11	TIMER10	TIMER9	TIMER8

7	6	5	4	3	2	1	0
TIMER7	TIMER6	TIMER5	TIMER4	TIMER3	TIMER2	TIMER1	TIMER0

- **TIMERx: Timer**

This field represents the internal CAN controller 16-bit timer value.

40.8.8 CAN Timestamp Register

Name: CAN_TIMESTP

Address: 0xFFFAC01C

Access: Read-only

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

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

15	14	13	12	11	10	9	8
MTIMESTAMP15	MTIMESTAMP14	MTIMESTAMP13	MTIMESTAMP12	MTIMESTAMP11	MTIMESTAMP10	MTIMESTAMP9	MTIMESTAMP8

7	6	5	4	3	2	1	0
MTIMESTAMP7	MTIMESTAMP6	MTIMESTAMP5	MTIMESTAMP4	MTIMESTAMP3	MTIMESTAMP2	MTIMESTAMP1	MTIMESTAMP0

- **MTIMESTAMPx: Timestamp**

This field represents the internal CAN controller 16-bit timer value.

If the TEOF bit is cleared in the CAN_MR register, the internal Timer Counter value is captured in the MTIMESTAMP field at each start of frame. Else the value is captured at each end of frame. When the value is captured, the TSTP flag is set in the CAN_SR register. If the TSTP mask in the CAN_IMR register is set, an interrupt is generated while TSTP flag is set in the CAN_SR register. This flag is cleared by reading the CAN_SR register.

Note: The CAN_TIMESTP register is reset when the CAN is disabled then enabled thanks to the CANEN bit in the CAN_MR.

40.8.9 CAN Error Counter Register

Name: CAN_ECR

Address: 0xFFFAC020

Access: Read-only

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

23	22	21	20	19	18	17	16
TEC							

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
REC							

- **REC: Receive Error Counter**

When a receiver detects an error, REC will be increased by one, except when the detected error is a BIT ERROR while sending an ACTIVE ERROR FLAG or an OVERLOAD FLAG.

When a receiver detects a dominant bit as the first bit after sending an ERROR FLAG, REC is increased by 8.

When a receiver detects a BIT ERROR while sending an ACTIVE ERROR FLAG, REC is increased by 8.

Any node tolerates up to 7 consecutive dominant bits after sending an ACTIVE ERROR FLAG, PASSIVE ERROR FLAG or OVERLOAD FLAG. After detecting the 14th consecutive dominant bit (in case of an ACTIVE ERROR FLAG or an OVERLOAD FLAG) or after detecting the 8th consecutive dominant bit following a PASSIVE ERROR FLAG, and after each sequence of additional eight consecutive dominant bits, each receiver increases its REC by 8.

After successful reception of a message, REC is decreased by 1 if it was between 1 and 127. If REC was 0, it stays 0, and if it was greater than 127, then it is set to a value between 119 and 127.

- **TEC: Transmit Error Counter**

When a transmitter sends an ERROR FLAG, TEC is increased by 8 except when

– the transmitter is error passive and detects an ACKNOWLEDGMENT ERROR because of not detecting a dominant ACK and does not detect a dominant bit while sending its PASSIVE ERROR FLAG.

– the transmitter sends an ERROR FLAG because a STUFF ERROR occurred during arbitration and should have been recessive and has been sent as recessive but monitored as dominant.

When a transmitter detects a BIT ERROR while sending an ACTIVE ERROR FLAG or an OVERLOAD FLAG, the TEC will be increased by 8.

Any node tolerates up to 7 consecutive dominant bits after sending an ACTIVE ERROR FLAG, PASSIVE ERROR FLAG or OVERLOAD FLAG. After detecting the 14th consecutive dominant bit (in case of an ACTIVE ERROR FLAG or an OVERLOAD FLAG) or after detecting the 8th consecutive dominant bit following a PASSIVE ERROR FLAG, and after each sequence of additional eight consecutive dominant bits every transmitter increases its TEC by 8.

After a successful transmission the TEC is decreased by 1 unless it was already 0.

40.8.10 CAN Transfer Command Register

Name: CAN_TCR

Address: 0xFFFAC024

Access: Write-only

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

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

15	14	13	12	11	10	9	8
MB15	MB14	MB13	MB12	MB11	MB10	MB9	MB8

7	6	5	4	3	2	1	0
MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0

This register initializes several transfer requests at the same time.

- **MBx: Transfer Request for Mailbox x**

Mailbox Object Type	Description
Receive	It receives the next message.
Receive with overwrite	This triggers a new reception.
Transmit	Sends data prepared in the mailbox as soon as possible.
Consumer	Sends a remote frame.
Producer	Sends data prepared in the mailbox after receiving a remote frame from a consumer.

This flag clears the MRDY and MABT flags in the corresponding CAN_MSRx register.

When several mailboxes are requested to be transmitted simultaneously, they are transmitted in turn, starting with the mailbox with the highest priority. If several mailboxes have the same priority, then the mailbox with the lowest number is sent first (i.e., MB0 will be transferred before MB1).

- **TIMRST: Timer Reset**

Resets the internal timer counter. If the internal timer counter is frozen, this command automatically re-enables it. This command is useful in Time Triggered mode.

40.8.11 CAN Abort Command Register

Name: CAN_ACR

Address: 0xFFFAC028

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
MB15	MB14	MB13	MB12	MB11	MB10	MB9	MB8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0

This register initializes several abort requests at the same time.

- **MBx: Abort Request for Mailbox x**

Mailbox Object Type	Description
Receive	No action
Receive with overwrite	No action
Transmit	Cancels transfer request if the message has not been transmitted to the CAN transceiver.
Consumer	Cancels the current transfer before the remote frame has been sent.
Producer	Cancels the current transfer. The next remote frame is not serviced.

It is possible to set MACR field (in the CAN_MCRx register) for each mailbox.

40.8.12 CAN Message Mode Register

Name: CAN_MMRx [x=0..15]

Addresses: 0xFFFAC200 [0], 0xFFFAC220 [1], 0xFFFAC240 [2], 0xFFFAC260 [3], 0xFFFAC280 [4], 0xFFFAC2A0 [5], 0xFFFAC2C0 [6], 0xFFFAC2E0 [7], 0xFFFAC300 [8], 0xFFFAC320 [9], 0xFFFAC340 [10], 0xFFFAC360 [11], 0xFFFAC380 [12], 0xFFFAC3A0 [13], 0xFFFAC3C0 [14], 0xFFFAC3E0 [15]

Access: Read-write

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

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

15	14	13	12	11	10	9	8
MTIMEMARK 15	MTIMEMARK 14	MTIMEMARK 13	MTIMEMARK 12	MTIMEMARK 11	MTIMEMARK 10	MTIMEMARK 9	MTIMEMARK 8

7	6	5	4	3	2	1	0
MTIMEMARK 7	MTIMEMARK 6	MTIMEMARK 5	MTIMEMARK 4	MTIMEMARK 3	MTIMEMARK 2	MTIMEMARK 1	MTIMEMARK 0

- **MTIMEMARKx: Mailbox Timemark**

This field is active in Time Triggered Mode. Transmit operations are allowed when the internal timer counter reaches the Mailbox Timemark. See “Transmitting within a Time Window” on page 757.

In Timestamp Mode, MTIMEMARK is set to 0.

- **PRIOR: Mailbox Priority**

This field has no effect in receive and receive with overwrite modes. In these modes, the mailbox with the lowest number is serviced first.

When several mailboxes try to transmit a message at the same time, the mailbox with the highest priority is serviced first. If several mailboxes have the same priority, the mailbox with the lowest number is serviced first (i.e., MBx0 is serviced before MBx 15 if they have the same priority).

- **MOT: Mailbox Object Type**

This field allows the user to define the type of the mailbox. All mailboxes are independently configurable. Five different types are possible for each mailbox:

MOT			Mailbox Object Type
0	0	0	Mailbox is disabled. This prevents receiving or transmitting any messages with this mailbox.
0	0	1	Reception Mailbox. Mailbox is configured for reception. If a message is received while the mailbox data register is full, it is discarded.
0	1	0	Reception mailbox with overwrite. Mailbox is configured for reception. If a message is received while the mailbox is full, it overwrites the previous message.
0	1	1	Transmit mailbox. Mailbox is configured for transmission.

1	0	0	Consumer Mailbox. Mailbox is configured in reception but behaves as a Transmit Mailbox, i.e., it sends a remote frame and waits for an answer.
1	0	1	Producer Mailbox. Mailbox is configured in transmission but also behaves like a reception mailbox, i.e., it waits to receive a Remote Frame before sending its contents.
1	1	X	Reserved

40.8.13 CAN Message Acceptance Mask Register

Name: CAN_MAMx [x=0..15]

Addresses: 0xFFFAC204 [0], 0xFFFAC224 [1], 0xFFFAC244 [2], 0xFFFAC264 [3], 0xFFFAC284 [4], 0xFFFAC2A4 [5], 0xFFFAC2C4 [6], 0xFFFAC2E4 [7], 0xFFFAC304 [8], 0xFFFAC324 [9], 0xFFFAC344 [10], 0xFFFAC364 [11], 0xFFFAC384 [12], 0xFFFAC3A4 [13], 0xFFFAC3C4 [14], 0xFFFAC3E4 [15]

Access: Read-write

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

23	22	21	20	19	18	17	16
MIDvA						MIDvB	

15	14	13	12	11	10	9	8
MIDvB							

7	6	5	4	3	2	1	0
MIDvB							

To prevent concurrent access with the internal CAN core, the application must disable the mailbox before writing to CAN_MAMx registers.

- **MIDvB: Complementary bits for identifier in extended frame mode**

Acceptance mask for corresponding field of the message IDvB register of the mailbox.

- **MIDvA: Identifier for standard frame mode**

Acceptance mask for corresponding field of the message IDvA register of the mailbox.

- **MIDE: Identifier Version**

0= Compares IDvA from the received frame with the CAN_MIDx register masked with CAN_MAMx register.

1= Compares IDvA and IDvB from the received frame with the CAN_MIDx register masked with CAN_MAMx register.

40.8.14 CAN Message ID Register

Name: CAN_MIDx [x=0..15]

Addresses: 0xFFFAC208 [0], 0xFFFAC228 [1], 0xFFFAC248 [2], 0xFFFAC268 [3], 0xFFFAC288 [4], 0xFFFAC2A8 [5], 0xFFFAC2C8 [6], 0xFFFAC2E8 [7], 0xFFFAC308 [8], 0xFFFAC328 [9], 0xFFFAC348 [10], 0xFFFAC368 [11], 0xFFFAC388 [12], 0xFFFAC3A8 [13], 0xFFFAC3C8 [14], 0xFFFAC3E8 [15]

Access: Read-write

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

23	22	21	20	19	18	17	16
MIDvA						MIDvB	

15	14	13	12	11	10	9	8
MIDvB							

7	6	5	4	3	2	1	0
MIDvB							

To prevent concurrent access with the internal CAN core, the application must disable the mailbox before writing to CAN_MIDx registers.

- **MIDvB: Complementary bits for identifier in extended frame mode**

If MIDE is cleared, MIDvB value is 0.

- **MIDE: Identifier Version**

This bit allows the user to define the version of messages processed by the mailbox. If set, mailbox is dealing with version 2.0 Part B messages; otherwise, mailbox is dealing with version 2.0 Part A messages.

- **MIDvA: Identifier for standard frame mode**

40.8.15 CAN Message Family ID Register

Name: CAN_MFIDx [x=0..15]

Addresses: 0xFFFAC20C [0], 0xFFFAC22C [1], 0xFFFAC24C [2], 0xFFFAC26C [3], 0xFFFAC28C [4], 0xFFFAC2AC [5], 0xFFFAC2CC [6], 0xFFFAC2EC [7], 0xFFFAC30C [8], 0xFFFAC32C [9], 0xFFFAC34C [10], 0xFFFAC36C [11], 0xFFFAC38C [12], 0xFFFAC3AC [13], 0xFFFAC3CC [14], 0xFFFAC3EC [15]

Access: Read-only

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

- **MFID: Family ID**

This field contains the concatenation of CAN_MIDx register bits masked by the CAN_MAMx register. This field is useful to speed up message ID decoding. The message acceptance procedure is described below.

As an example:

```
CAN_MIDx = 0x305A4321
CAN_MAMx = 0x3FF0F0FF
CAN_MFIDx = 0x000000A3
```

40.8.16 CAN Message Status Register

Name: CAN_MSRx [x=0..15]

Addresses: 0xFFFAC210 [0], 0xFFFAC230 [1], 0xFFFAC250 [2], 0xFFFAC270 [3], 0xFFFAC290 [4], 0xFFFAC2B0 [5], 0xFFFAC2D0 [6], 0xFFFAC2F0 [7], 0xFFFAC310 [8], 0xFFFAC330 [9], 0xFFFAC350 [10], 0xFFFAC370 [11], 0xFFFAC390 [12], 0xFFFAC3B0 [13], 0xFFFAC3D0 [14], 0xFFFAC3F0 [15]

Access: Read-only

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

23	22	21	20	19	18	17	16
MRDY	MABT	–	MRTR	MDLC			

15	14	13	12	11	10	9	8
MTIMESTAMP15	MTIMESTAMP14	MTIMESTAMP13	MTIMESTAMP12	MTIMESTAMP11	MTIMESTAMP10	MTIMESTAMP9	MTIMESTAMP8

7	6	5	4	3	2	1	0
MTIMESTAMP7	MTIMESTAMP6	MTIMESTAMP5	MTIMESTAMP4	MTIMESTAMP3	MTIMESTAMP2	MTIMESTAMP1	MTIMESTAMP0

These register fields are updated each time a message transfer is received or aborted.

MMI is cleared by reading the CAN_MSRx register.

MRDY, MABT are cleared by writing MTCR or MACR in the CAN_MCRx register.

Warning: MRTR and MDLC state depends partly on the mailbox object type.

- **MTIMESTAMPx: Timer value**

This field is updated only when time-triggered operations are disabled (TTM cleared in CAN_MR register). If the TEOF field in the CAN_MR register is cleared, TIMESTAMP is the internal timer value at the start of frame of the last message received or sent by the mailbox. If the TEOF field in the CAN_MR register is set, TIMESTAMP is the internal timer value at the end of frame of the last message received or sent by the mailbox.

In Time Triggered Mode, MTIMESTAMP is set to 0.

- **MDLC: Mailbox Data Length Code**

Mailbox Object Type	Description
Receive	Length of the first mailbox message received
Receive with overwrite	Length of the last mailbox message received
Transmit	No action
Consumer	Length of the mailbox message received
Producer	Length of the mailbox message to be sent after the remote frame reception

- **MRTR: Mailbox Remote Transmission Request**

Mailbox Object Type	Description
Receive	The first frame received has the RTR bit set.
Receive with overwrite	The last frame received has the RTR bit set.
Transmit	Reserved
Consumer	Reserved. After setting the MOT field in the CAN_MMR, MRTR is reset to 1.
Producer	Reserved. After setting the MOT field in the CAN_MMR, MRTR is reset to 0.

- **MABT: Mailbox Message Abort**

An interrupt is triggered when MABT is set.

0 = Previous transfer is not aborted.

1 = Previous transfer has been aborted.

This flag is cleared by writing to CAN_MCRx register

Mailbox Object Type	Description
Receive	Reserved
Receive with overwrite	Reserved
Transmit	Previous transfer has been aborted
Consumer	The remote frame transfer request has been aborted.
Producer	The response to the remote frame transfer has been aborted.

- **MRDY: Mailbox Ready**

An interrupt is triggered when MRDY is set.

0 = Mailbox data registers can not be read/written by the software application. CAN_MDx are locked by the CAN_MDx.

1 = Mailbox data registers can be read/written by the software application.

This flag is cleared by writing to CAN_MCRx register.

Mailbox Object Type	Description
Receive	At least one message has been received since the last mailbox transfer order. Data from the first frame received can be read in the CAN_MDxx registers. After setting the MOT field in the CAN_MMR, MRDY is reset to 0.
Receive with overwrite	At least one frame has been received since the last mailbox transfer order. Data from the last frame received can be read in the CAN_MDxx registers. After setting the MOT field in the CAN_MMR, MRDY is reset to 0.
Transmit	Mailbox data have been transmitted. After setting the MOT field in the CAN_MMR, MRDY is reset to 1.
Consumer	At least one message has been received since the last mailbox transfer order. Data from the first message received can be read in the CAN_MDxx registers. After setting the MOT field in the CAN_MMR, MRDY is reset to 0.
Producer	A remote frame has been received, mailbox data have been transmitted. After setting the MOT field in the CAN_MMR, MRDY is reset to 1.

- **MMI: Mailbox Message Ignored**

0 = No message has been ignored during the previous transfer

1 = At least one message has been ignored during the previous transfer

Cleared by reading the CAN_MSRx register.

Mailbox Object Type	Description
Receive	Set when at least two messages intended for the mailbox have been sent. The first one is available in the mailbox data register. Others have been ignored. A mailbox with a lower priority may have accepted the message.
Receive with overwrite	Set when at least two messages intended for the mailbox have been sent. The last one is available in the mailbox data register. Previous ones have been lost.
Transmit	Reserved
Consumer	A remote frame has been sent by the mailbox but several messages have been received. The first one is available in the mailbox data register. Others have been ignored. Another mailbox with a lower priority may have accepted the message.
Producer	A remote frame has been received, but no data are available to be sent.

40.8.17 CAN Message Data Low Register

Name: CAN_MDLx [x=0..15]

Addresses: 0xFFFAC214 [0], 0xFFFAC234 [1], 0xFFFAC254 [2], 0xFFFAC274 [3], 0xFFFAC294 [4], 0xFFFAC2B4 [5], 0xFFFAC2D4 [6], 0xFFFAC2F4 [7], 0xFFFAC314 [8], 0xFFFAC334 [9], 0xFFFAC354 [10], 0xFFFAC374 [11], 0xFFFAC394 [12], 0xFFFAC3B4 [13], 0xFFFAC3D4 [14], 0xFFFAC3F4 [15]

Access: Read-write

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

- **MDL: Message Data Low Value**

When MRDY field is set in the CAN_MSRx register, the lower 32 bits of a received message can be read or written by the software application. Otherwise, the MDL value is locked by the CAN controller to send/receive a new message.

In Receive with overwrite, the CAN controller may modify MDL value while the software application reads MDH and MDL registers. To check that MDH and MDL do not belong to different messages, the application has to check the MMI field in the CAN_MSRx register. In this mode, the software application must re-read CAN_MDH and CAN_MDL, while the MMI bit in the CAN_MSRx register is set.

Bytes are received/sent on the bus in the following order:

1. CAN_MDL[7:0]
2. CAN_MDL[15:8]
3. CAN_MDL[23:16]
4. CAN_MDL[31:24]
5. CAN_MDH[7:0]
6. CAN_MDH[15:8]
7. CAN_MDH[23:16]
8. CAN_MDH[31:24]

40.8.18 CAN Message Data High Register

Name: CAN_MDHx [x=0..15]

Addresses: 0xFFFAC218 [0], 0xFFFAC238 [1], 0xFFFAC258 [2], 0xFFFAC278 [3], 0xFFFAC298 [4], 0xFFFAC2B8 [5], 0xFFFAC2D8 [6], 0xFFFAC2F8 [7], 0xFFFAC318 [8], 0xFFFAC338 [9], 0xFFFAC358 [10], 0xFFFAC378 [11], 0xFFFAC398 [12], 0xFFFAC3B8 [13], 0xFFFAC3D8 [14], 0xFFFAC3F8 [15]

Access: Read-write

31	30	29	28	27	26	25	24
MDH							

23	22	21	20	19	18	17	16
MDH							

15	14	13	12	11	10	9	8
MDH							

7	6	5	4	3	2	1	0
MDH							

- **MDH: Message Data High Value**

When MRDY field is set in the CAN_MSRx register, the upper 32 bits of a received message are read or written by the software application. Otherwise, the MDH value is locked by the CAN controller to send/receive a new message.

In Receive with overwrite, the CAN controller may modify MDH value while the software application reads MDH and MDL registers. To check that MDH and MDL do not belong to different messages, the application has to check the MMI field in the CAN_MSRx register. In this mode, the software application must re-read CAN_MDH and CAN_MDL, while the MMI bit in the CAN_MSRx register is set.

Bytes are received/sent on the bus in the following order:

1. CAN_MDL[7:0]
2. CAN_MDL[15:8]
3. CAN_MDL[23:16]
4. CAN_MDL[31:24]
5. CAN_MDH[7:0]
6. CAN_MDH[15:8]
7. CAN_MDH[23:16]
8. CAN_MDH[31:24]

40.8.19 CAN Message Control Register

Name: CAN_MCRx [x=0..15]

Addresses: 0xFFFAC21C [0], 0xFFFAC23C [1], 0xFFFAC25C [2], 0xFFFAC27C [3], 0xFFFAC29C [4], 0xFFFAC2BC [5], 0xFFFAC2DC [6], 0xFFFAC2FC [7], 0xFFFAC31C [8], 0xFFFAC33C [9], 0xFFFAC35C [10], 0xFFFAC37C [11], 0xFFFAC39C [12], 0xFFFAC3BC [13], 0xFFFAC3DC [14], 0xFFFAC3FC [15]

Access: Write-only

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

23	22	21	20	19	18	17	16
MTCR	MACR	–	MRTR	MDLC			

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **MDLC: Mailbox Data Length Code**

Mailbox Object Type	Description
Receive	No action.
Receive with overwrite	No action.
Transmit	Length of the mailbox message.
Consumer	No action.
Producer	Length of the mailbox message to be sent after the remote frame reception.

- **MRTR: Mailbox Remote Transmission Request**

Mailbox Object Type	Description
Receive	No action
Receive with overwrite	No action
Transmit	Set the RTR bit in the sent frame
Consumer	No action, the RTR bit in the sent frame is set automatically
Producer	No action

Consumer situations can be handled automatically by setting the mailbox object type in Consumer. This requires only one mailbox.

It can also be handled using two mailboxes, one in reception, the other in transmission. The MRTR and the MTCR bits must be set in the same time.

- **MACR: Abort Request for Mailbox x**

Mailbox Object Type	Description
Receive	No action
Receive with overwrite	No action
Transmit	Cancels transfer request if the message has not been transmitted to the CAN transceiver.
Consumer	Cancels the current transfer before the remote frame has been sent.
Producer	Cancels the current transfer. The next remote frame will not be serviced.

It is possible to set MACR field for several mailboxes in the same time, setting several bits to the CAN_ACR register.

- **MTCR: Mailbox Transfer Command**

Mailbox Object Type	Description
Receive	Allows the reception of the next message.
Receive with overwrite	Triggers a new reception.
Transmit	Sends data prepared in the mailbox as soon as possible.
Consumer	Sends a remote transmission frame.
Producer	Sends data prepared in the mailbox after receiving a remote frame from a Consumer.

This flag clears the MRDY and MABT flags in the CAN_MSRx register.

When several mailboxes are requested to be transmitted simultaneously, they are transmitted in turn. The mailbox with the highest priority is serviced first. If several mailboxes have the same priority, the mailbox with the lowest number is serviced first (i.e., MBx0 will be serviced before MBx 15 if they have the same priority).

It is possible to set MTCR for several mailboxes at the same time by writing to the CAN_TCR register.

41. Pulse Width Modulation Controller (PWM)

41.1 Description

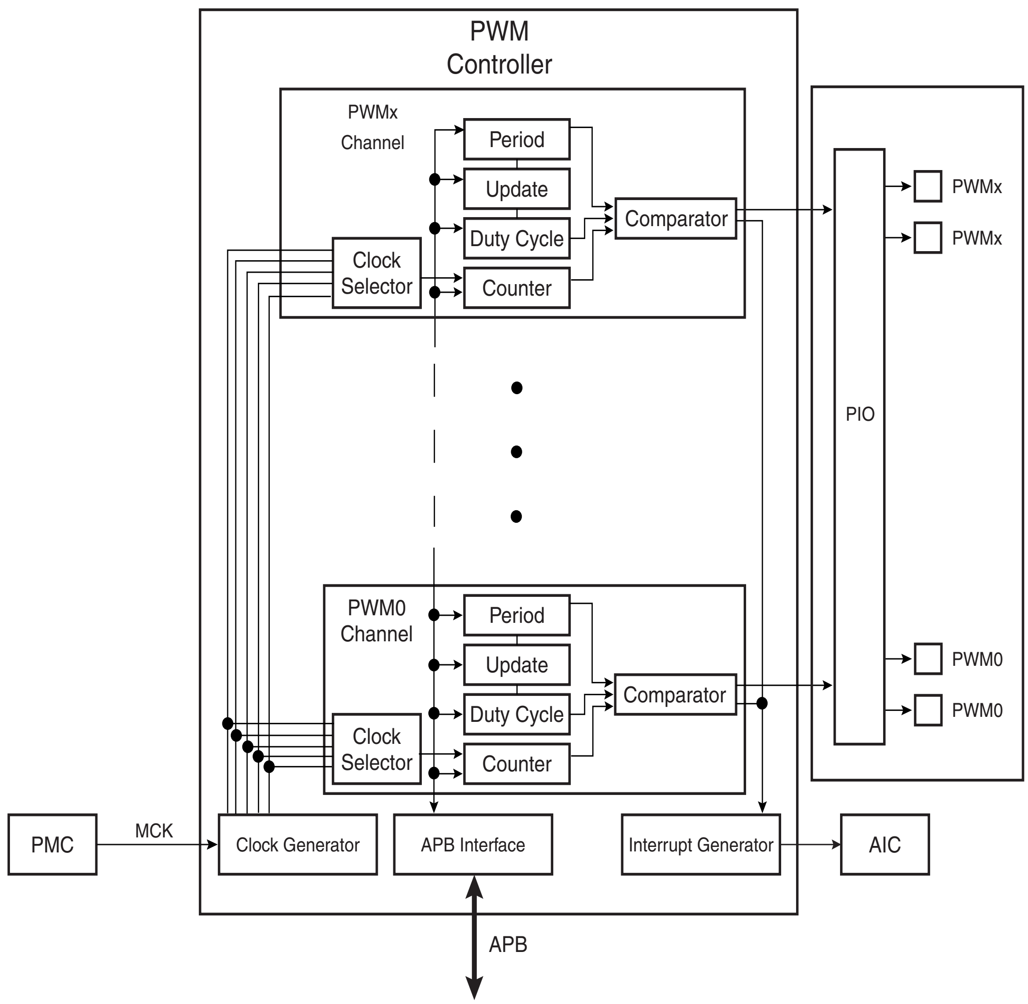
The PWM macrocell controls several channels independently. Each channel controls one square output waveform. Characteristics of the output waveform such as period, duty-cycle and polarity are configurable through the user interface. Each channel selects and uses one of the clocks provided by the clock 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.

41.2 Block Diagram

Figure 41-1. Pulse Width Modulation Controller Block Diagram



41.3 I/O Lines Description

Each channel outputs one waveform on one external I/O line.

Table 41-1. I/O Line Description

Name	Description	Type
PWMx	PWM Waveform Output for channel x	Output

41.4 Product Dependencies

41.4.1 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.

Table 41-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
PWM	PWM0	PB19	A
PWM	PWM0	PC28	A
PWM	PWM1	PA10	B
PWM	PWM1	PB20	A
PWM	PWM1	PC3	B
PWM	PWM2	PB8	B
PWM	PWM2	PC29	B
PWM	PWM3	PA11	B
PWM	PWM3	PB29	B

41.4.2 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.

41.4.3 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.

Table 41-3. Peripheral IDs

Instance	ID
PWM	21

41.5 Functional Description

The PWM macrocell is primarily composed of a clock generator module and 4 channels.

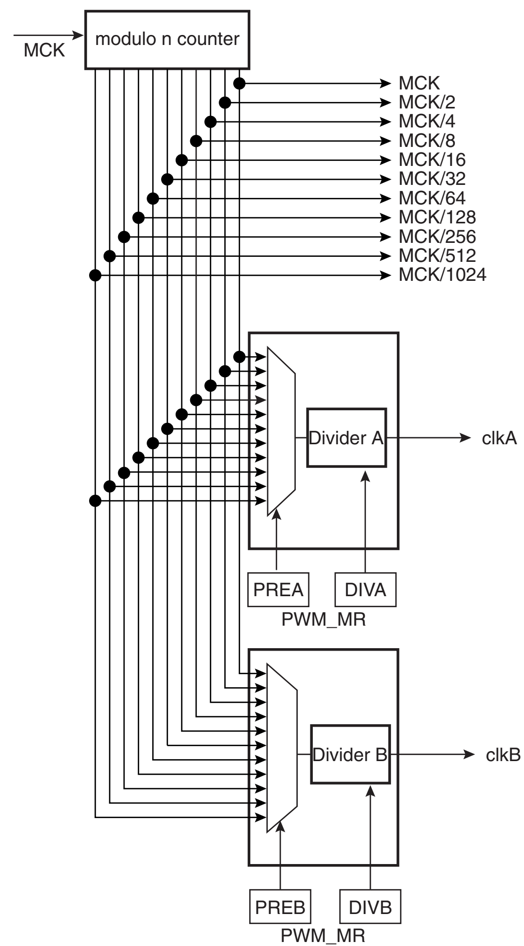
– Clocked by the system clock, MCK, the clock generator module provides 13 clocks.

– 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.

41.5.1 PWM Clock Generator

Figure 41-2. 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).

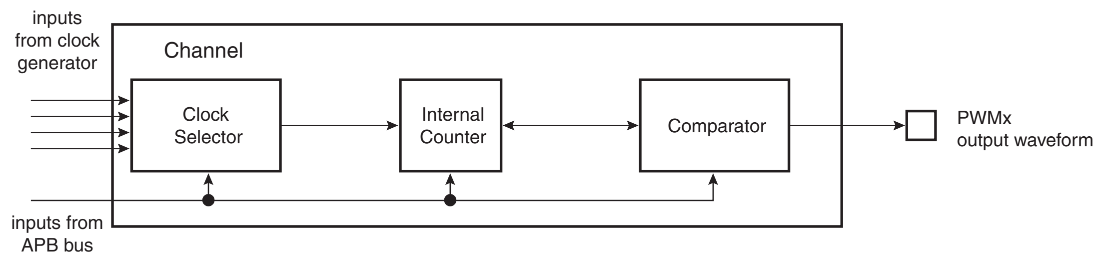
After a reset of the PWM controller, DIVA (DIVB) and PREA (PREB) in the PWM Mode register are set to 0. This implies that after reset clkA (clkB) are turned off.

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.

41.5.2 PWM Channel

41.5.2.1 Block Diagram

Figure 41-3. 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 41.5.1 “PWM Clock Generator” on page 793.
- 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 20 bits.
- A comparator used to generate events according to the internal counter value. It also computes the PWMx output waveform according to the configuration.

41.5.2.2 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:

$$\frac{(CPRD \times DIVA)}{MCK} \text{ or } \frac{(CPRD \times DIVAB)}{MCK}$$

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:

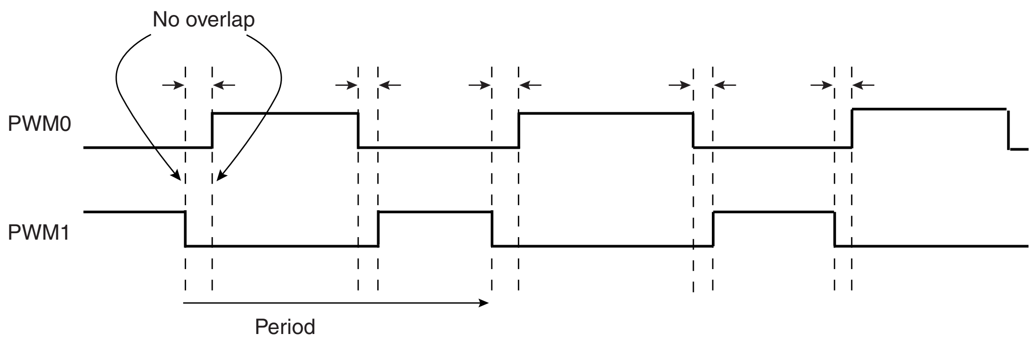
  $$\text{duty cycle} = \frac{(period - 1/\ \text{fchannel\_x\_clock} \times CDTY)}{period}$$

  If the waveform is center aligned, then:

  $$\text{duty cycle} = \frac{((period/\ 2) - 1/\ \text{fchannel\_x\_clock} \times CDTY))}{(period/\ 2)}$$

- 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 41-4. Non Overlapped Center Aligned Waveforms



Note: 1. See Figure 41-5 on page 797 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:

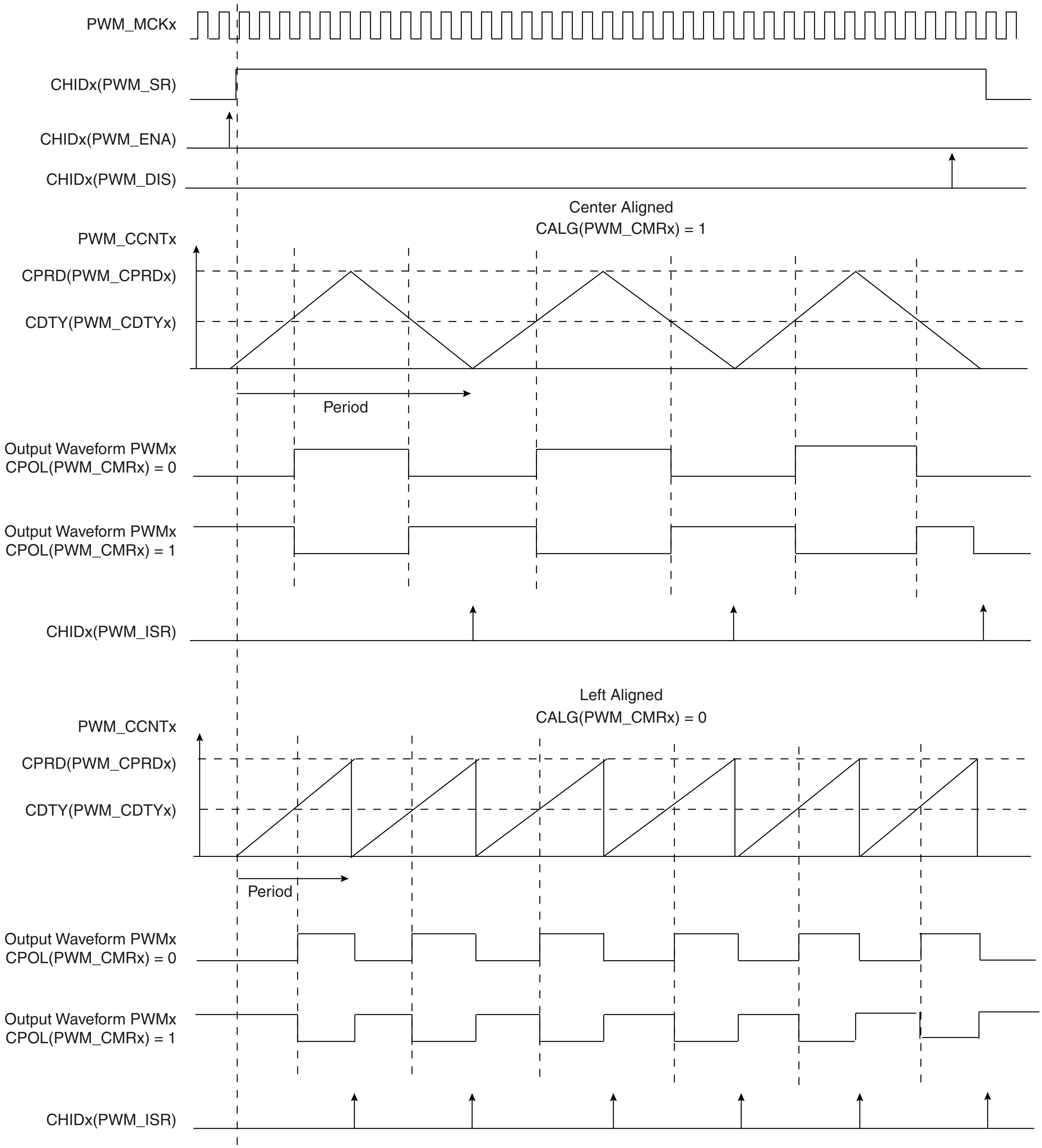
- CDTY = CPRD and CPOL = 0
- CDTY = 0 and CPOL = 1

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 41-5. Waveform Properties



41.5.3 PWM Controller Operations

41.5.3.1 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.

41.5.3.2 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.

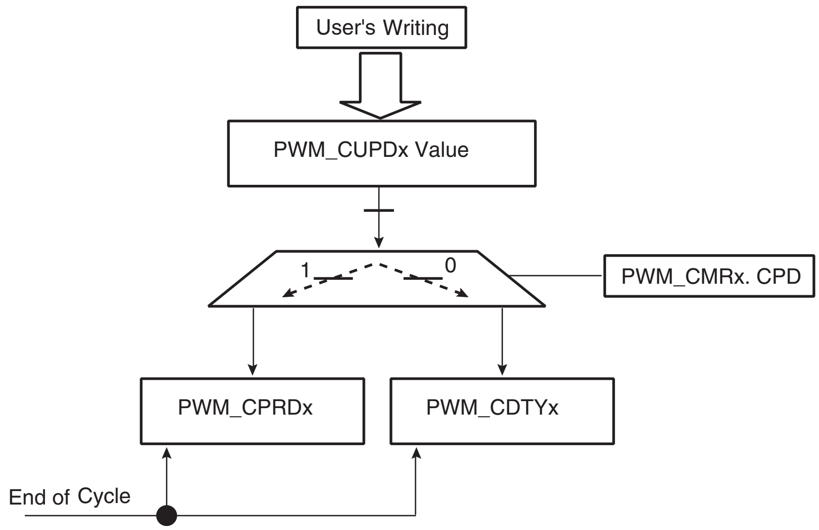
For example, if the user sets 15 (in decimal) in PWM_CPRDx, the user is able to set a value between 1 up to 14 in PWM_CDTYx Register. The resulting duty cycle quantum cannot be lower than 1/15 of the PWM period.

41.5.3.3 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 41-6. Synchronized Period or Duty Cycle Update



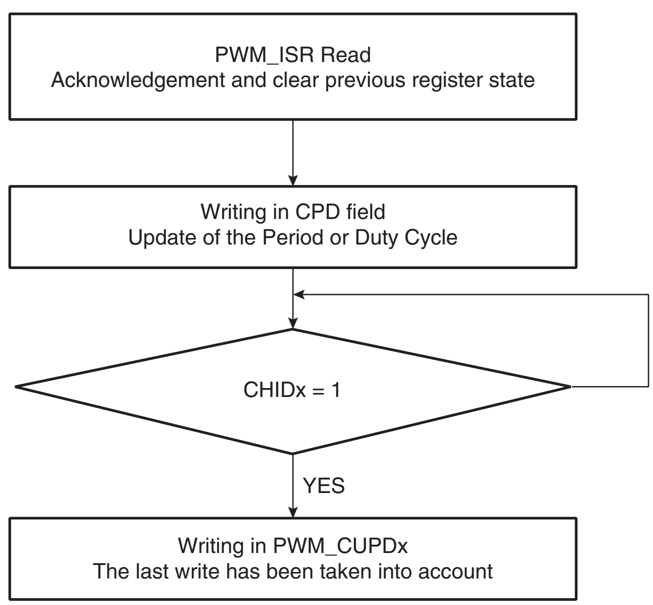
To prevent overwriting the PWM_CUPDx by software, the user can use status events in order to synchronize his software. Two methods are possible. In both, the user must enable the dedicated interrupt in PWM_IER at PWM Controller level.

The first method (polling method) consists of reading the relevant status bit in PWM_ISR Register according to the enabled channel(s). See Figure 41-7.

The second method uses an Interrupt Service Routine associated with the PWM channel.

Note: Reading the PWM_ISR register automatically clears CHIDx flags.

Figure 41-7. Polling Method



Note: Polarity and alignment can be modified only when the channel is disabled.

41.5.3.4 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.

41.6 Pulse Width Modulation Controller (PWM) User Interface

Table 41-4. Register Mapping(2)

Offset	Register	Name	Access	Reset
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
0x100 - 0x1FC	Reserved			
0x200 + ch_num * 0x20 + 0x00	PWM Channel Mode Register	PWM_CMR	Read-write	0x0
0x200 + ch_num * 0x20 + 0x04	PWM Channel Duty Cycle Register	PWM_CDTY	Read-write	0x0
0x200 + ch_num * 0x20 + 0x08	PWM Channel Period Register	PWM_CPRD	Read-write	0x0
0x200 + ch_num * 0x20 + 0x0C	PWM Channel Counter Register	PWM_CCNT	Read-only	0x0
0x200 + ch_num * 0x20 + 0x10	PWM Channel Update Register	PWM_CUPD	Write-only	-

2. Some registers are indexed with “ch_num” index ranging from 0 to 3.

41.6.1 PWM Mode Register

Name: PWM_MR

Address: 0xFFFB8000

Access: 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

41.6.2 PWM Enable Register

Name: PWM_ENA

Address: 0xFFFB8004

Access: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

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.

41.6.3 PWM Disable Register

Name: PWM_DIS

Address: 0xFFFB8008

Access: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

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.

41.6.4 PWM Status Register

Name: PWM_SR

Address: 0xFFFB800C

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = PWM output for channel x is disabled.

1 = PWM output for channel x is enabled.

41.6.5 PWM Interrupt Enable Register

Name: PWM_IER

Address: 0xFFFB8010

Access: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

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.

41.6.6 PWM Interrupt Disable Register

Name: PWM_IDR

Address: 0xFFFB8014

Access: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

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.

41.6.7 PWM Interrupt Mask Register

Name: PWM_IMR

Address: 0xFFFB8018

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID.**

0 = Interrupt for PWM channel x is disabled.

1 = Interrupt for PWM channel x is enabled.

41.6.8 PWM Interrupt Status Register

Name: PWM_ISR

Address: 0xFFFB801C

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	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.

41.6.9 PWM Channel Mode Register

Name: PWM_CMR[0..3]

Addresses: 0xFFFB8200 [0], 0xFFFB8220 [1], 0xFFFB8240 [2], 0xFFFB8260 [3]

Access: 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.

41.6.10 PWM Channel Duty Cycle Register

Name: PWM_CDTY[0..3]

Addresses: 0xFFFB8204 [0], 0xFFFB8224 [1], 0xFFFB8244 [2], 0xFFFB8264 [3]

Access: 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 20 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).

41.6.11 PWM Channel Period Register

Name: PWM_CPRD[0..3]

Addresses: 0xFFFB8208 [0], 0xFFFB8228 [1], 0xFFFB8248 [2], 0xFFFB8268 [3]

Access: 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 20 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:

$$\frac{(CRPD \times DIVA)}{MCK} \text{ or } \frac{(CRPD \times DIVAB)}{MCK}$$

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:

$$\frac{(2 \times CPRD \times DIVA)}{MCK} \text{ or } \frac{(2 \times CPRD \times DIVB)}{MCK}$$

41.6.12 PWM Channel Counter Register

Name: PWM_CCNT[0..3]

Addresses: 0xFFFB820C [0], 0xFFFB822C [1], 0xFFFB824C [2], 0xFFFB826C [3]

Access: 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.

41.6.13 PWM Channel Update Register

Name: PWM_CUPD[0..3]

Addresses: 0xFFFB8210 [0], 0xFFFB8230 [1], 0xFFFB8250 [2], 0xFFFB8270 [3]

Access: 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 20 bits (internal channel counter size) are significant.

CPD (PWM_CMRx Register)	
0	The duty-cycle (CDTY in the PWM_CDTYx register) is updated with the CUPD value at the beginning of the next period.
1	The period (CPRD in the PWM_CPRDx register) is updated with the CUPD value at the beginning of the next period.

42. MultiMedia Card Interface (MCI)

42.1 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.

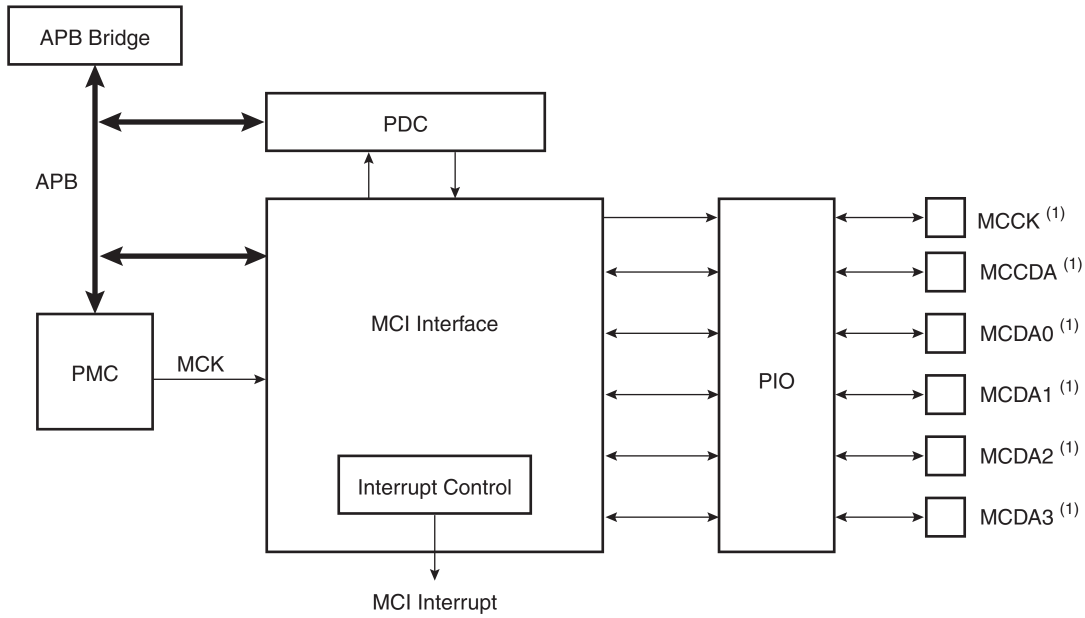
The MCI operates at a rate of up to Master Clock divided by 2 and supports the interfacing of 1 slot(s). Each slot may be used to interface with a MultiMediaCard bus (up to 30 Cards) or with a SD Memory Card. Only one slot can be selected at a time (slots are multiplexed). A bit field in the SD Card 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.

42.2 Block Diagram

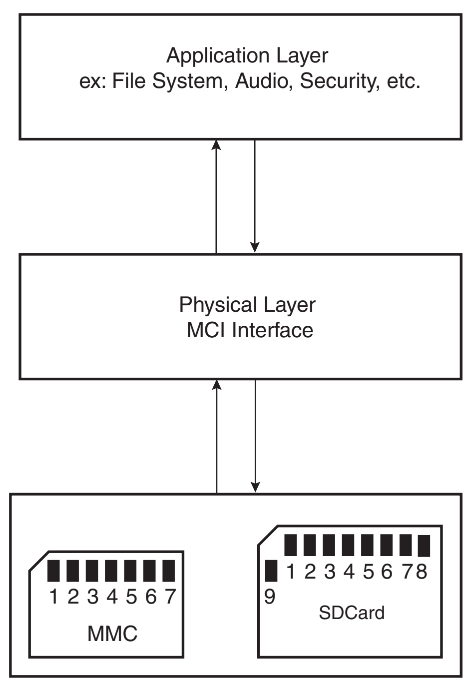
Figure 42-1. Block Diagram



Note: 1. When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx_CK, MCCDA to MCIx_CDA, MCDAy to MCIx_DAy.

42.3 Application Block Diagram

Figure 42-2. Application Block Diagram



42.4 Pin Name List

Table 42-1. I/O Lines Description

Pin Name[(2)]	Pin Description	Type[(1)]	Comments
MCCDA	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

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, MCDAy to MCIx_DAy.

42.5 Product Dependencies

42.5.1 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.

Table 42-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
MCI0	MCI0_CD	PA1	A
MCI0	MCI0_CK	PA2	A
MCI0	MCI0_D0	PA0	A
MCI0	MCI0_D1	PA3	A
MCI0	MCI0_D2	PA4	A
MCI0	MCI0_D3	PA5	A
MCI1	MCI1_CD	PA17	A
MCI1	MCI1_CK	PA16	A
MCI1	MCI1_D0	PA18	A
MCI1	MCI1_D1	PA19	A
MCI1	MCI1_D2	PA20	A
MCI1	MCI1_D3	PA21	A

42.5.2 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.

42.5.3 Interrupt

The MCI interface has an interrupt line connected to the Advanced Interrupt Controller (AIC).

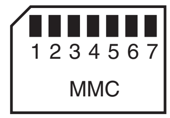
Handling the MCI interrupt requires programming the AIC before configuring the MCI.

Table 42-3. Peripheral IDs

Instance	ID
MCI0	11
MCI1	12

42.6 Bus Topology

Figure 42-3. Multimedia Memory Card Bus Topology



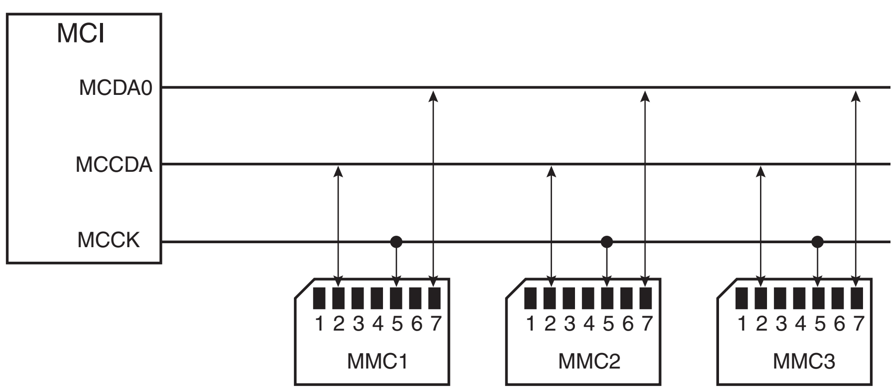
The MultiMedia Card communication is based on a 7-pin serial bus interface. It has three communication lines and four supply lines.

Table 42-4. Bus Topology

Pin Number	Name	Type[(1)]	Description	MCI Pin Name[(2)] (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	MCCK
6	VSS2	S	Supply voltage ground	VSS
7	DAT[0]	I/O/PP	Data 0	MCDz0

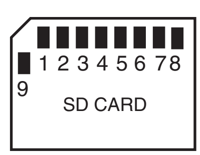
Notes: 1. I: Input, O: Output, PP: Push/Pull, OD: Open Drain.
2. When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx_CK, MCCDA to MCIx_CDA, MCDAy to MCIx_DAy.

Figure 42-4. MMC Bus Connections (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 42-5. SD Memory Card Bus Topology



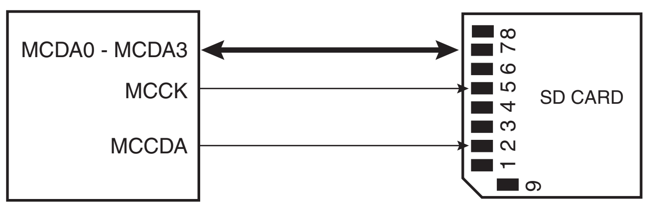
The SD Memory Card bus includes the signals listed in Table 42-5.

Table 42-5. SD Memory Card Bus Signals

Pin Number	Name	Type(1)	Description	MCI Pin Name(2) (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, MCDAy to MCIx_DAy.

Figure 42-6. 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.

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.

42.7 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 42-6 on page 821.

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 822.).

The MCI provides a set of registers to perform the entire range of MultiMedia Card operations.

42.7.1 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 will guarantee 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:

	Host Command					**N_{ID} Cycles**					**CID**			
CMD	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 42-6 and Table 42-7.

Table 42-6. 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 42-7. 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.

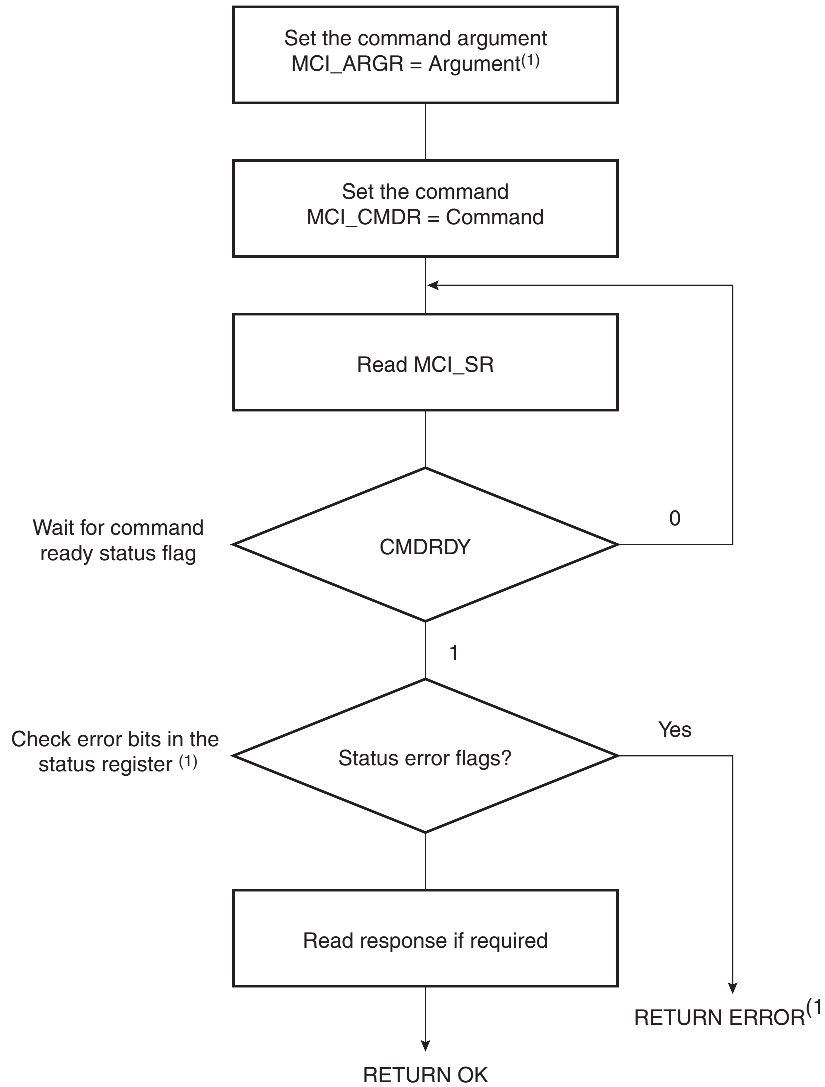
To send a command, the user must perform the following steps:

- Fill the argument register (MCI_ARGR) with the command argument.
- Set the command register (MCI_CMDR) (see Table 42-7).

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 42-7. 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).

42.7.2 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_BLKR. 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 will continuously transfer (or program) data blocks until a stop transmission command is received.

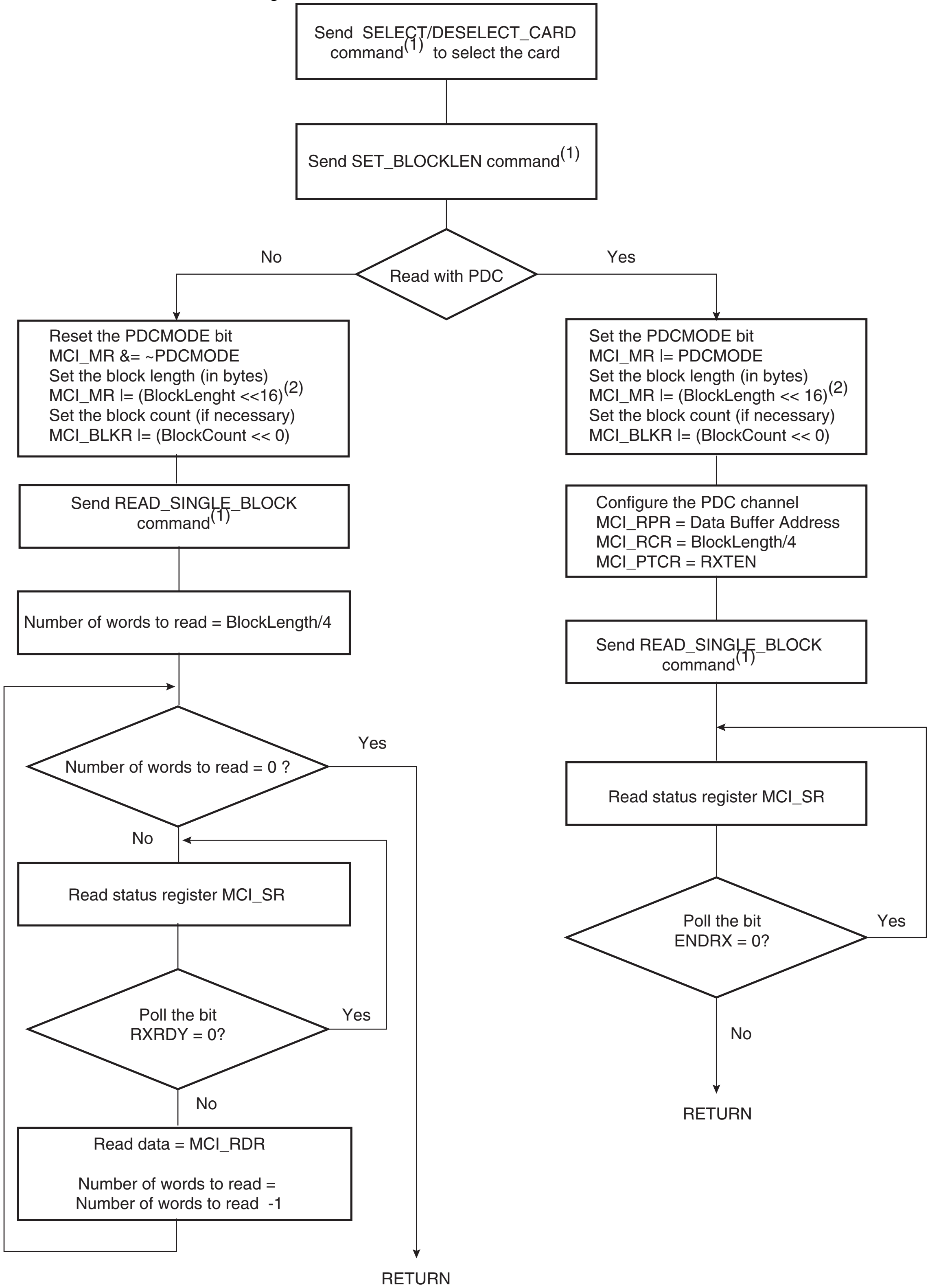
- Multiple block read (or write) with pre-defined block count (since version 3.1 and higher):

  The card will transfer (or program) 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_BLKR). Otherwise the card will start 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.

42.7.3 Read Operation

The following flowchart shows how to read a single block with or without use of PDC facilities. In this example (see Figure 42-8), 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 42-8. Read Functional Flow Diagram



Note: 1. It is assumed that this command has been correctly sent (see Figure 42-7).

2. This field is also accessible in the MCI Block Register (MCI_BLKR).

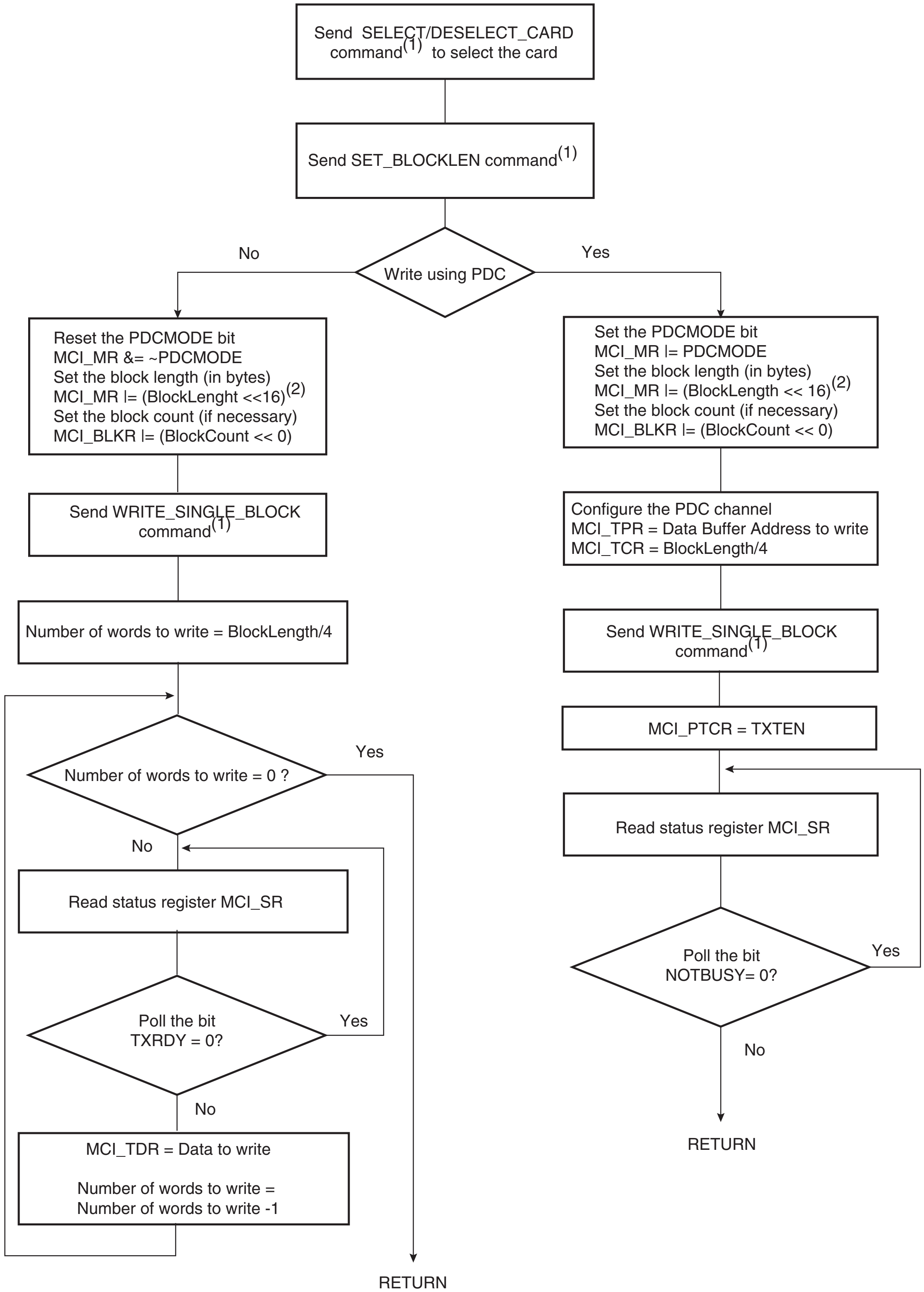
42.7.4 Write Operation

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 42-9). 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 42-9. Write Functional Flow Diagram

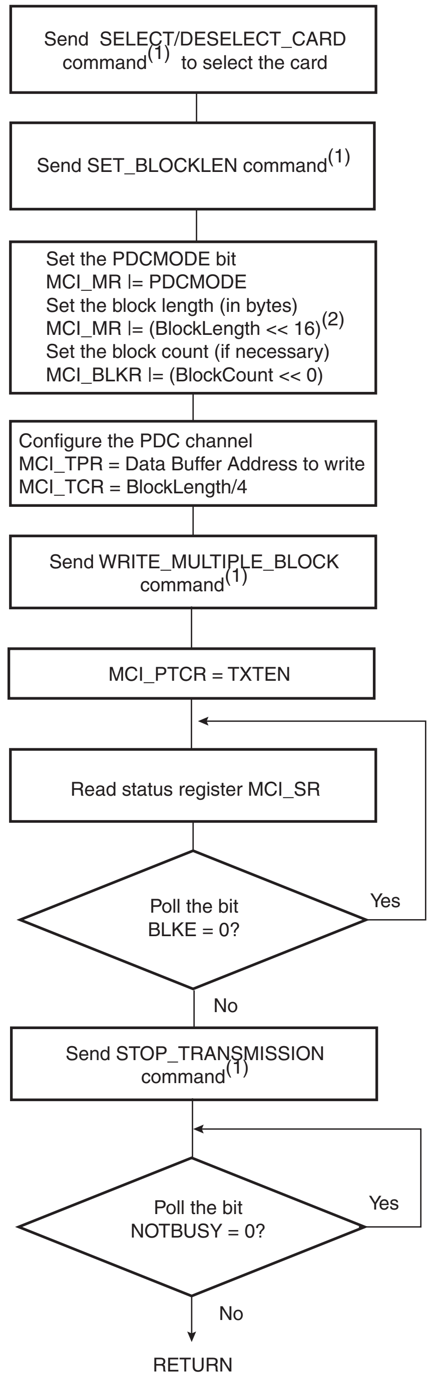


Note: 1. It is assumed that this command has been correctly sent (see Figure 42-7).

2. This field is also accessible in the MCI Block Register (MCI_BLKR).

The following flowchart shows how to manage a multiple write block transfer with the PDC (see Figure 42-10). 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 42-10. Multiple Write Functional Flow Diagram



Notes: 1. It is assumed that this command has been correctly sent (see Figure 42-7).

2. This field is also accessible in the MCI Block Register (MCI_BLKR).

42.8 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).

42.8.1 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_BLKR). 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.

42.8.2 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.

42.9 MultiMedia Card Interface (MCI) User Interface

Table 42-8. Register Mapping

Offset	Register	Register Name	Access	Reset
0x00	Control Register	MCI_CR	Write-only	–
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-only	–
0x18	Block Register	MCI_BLKR	Read-write	0x0
0x1C	Reserved	–	–	–
0x20	Response Register(1)	MCI_RSPR	Read-only	0x0
0x24	Response Register(1)	MCI_RSPR	Read-only	0x0
0x28	Response Register(1)	MCI_RSPR	Read-only	0x0
0x2C	Response Register(1)	MCI_RSPR	Read-only	0x0
0x30	Receive Data Register	MCI_RDR	Read-only	0x0
0x34	Transmit Data Register	MCI_TDR	Write-only	–
0x38 - 0x3C	Reserved	–	–	–
0x40	Status Register	MCI_SR	Read-only	0xC0E5
0x44	Interrupt Enable Register	MCI_IER	Write-only	–
0x48	Interrupt Disable Register	MCI_IDR	Write-only	–
0x4C	Interrupt Mask Register	MCI_IMR	Read-only	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.

42.9.1 MCI Control Register

Name: MCI_CR

Addresses: 0xFFF80000 (0), 0xFFF84000 (1)

Access: Write-only

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

42.9.2 MCI Mode Register

Name: MCI_MR

Addresses: 0xFFF80004 (0), 0xFFF84004 (1)

Access: 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.

<u>**Warning:**</u> 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 will guarantee 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 will guarantee 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.

<u>**Warning:**</u> 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_BLKR).

Bits 16 and 17 must be set to 0 if PDCFBYTE is disabled.

Note: In SDIO Byte mode, BLKLEN field is not used.

42.9.3 MCI Data Timeout Register

Name: MCI_DTOR

Addresses: 0xFFF80008 (0), 0xFFF84008 (1)

Access: Read-write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	DTOMUL			DTOCYC			

- **DTOCYC: Data Timeout Cycle Number**

Defines a number of Master Clock cycles with DTOMUL.

- **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 (DTOCYC 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 DTOCYC and DTOMUL has been exceeded, the Data Time-out Error flag (DTOE) in the MCI Status Register (MCI_SR) raises.

42.9.4 MCI SDCard/SDIO Register

Name: MCI_SDCR

Addresses: 0xFFF8000C (0), 0xFFF8400C (1)

Access: 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	Reserved
1	0	Reserved
1	1	Reserved

- **SDCBUS: SDCard/SDIO Bus Width**

0 = 1-bit data bus

1 = 4-bit data bus

42.9.5 MCI Argument Register

Name: MCI_ARGR

Addresses: 0xFFF80010 (0), 0xFFF84010 (1)

Access: Read-write

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

- **ARG: Command Argument**

42.9.6 MCI Command Register

Name: MCI_CMDR

Addresses: 0xFFF80014 (0), 0xFFF84014 (1)

Access: Write-only

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

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**

MultiMedia Card bus command numbers are defined in the MultiMedia Card specification.

- **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

42.9.7 MCI Block Register

Name: MCI_BLKR

Addresses: 0xFFF80018 (0), 0xFFF84018 (1)

Access: 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**
BCNT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BCNT							

- **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 65535: 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.

42.9.8 MCI Response Register

Name: MCI_RSPR

Addresses: 0xFFF80020 (0), 0xFFF84020 (1)

Access: Read-only

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

- **RSP: Response**

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.

42.9.9 MCI Receive Data Register

Name: MCI_RDR

Addresses: 0xFFF80030 (0), 0xFFF84030 (1)

Access: 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**

42.9.10 MCI Transmit Data Register

Name: MCI_TDR

Addresses: 0xFFF80034 (0), 0xFFF84034 (1)

Access: 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**

42.9.11 MCI Status Register

Name: MCI_SR

Addresses: 0xFFF80040 (0), 0xFFF84040 (1)

Access: 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	–	–	–	–	–	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. Cleared 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. Cleared 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.

- **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.

42.9.12 MCI Interrupt Enable Register

Name: MCI_IER

Addresses: 0xFFF80044 (0), 0xFFF84044 (1)

Access: 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	–	–	–	–	–	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**
- **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.

42.9.13 MCI Interrupt Disable Register

Name: MCI_IDR

Addresses: 0xFFF80048 (0), 0xFFF84048 (1)

Access: 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	–	–	–	–	–	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**
- **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.

42.9.14 MCI Interrupt Mask Register

Name: MCI_IMR

Addresses: 0xFFF8004C (0), 0xFFF8404C (1)

Access: 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	–	–	–	–	–	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**
- **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.

43. 10/100 Ethernet MAC (EMAC)

43.1 Description

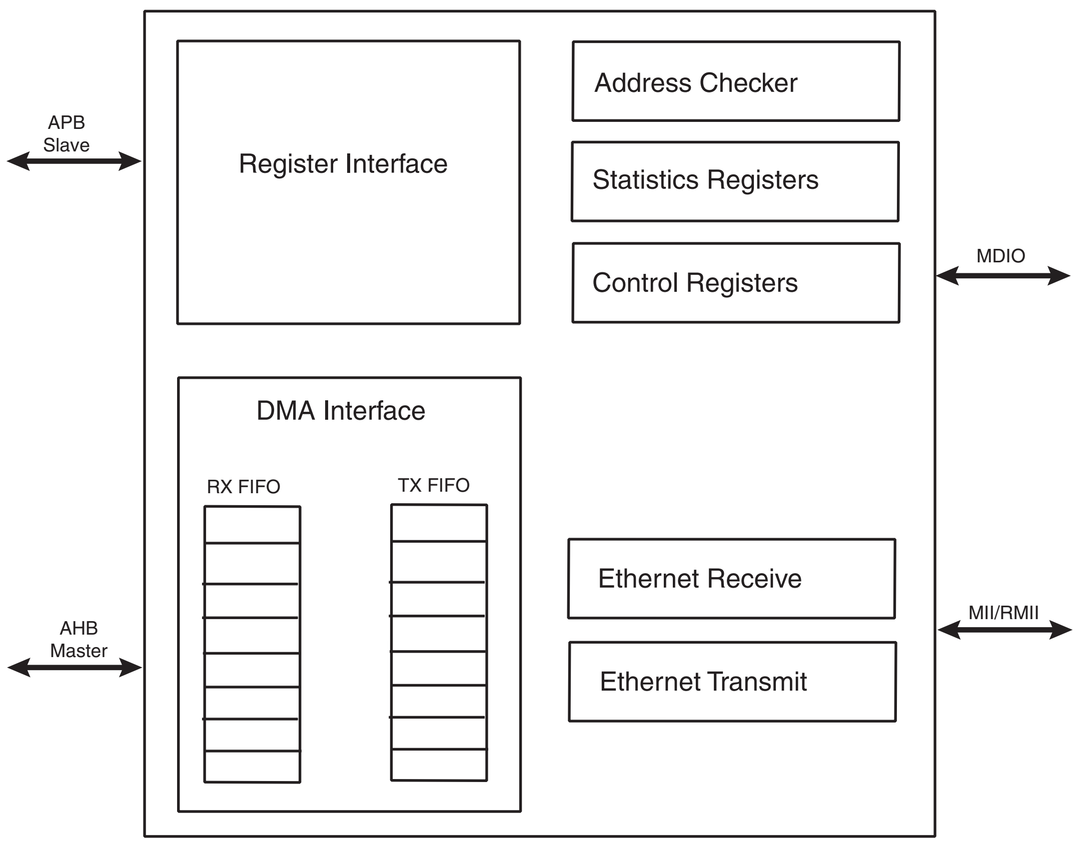
The EMAC module implements a 10/100 Ethernet MAC compatible with the IEEE 802.3 standard using an address checker, statistics and control registers, receive and transmit blocks, and a DMA interface.

The address checker recognizes four specific 48-bit addresses and contains a 64-bit hash register for matching multicast and unicast addresses. It can recognize the broadcast address of all ones, copy all frames, and act on an external address match signal.

The statistics register block contains registers for counting various types of event associated with transmit and receive operations. These registers, along with the status words stored in the receive buffer list, enable software to generate network management statistics compatible with IEEE 802.3.

43.2 Block Diagram

Figure 43-1. EMAC Block Diagram



43.3 Functional Description

The MACB has several clock domains:

- System bus clock (AHB and APB): DMA and register blocks
- Transmit clock: transmit block
- Receive clock: receive and address checker blocks

The only system constraint is 160 MHz for the system bus clock, above which MDC would toggle at above 2.5 MHz.

The system bus clock must run at least as fast as the receive clock and transmit clock (25 MHz at 100 Mbps, and 2.5 MHZ at 10 Mbps).

Figure 43-1 illustrates the different blocks of the EMAC module.

The control registers drive the MDIO interface, setup up DMA activity, start frame transmission and select modes of operation such as full- or half-duplex.

The receive block checks for valid preamble, FCS, alignment and length, and presents received frames to the address checking block and DMA interface.

The transmit block takes data from the DMA interface, adds preamble and, if necessary, pad and FCS, and transmits data according to the CSMA/CD (carrier sense multiple access with collision detect) protocol. The start of transmission is deferred if CRS (carrier sense) is active.

If COL (collision) becomes active during transmission, a jam sequence is asserted and the transmission is retried after a random back off. CRS and COL have no effect in full duplex mode.

The DMA block connects to external memory through its AHB bus interface. It contains receive and transmit FIFOs for buffering frame data. It loads the transmit FIFO and empties the receive FIFO using AHB bus master operations. Receive data is not sent to memory until the address checking logic has determined that the frame should be copied. Receive or transmit frames are stored in one or more buffers. Receive buffers have a fixed length of 128 bytes. Transmit buffers range in length between 0 and 2047 bytes, and up to 128 buffers are permitted per frame. The DMA block manages the transmit and receive framebuffer queues. These queues can hold multiple frames.

43.3.1 Clock

The Synchronization module in the EMAC requires that the bus clock (hclk) runs at the speed of the macb_tx/rx_clk at least, which is 25 MHz at 100 Mbps, and 2.5 MHz at 10 Mbps.

43.3.2 Memory Interface

Frame data is transferred to and from the EMAC through the DMA interface. All transfers are 32-bit words and may be single accesses or bursts of 2, 3 or 4 words. Burst accesses do not cross sixteen-byte boundaries. Bursts of 4 words are the default data transfer; single accesses or bursts of less than four words may be used to transfer data at the beginning or the end of a buffer.

The DMA controller performs six types of operation on the bus. In order of priority, these are:

1. Receive buffer manager write
2. Receive buffer manager read
3. Transmit data DMA read
4. Receive data DMA write

5. Transmit buffer manager read
6. Transmit buffer manager write

43.3.2.1 FIFO

The FIFO depths are 128 bytes for receive and 128 bytes for transmit and are a function of the system clock speed, memory latency and network speed.

Data is typically transferred into and out of the FIFOs in bursts of four words. For receive, a bus request is asserted when the FIFO contains four words and has space for 28 more. For transmit, a bus request is generated when there is space for four words, or when there is space for 27 words if the next transfer is to be only one or two words.

Thus the bus latency must be less than the time it takes to load the FIFO and transmit or receive three words (112 bytes) of data.

At 100 Mbit/s, it takes 8960 ns to transmit or receive 112 bytes of data. In addition, six master clock cycles should be allowed for data to be loaded from the bus and to propagate through the FIFOs. For a 133 MHz master clock this takes 45 ns, making the bus latency requirement 8915 ns.

43.3.2.2 Receive Buffers

Received frames, including CRC/FCS optionally, are written to receive buffers stored in memory. Each receive buffer is 128 bytes long. The start location for each receive buffer is stored in memory in a list of receive buffer descriptors at a location pointed to by the receive buffer queue pointer register. The receive buffer start location is a word address. For the first buffer of a frame, the start location can be offset by up to three bytes depending on the value written to bits 14 and 15 of the network configuration register. If the start location of the buffer is offset the available length of the first buffer of a frame is reduced by the corresponding number of bytes.

Each list entry consists of two words, the first being the address of the receive buffer and the second being the receive status. If the length of a receive frame exceeds the buffer length, the status word for the used buffer is written with zeroes except for the “start of frame” bit and the offset bits, if appropriate. Bit zero of the address field is written to one to show the buffer has been used. The receive buffer manager then reads the location of the next receive buffer and fills that with receive frame data. The final buffer descriptor status word contains the complete frame status. Refer to Table 43-1 for details of the receive buffer descriptor list.

Table 43-1. Receive Buffer Descriptor Entry

Bit	Function
Word 0	
31:2	Address of beginning of buffer
1	Wrap - marks last descriptor in receive buffer descriptor list.
0	Ownership - needs to be zero for the EMAC to write data to the receive buffer. The EMAC sets this to one once it has successfully written a frame to memory. Software has to clear this bit before the buffer can be used again.
Word 1	
31	Global all ones broadcast address detected
30	Multicast hash match
29	Unicast hash match

Table 43-1. Receive Buffer Descriptor Entry (Continued)

Bit	Function
28	External address match
27	Reserved for future use
26	Specific address register 1 match
25	Specific address register 2 match
24	Specific address register 3 match
23	Specific address register 4 match
22	Type ID match
21	VLAN tag detected (i.e., type id of 0x8100)
20	Priority tag detected (i.e., type id of 0x8100 and null VLAN identifier)
19:17	VLAN priority (only valid if bit 21 is set)
16	Concatenation format indicator (CFI) bit (only valid if bit 21 is set)
15	End of frame - when set the buffer contains the end of a frame. If end of frame is not set, then the only other valid status are bits 12, 13 and 14.
14	Start of frame - when set the buffer contains the start of a frame. If both bits 15 and 14 are set, then the buffer contains a whole frame.
13:12	Receive buffer offset - indicates the number of bytes by which the data in the first buffer is offset from the word address. Updated with the current values of the network configuration register. If jumbo frame mode is enabled through bit 3 of the network configuration register, then bits 13:12 of the receive buffer descriptor entry are used to indicate bits 13:12 of the frame length.
11:0	Length of frame including FCS (if selected). Bits 13:12 are also used if jumbo frame mode is selected.

To receive frames, the buffer descriptors must be initialized by writing an appropriate address to bits 31 to 2 in the first word of each list entry. Bit zero must be written with zero. Bit one is the wrap bit and indicates the last entry in the list.

The start location of the receive buffer descriptor list must be written to the receive buffer queue pointer register before setting the receive enable bit in the network control register to enable receive. As soon as the receive block starts writing received frame data to the receive FIFO, the receive buffer manager reads the first receive buffer location pointed to by the receive buffer queue pointer register.

If the filter block then indicates that the frame should be copied to memory, the receive data DMA operation starts writing data into the receive buffer. If an error occurs, the buffer is recovered. If the current buffer pointer has its wrap bit set or is the 1024^{th} descriptor, the next receive buffer location is read from the beginning of the receive descriptor list. Otherwise, the next receive buffer location is read from the next word in memory.

There is an 11-bit counter to count out the 2048 word locations of a maximum length, receive buffer descriptor list. This is added with the value originally written to the receive buffer queue pointer register to produce a pointer into the list. A read of the receive buffer queue pointer register returns the pointer value, which is the queue entry currently being accessed. The counter is reset after receive status is written to a descriptor that has its wrap bit set or rolls over to zero after 1024 descriptors have been accessed. The value written to the receive buffer pointer register may be any word-aligned address, provided that there are at least 2048 word locations available between the pointer and the top of the memory.

Section 3.6 of the AMBA 2.0 specification states that bursts should not cross 1K boundaries. As receive buffer manager writes are bursts of two words, to ensure that this does not occur, it is best to write the pointer register with the least three significant bits set to zero. As receive buffers are used, the receive buffer manager sets bit zero of the first word of the descriptor to indicate *used*. If a receive error is detected the receive buffer currently being written is recovered. Previous buffers are not recovered. Software should search through the *used* bits in the buffer descriptors to find out how many frames have been received. It should be checking the start-of-frame and end-of-frame bits, and not rely on the value returned by the receive buffer queue pointer register which changes continuously as more buffers are used.

For CRC errored frames, excessive length frames or length field mismatched frames, all of which are counted in the statistics registers, it is possible that a frame fragment might be stored in a sequence of receive buffers. Software can detect this by looking for start of frame bit set in a buffer following a buffer with no end of frame bit set.

For a properly working Ethernet system, there should be no excessively long frames or frames greater than 128 bytes with CRC/FCS errors. Collision fragments are less than 128 bytes long. Therefore, it is a rare occurrence to find a frame fragment in a receive buffer.

If bit zero is set when the receive buffer manager reads the location of the receive buffer, then the buffer has already been used and cannot be used again until software has processed the frame and cleared bit zero. In this case, the DMA block sets the buffer not available bit in the receive status register and triggers an interrupt.

If bit zero is set when the receive buffer manager reads the location of the receive buffer and a frame is being received, the frame is discarded and the receive resource error statistics register is incremented.

A receive overrun condition occurs when bus was not granted in time or because HRESP was not OK (bus error). In a receive overrun condition, the receive overrun interrupt is asserted and the buffer currently being written is recovered. The next frame received with an address that is recognized reuses the buffer.

If bit 17 of the network configuration register is set, the FCS of received frames shall not be copied to memory. The frame length indicated in the receive status field shall be reduced by four bytes in this case.

43.3.2.3 Transmit Buffer

Frames to be transmitted are stored in one or more transmit buffers. Transmit buffers can be between 0 and 2047 bytes long, so it is possible to transmit frames longer than the maximum length specified in IEEE Standard 802.3. Zero length buffers are allowed. The maximum number of buffers permitted for each transmit frame is 128.

The start location for each transmit buffer is stored in memory in a list of transmit buffer descriptors at a location pointed to by the transmit buffer queue pointer register. Each list entry consists of two words, the first being the byte address of the transmit buffer and the second containing the transmit control and status. Frames can be transmitted with or without automatic CRC generation. If CRC is automatically generated, pad is also automatically generated to take frames to a minimum length of 64 bytes. Table 43-2 on page 852 defines an entry in the transmit buffer descriptor list. To transmit frames, the buffer descriptors must be initialized by writing an appropriate byte address to bits 31 to 0 in the first word of each list entry. The second transmit buffer descriptor is initialized with control information that indicates the length of the buffer, whether or not it is to be transmitted with CRC and whether the buffer is the last buffer in the frame.

After transmission, the control bits are written back to the second word of the first buffer along with the "used" bit and other status information. Bit 31 is the "used" bit which must be zero when the control word is read if transmission is to happen. It is written to one when a frame has been transmitted. Bits 27, 28 and 29 indicate various transmit error conditions. Bit 30 is the "wrap" bit which can be set for any buffer within a frame. If no wrap bit is encountered after 1024 descriptors, the queue pointer rolls over to the start in a similar fashion to the receive queue.

The transmit buffer queue pointer register must not be written while transmit is active. If a new value is written to the transmit buffer queue pointer register, the queue pointer resets itself to point to the beginning of the new queue. If transmit is disabled by writing to bit 3 of the network control, the transmit buffer queue pointer register resets to point to the beginning of the transmit queue. Note that disabling receive does not have the same effect on the receive queue pointer.

Once the transmit queue is initialized, transmit is activated by writing to bit 9, the *Transmit Start* bit of the network control register. Transmit is halted when a buffer descriptor with its *used* bit set is read, or if a transmit error occurs, or by writing to the transmit halt bit of the network control register. (Transmission is suspended if a pause frame is received while the pause enable bit is set in the network configuration register.) Rewriting the start bit while transmission is active is allowed.

Transmission control is implemented with a Tx_go variable which is readable in the transmit status register at bit location 3. The Tx_go variable is reset when:

– transmit is disabled
– a buffer descriptor with its ownership bit set is read
– a new value is written to the transmit buffer queue pointer register
– bit 10, tx_halt, of the network control register is written
– there is a transmit error such as too many retries or a transmit underrun.

To set tx_go, write to bit 9, tx_start, of the network control register. Transmit halt does not take effect until any ongoing transmit finishes. If a collision occurs during transmission of a multi-buffer frame, transmission automatically restarts from the first buffer of the frame. If a "used" bit is read midway through transmission of a multi-buffer frame, this is treated as a transmit error. Transmission stops, tx_er is asserted and the FCS is bad.

If transmission stops due to a transmit error, the transmit queue pointer resets to point to the beginning of the transmit queue. Software needs to re-initialize the transmit queue after a transmit error.

If transmission stops due to a "used" bit being read at the start of the frame, the transmission queue pointer is not reset and transmit starts from the same transmit buffer descriptor when the transmit start bit is written

Table 43-2. Transmit Buffer Descriptor Entry

Bit	Function
Word 0	
31:0	**Byte Address of buffer**
Word 1	

Table 43-2. Transmit Buffer Descriptor Entry (Continued)

Bit	Function
31	Used. Needs to be zero for the EMAC to read data from the transmit buffer. The EMAC sets this to one for the first buffer of a frame once it has been successfully transmitted. Software has to clear this bit before the buffer can be used again. Note: This bit is only set for the first buffer in a frame unlike receive where all buffers have the Used bit set once used.
30	Wrap. Marks last descriptor in transmit buffer descriptor list.
29	Retry limit exceeded, transmit error detected
28	Transmit underrun, occurs either when hresp is not OK (bus error) or the transmit data could not be fetched in time or when buffers are exhausted in mid frame.
27	Buffers exhausted in mid frame
26:17	Reserved
16	No CRC. When set, no CRC is appended to the current frame. This bit only needs to be set for the last buffer of a frame.
15	Last buffer. When set, this bit indicates the last buffer in the current frame has been reached.
14:11	Reserved
10:0	Length of buffer

43.3.3 Transmit Block

This block transmits frames in accordance with the Ethernet IEEE 802.3 CSMA/CD protocol. Frame assembly starts by adding preamble and the start frame delimiter. Data is taken from the transmit FIFO a word at a time. Data is transmitted least significant nibble first. If necessary, padding is added to increase the frame length to 60 bytes. CRC is calculated as a 32-bit polynomial. This is inverted and appended to the end of the frame, taking the frame length to a minimum of 64 bytes. If the No CRC bit is set in the second word of the last buffer descriptor of a transmit frame, neither pad nor CRC are appended.

In full-duplex mode, frames are transmitted immediately. Back-to-back frames are transmitted at least 96 bit times apart to guarantee the interframe gap.

In half-duplex mode, the transmitter checks carrier sense. If asserted, it waits for it to de-assert and then starts transmission after the interframe gap of 96 bit times. If the collision signal is asserted during transmission, the transmitter transmits a jam sequence of 32 bits taken from the data register and then retry transmission after the back off time has elapsed.

The back-off time is based on an XOR of the 10 least significant bits of the data coming from the transmit FIFO and a 10-bit pseudo random number generator. The number of bits used depends on the number of collisions seen. After the first collision, 1 bit is used, after the second 2, and so on up to 10. Above 10, all 10 bits are used. An error is indicated and no further attempts are made if 16 attempts cause collisions.

If transmit DMA underruns, bad CRC is automatically appended using the same mechanism as jam insertion and the tx_er signal is asserted. For a properly configured system, this should never happen.

If the back pressure bit is set in the network control register in half duplex mode, the transmit block transmits 64 bits of data, which can consist of 16 nibbles of 1011 or in bit-rate mode 64 1s, whenever it sees an incoming frame to force a collision. This provides a way of implementing flow control in half-duplex mode.

43.3.4 Pause Frame Support

The start of an 802.3 pause frame is as follows:

Table 43-3. Start of an 802.3 Pause Frame

Destination Address	Source Address	Type (Mac Control Frame)	Pause Opcode	Pause Time
0x0180C2000001	6 bytes	0x8808	0x0001	2 bytes

The network configuration register contains a receive pause enable bit (13). If a valid pause frame is received, the pause time register is updated with the frame's pause time, regardless of its current contents and regardless of the state of the configuration register bit 13. An interrupt (12) is triggered when a pause frame is received, assuming it is enabled in the interrupt mask register. If bit 13 is set in the network configuration register and the value of the pause time register is non-zero, no new frame is transmitted until the pause time register has decremented to zero.

The loading of a new pause time, and hence the pausing of transmission, only occurs when the EMAC is configured for full-duplex operation. If the EMAC is configured for half-duplex, there is no transmission pause, but the pause frame received interrupt is still triggered.

A valid pause frame is defined as having a destination address that matches either the address stored in specific address register 1 or matches 0x0180C2000001 and has the MAC control frame type ID of 0x8808 and the pause opcode of 0x0001. Pause frames that have FCS or other errors are treated as invalid and are discarded. Valid pause frames received increment the Pause Frame Received statistic register.

The pause time register decrements every 512 bit times (i.e., 128 `rx_clks` in nibble mode) once transmission has stopped. For test purposes, the register decrements every `rx_clk` cycle once transmission has stopped if bit 12 (retry test) is set in the network configuration register. If the pause enable bit (13) is not set in the network configuration register, then the decrementing occurs regardless of whether transmission has stopped or not.

An interrupt (13) is asserted whenever the pause time register decrements to zero (assuming it is enabled in the interrupt mask register).

43.3.5 Receive Block

The receive block checks for valid preamble, FCS, alignment and length, presents received frames to the DMA block and stores the frames destination address for use by the address checking block. If, during frame reception, the frame is found to be too long or rx_er is asserted, a bad frame indication is sent to the DMA block. The DMA block then ceases sending data to memory. At the end of frame reception, the receive block indicates to the DMA block whether the frame is good or bad. The DMA block recovers the current receive buffer if the frame was bad. The receive block signals the register block to increment the alignment error, the CRC (FCS) error, the short frame, long frame, jabber error, the receive symbol error statistics and the length field mismatch statistics.

The enable bit for jumbo frames in the network configuration register allows the EMAC to receive jumbo frames of up to 10240 bytes in size. This operation does not form part of the IEEE802.3 specification and is disabled by default. When jumbo frames are enabled, frames received with a frame size greater than 10240 bytes are discarded.

43.3.6 Address Checking Block

The address checking (or filter) block indicates to the DMA block which receive frames should be copied to memory. Whether a frame is copied depends on what is enabled in the network configuration register, the state of the external match pin, the contents of the specific address and hash registers and the frame's destination address. In this implementation of the EMAC, the frame's source address is not checked. Provided that bit 18 of the Network Configuration register is not set, a frame is not copied to memory if the EMAC is transmitting in half duplex mode at the time a destination address is received. If bit 18 of the Network Configuration register is set, frames can be received while transmitting in half-duplex mode.

Ethernet frames are transmitted a byte at a time, least significant bit first. The first six bytes (48 bits) of an Ethernet frame make up the destination address. The first bit of the destination address, the LSB of the first byte of the frame, is the group/individual bit: this is *One* for multicast addresses and *Zero* for unicast. The *All Ones* address is the broadcast address, and a special case of multicast.

The EMAC supports recognition of four specific addresses. Each specific address requires two registers, specific address register bottom and specific address register top. Specific address register bottom stores the first four bytes of the destination address and specific address register top contains the last two bytes. The addresses stored can be specific, group, local or universal.

The destination address of received frames is compared against the data stored in the specific address registers once they have been activated. The addresses are deactivated at reset or when their corresponding specific address register bottom is written. They are activated when specific address register top is written. If a receive frame address matches an active address, the frame is copied to memory.

The following example illustrates the use of the address match registers for a MAC address of 21:43:65:87:A9:CB.

Preamble 55

SFD D5

DA (Octet0 - LSB) 21

DA(Octet 1) 43

DA(Octet 2) 65

DA(Octet 3) 87

DA(Octet 4) A9

DA (Octet5 - MSB) CB

SA (LSB) 00

SA 00

SA 00

SA 00

SA 00

SA (MSB) 43

SA (LSB) 21

The sequence above shows the beginning of an Ethernet frame. Byte order of transmission is from top to bottom as shown. For a successful match to specific address 1, the following address matching registers must be set up:

- Base address + 0x98 0x87654321 (Bottom)
- Base address + 0x9C 0x0000CBA9 (Top)

And for a successful match to the Type ID register, the following should be set up:

- Base address + 0xB8 0x00004321

43.3.7 Broadcast Address

The broadcast address of 0xFFFFFFFFFFFF is recognized if the 'no broadcast' bit in the network configuration register is zero.

43.3.8 Hash Addressing

The hash address register is 64 bits long and takes up two locations in the memory map. The least significant bits are stored in hash register bottom and the most significant bits in hash register top.

The unicast hash enable and the multicast hash enable bits in the network configuration register enable the reception of hash matched frames. The destination address is reduced to a 6-bit index into the 64-bit hash register using the following hash function. The hash function is an *exclusive or* of every sixth bit of the destination address.

hash_index[5] = da[5] ^ da[11] ^ da[17] ^ da[23] ^ da[29] ^ da[35] ^ da[41] ^ da[47]

hash_index[4] = da[4] ^ da[10] ^ da[16] ^ da[22] ^ da[28] ^ da[34] ^ da[40] ^ da[46]

hash_index[3] = da[3] ^ da[09] ^ da[15] ^ da[21] ^ da[27] ^ da[33] ^ da[39] ^ da[45]

hash_index[2] = da[2] ^ da[08] ^ da[14] ^ da[20] ^ da[26] ^ da[32] ^ da[38] ^ da[44]

hash_index[1] = da[1] ^ da[07] ^ da[13] ^ da[19] ^ da[25] ^ da[31] ^ da[37] ^ da[43]

hash_index[0] = da[0] ^ da[06] ^ da[12] ^ da[18] ^ da[24] ^ da[30] ^ da[36] ^ da[42]

`da[0]` represents the least significant bit of the first byte received, that is, the multicast/unicast indicator, and `da[47]` represents the most significant bit of the last byte received.

If the hash index points to a bit that is set in the hash register, then the frame is matched according to whether the frame is multicast or unicast.

A multicast match is signalled if the multicast hash enable bit is set. da[0] is 1 and the hash index points to a bit set in the hash register.

A unicast match is signalled if the unicast hash enable bit is set. da[0] is 0 and the hash index points to a bit set in the hash register.

To receive all multicast frames, the hash register should be set with all ones and the multicast hash enable bit should be set in the network configuration register.

43.3.9 Copy All Frames (or Promiscuous Mode)

If the copy all frames bit is set in the network configuration register, then all non-errored frames are copied to memory. For example, frames that are too long, too short, or have FCS errors or rx_er asserted during reception are discarded and all others are received. Frames with FCS errors are copied to memory if bit 19 in the network configuration register is set.

43.3.10 Type ID Checking

The contents of the type_id register are compared against the length/type ID of received frames (i.e., bytes 13 and 14). Bit 22 in the receive buffer descriptor status is set if there is a match. The reset state of this register is zero which is unlikely to match the length/type ID of any valid Ethernet frame.

Note: A type ID match does not affect whether a frame is copied to memory.

43.3.11 VLAN Support

An Ethernet encoded 802.1Q VLAN tag looks like this:

Table 43-4. 802.1Q VLAN Tag

TPID (Tag Protocol Identifier) 16 bits	TCI (Tag Control Information) 16 bits
0x8100	First 3 bits priority, then CFI bit, last 12 bits VID

The VLAN tag is inserted at the 13th byte of the frame, adding an extra four bytes to the frame. If the VID (VLAN identifier) is null (0x000), this indicates a priority-tagged frame. The MAC can support frame lengths up to 1536 bytes, 18 bytes more than the original Ethernet maximum frame length of 1518 bytes. This is achieved by setting bit 8 in the network configuration register.

The following bits in the receive buffer descriptor status word give information about VLAN tagged frames:

- Bit 21 set if receive frame is VLAN tagged (i.e. type id of 0x8100)
- Bit 20 set if receive frame is priority tagged (i.e. type id of 0x8100 and null VID). (If bit 20 is set bit 21 is set also.)
- Bit 19, 18 and 17 set to priority if bit 21 is set
- Bit 16 set to CFI if bit 21 is set

43.3.12 Wake-on-LAN Support

The receive block supports Wake-on-LAN by detecting the following events on incoming receive frames:

- Magic packet
- ARP request to the device IP address
- Specific address 1 filter match
- Multicast hash filter match

If one of these events occurs Wake-on-LAN detection is indicated by asserting the `wol` output pin for 64 `rx_clk` cycles. These events can be individually enabled through bits[19:16] of the Wake-on-LAN register. Also, for Wake-on-LAN detection to occur, receive enable must be set in the network control register, however a receive buffer does not have to be available. `wol` assertion due to ARP request, specific address 1 or multicast filter events occurs even if the frame is errored. For magic packet events, the frame must be correctly formed and error free.

A magic packet event is detected if all of the following are true:

- magic packet events are enabled through bit 16 of the Wake-on-LAN register
- the frame's destination address matches specific address 1

- the frame is correctly formed with no errors
- the frame contains at least 6 bytes of 0xFF for synchronization
- there are 16 repetitions of the contents of specific address 1 register immediately following the synchronization

An ARP request event is detected if all of the following are true:

- ARP request events are enabled through bit 17 of the Wake-on-LAN register
- broadcasts are allowed by bit 5 in the network configuration register
- the frame has a broadcast destination address (bytes 1 to 6)
- the frame has a type ID field of 0x0806 (bytes 13 and 14)
- the frame has an ARP operation field of 0x0001 (bytes 21 and 22)
- the least significant 16 bits of the frame's ARP target protocol address (bytes 41 and 42) match the value programmed in bits[15:0] of the Wake-on-LAN register

The decoding of the ARP fields adjusts automatically if a VLAN tag is detected within the frame. The reserved value of 0x0000 for the Wake-on-LAN target address value does not cause an ARP request event, even if matched by the frame.

A specific address 1 filter match event occurs if all of the following are true:

- specific address 1 events are enabled through bit 18 of the Wake-on-LAN register
- the frame's destination address matches the value programmed in the specific address 1 registers

A multicast filter match event occurs if all of the following are true:

- multicast hash events are enabled through bit 19 of the Wake-on-LAN register
- multicast hash filtering is enabled through bit 6 of the network configuration register
- the frame's destination address matches against the multicast hash filter
- the frame's destination address is not a broadcast

43.3.13 PHY Maintenance

The register EMAC_MAN enables the EMAC to communicate with a PHY by means of the MDIO interface. It is used during auto-negotiation to ensure that the EMAC and the PHY are configured for the same speed and duplex configuration.

The PHY maintenance register is implemented as a shift register. Writing to the register starts a shift operation which is signalled as complete when bit two is set in the network status register (about 2000 MCK cycles later when bit ten is set to zero, and bit eleven is set to one in the network configuration register). An interrupt is generated as this bit is set. During this time, the MSB of the register is output on the MDIO pin and the LSB updated from the MDIO pin with each MDC cycle. This causes transmission of a PHY management frame on MDIO.

Reading during the shift operation returns the current contents of the shift register. At the end of management operation, the bits have shifted back to their original locations. For a read operation, the data bits are updated with data read from the PHY. It is important to write the correct values to the register to ensure a valid PHY management frame is produced.

The MDIO interface can read IEEE 802.3 clause 45 PHYs as well as clause 22 PHYs. To read clause 45 PHYs, bits[31:28] should be written as 0x0011. For a description of MDC generation, see the network configuration register in the "Network Control Register" on page 865.

43.3.14 Media Independent Interface

The Ethernet MAC is capable of interfacing to both RMII and MII Interfaces. The RMII bit in the EMAC_USRIO register controls the interface that is selected. When this bit is set, the RMII interface is selected, else the MII interface is selected.

The MII and RMII interface are capable of both 10Mb/s and 100Mb/s data rates as described in the IEEE 802.3u standard. The signals used by the MII and RMII interfaces are described in Table 43-5.

Table 43-5. Pin Configuration

Pin Name	MII	RMII
ETXCK_EREFCK	ETXCK: Transmit Clock	EREFCK: Reference Clock
ECRS	ECRS: Carrier Sense	
ECOL	ECOL: Collision Detect	
ERXDV	ERXDV: Data Valid	ECRSDV: Carrier Sense/Data Valid
ERX0 - ERX3	ERX0 - ERX3: 4-bit Receive Data	ERX0 - ERX1: 2-bit Receive Data
ERXER	ERXER: Receive Error	ERXER: Receive Error
ERXCK	ERXCK: Receive Clock	
ETXEN	ETXEN: Transmit Enable	ETXEN: Transmit Enable
ETX0-ETX3	ETX0 - ETX3: 4-bit Transmit Data	ETX0 - ETX1: 2-bit Transmit Data
ETXER	ETXER: Transmit Error	

The intent of the RMII is to provide a reduced pin count alternative to the IEEE 802.3u MII. It uses 2 bits for transmit (ETX0 and ETX1) and two bits for receive (ERX0 and ERX1). There is a Transmit Enable (ETXEN), a Receive Error (ERXER), a Carrier Sense (ECRS_DV), and a 50 MHz Reference Clock (ETXCK_EREFCK) for 100Mb/s data rate.

43.3.14.1 RMII Transmit and Receive Operation

The same signals are used internally for both the RMII and the MII operations. The RMII maps these signals in a more pin-efficient manner. The transmit and receive bits are converted from a 4-bit parallel format to a 2-bit parallel scheme that is clocked at twice the rate. The carrier sense and data valid signals are combined into the ECRSDV signal. This signal contains information on carrier sense, FIFO status, and validity of the data. Transmit error bit (ETXER) and collision detect (ECOL) are not used in RMII mode.

43.4 Programming Interface

43.4.1 Initialization

43.4.1.1 Configuration

Initialization of the EMAC configuration (e.g., loop-back mode, frequency ratios) must be done while the transmit and receive circuits are disabled. See the description of the network control register and network configuration register earlier in this document.

To change loop-back mode, the following sequence of operations must be followed:

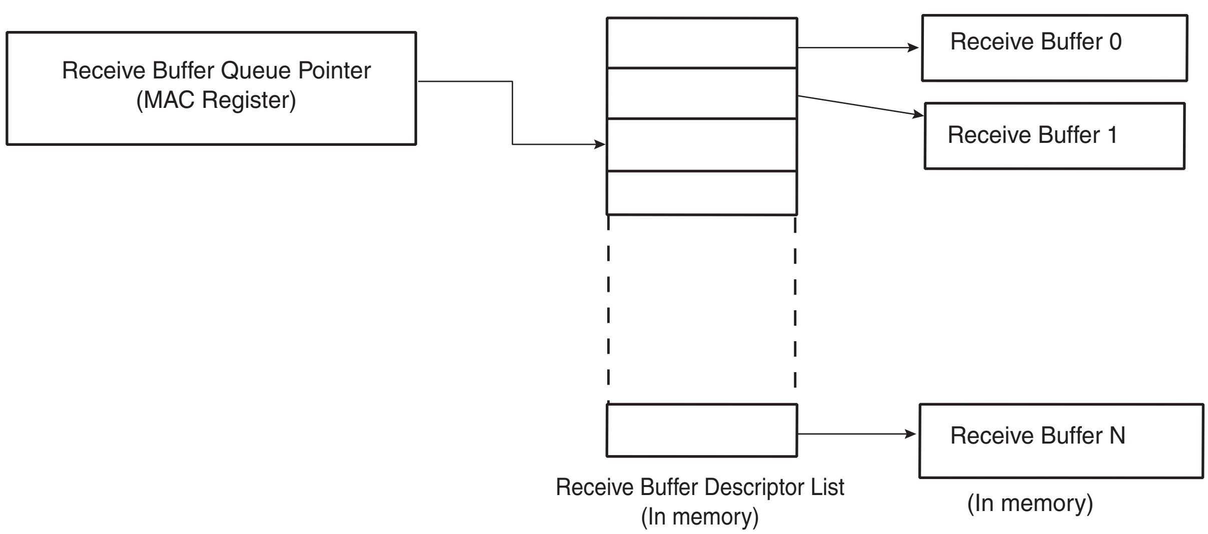
1. Write to network control register to disable transmit and receive circuits.
2. Write to network control register to change loop-back mode.
3. Write to network control register to re-enable transmit or receive circuits.

Note: These writes to network control register cannot be combined in any way.

43.4.1.2 Receive Buffer List

Receive data is written to areas of data (i.e., buffers) in system memory. These buffers are listed in another data structure that also resides in main memory. This data structure (receive buffer queue) is a sequence of descriptor entries as defined in "Receive Buffer Descriptor Entry" on page 849. It points to this data structure.

Figure 43-2. Receive Buffer List



To create the list of buffers:

1. Allocate a number (*n*) of buffers of 128 bytes in system memory.
2. Allocate an area 2*n* words for the receive buffer descriptor entry in system memory and create *n* entries in this list. Mark all entries in this list as owned by EMAC, i.e., bit 0 of word 0 set to 0.
3. If less than 1024 buffers are defined, the last descriptor must be marked with the wrap bit (bit 1 in word 0 set to 1).
4. Write address of receive buffer descriptor entry to EMAC register `receive_buffer` queue pointer.
5. The receive circuits can then be enabled by writing to the address recognition registers and then to the network control register.

43.4.1.3 Transmit Buffer List

Transmit data is read from areas of data (the buffers) in system memory These buffers are listed in another data structure that also resides in main memory. This data structure (Transmit Buffer Queue) is a sequence of descriptor entries (as defined in Table 43-2 on page 852) that points to this data structure.

To create this list of buffers:

1. Allocate a number (*n*) of buffers of between 1 and 2047 bytes of data to be transmitted in system memory. Up to 128 buffers per frame are allowed.
2. Allocate an area 2*n* words for the transmit buffer descriptor entry in system memory and create N entries in this list. Mark all entries in this list as owned by EMAC, i.e. bit 31 of word 1 set to 0.
3. If fewer than 1024 buffers are defined, the last descriptor must be marked with the wrap bit — bit 30 in word 1 set to 1.
4. Write address of transmit buffer descriptor entry to EMAC register transmit_buffer queue pointer.
5. The transmit circuits can then be enabled by writing to the network control register.

43.4.1.4 Address Matching

The EMAC register-pair hash address and the four specific address register-pairs must be written with the required values. Each register-pair comprises a bottom register and top register, with the bottom register being written first. The address matching is disabled for a particular register-pair after the bottom-register has been written and re-enabled when the top register is written. See “Address Checking Block” on page 855. for details of address matching. Each register-pair may be written at any time, regardless of whether the receive circuits are enabled or disabled.

43.4.1.5 Interrupts

There are 15 interrupt conditions that are detected within the EMAC. These are ORed to make a single interrupt. Depending on the overall system design, this may be passed through a further level of interrupt collection (interrupt controller). On receipt of the interrupt signal, the CPU enters the interrupt handler (Refer to the AIC programmer datasheet). To ascertain which interrupt has been generated, read the interrupt status register. Note that this register clears itself when read. At reset, all interrupts are disabled. To enable an interrupt, write to interrupt enable register with the pertinent interrupt bit set to 1. To disable an interrupt, write to interrupt disable register with the pertinent interrupt bit set to 1. To check whether an interrupt is enabled or disabled, read interrupt mask register: if the bit is set to 1, the interrupt is disabled.

43.4.1.6 Transmitting Frames

To set up a frame for transmission:

1. Enable transmit in the network control register.
2. Allocate an area of system memory for transmit data. This does not have to be contiguous, varying byte lengths can be used as long as they conclude on byte borders.
3. Set-up the transmit buffer list.
4. Set the network control register to enable transmission and enable interrupts.
5. Write data for transmission into these buffers.
6. Write the address to transmit buffer descriptor queue pointer.
7. Write control and length to word one of the transmit buffer descriptor entry.

8. Write to the transmit start bit in the network control register.

43.4.1.7 Receiving Frames

When a frame is received and the receive circuits are enabled, the EMAC checks the address and, in the following cases, the frame is written to system memory:

- if it matches one of the four specific address registers.
- if it matches the hash address function.
- if it is a broadcast address (0xFFFFFFFFFFFF) and broadcasts are allowed.
- if the EMAC is configured to copy all frames.

The register receive buffer queue pointer points to the next entry (see Table 43-1 on page 849) and the EMAC uses this as the address in system memory to write the frame to. Once the frame has been completely and successfully received and written to system memory, the EMAC then updates the receive buffer descriptor entry with the reason for the address match and marks the area as being owned by software. Once this is complete an interrupt receive complete is set. Software is then responsible for handling the data in the buffer and then releasing the buffer by writing the ownership bit back to 0.

If the EMAC is unable to write the data at a rate to match the incoming frame, then an interrupt receive overrun is set. If there is no receive buffer available, i.e., the next buffer is still owned by software, the interrupt receive buffer not available is set. If the frame is not successfully received, a statistic register is incremented and the frame is discarded without informing software.

43.5 Ethernet MAC 10/100 (EMAC) User Interface

Table 43-6. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Network Control Register	EMAC_NCR	Read-write	0
0x04	Network Configuration Register	EMAC_NCFG	Read-write	0x800
0x08	Network Status Register	EMAC_NSR	Read-only	-
0x0C	Reserved			
0x10	Reserved			
0x14	Transmit Status Register	EMAC_TSR	Read-write	0x0000_0000
0x18	Receive Buffer Queue Pointer Register	EMAC_RBQP	Read-write	0x0000_0000
0x1C	Transmit Buffer Queue Pointer Register	EMAC_TBQP	Read-write	0x0000_0000
0x20	Receive Status Register	EMAC_RSR	Read-write	0x0000_0000
0x24	Interrupt Status Register	EMAC_ISR	Read-write	0x0000_0000
0x28	Interrupt Enable Register	EMAC_IER	Write-only	-
0x2C	Interrupt Disable Register	EMAC_IDR	Write-only	-
0x30	Interrupt Mask Register	EMAC_IMR	Read-only	0x0000_7FFF
0x34	Phy Maintenance Register	EMAC_MAN	Read-write	0x0000_0000
0x38	Pause Time Register	EMAC_PTR	Read-write	0x0000_0000
0x3C	Pause Frames Received Register	EMAC_PFR	Read-write	0x0000_0000
0x40	Frames Transmitted Ok Register	EMAC_FTO	Read-write	0x0000_0000
0x44	Single Collision Frames Register	EMAC_SCF	Read-write	0x0000_0000
0x48	Multiple Collision Frames Register	EMAC_MCF	Read-write	0x0000_0000
0x4C	Frames Received Ok Register	EMAC_FRO	Read-write	0x0000_0000
0x50	Frame Check Sequence Errors Register	EMAC_FCSE	Read-write	0x0000_0000
0x54	Alignment Errors Register	EMAC_ALE	Read-write	0x0000_0000
0x58	Deferred Transmission Frames Register	EMAC_DTF	Read-write	0x0000_0000
0x5C	Late Collisions Register	EMAC_LCOL	Read-write	0x0000_0000
0x60	Excessive Collisions Register	EMAC_ECOL	Read-write	0x0000_0000
0x64	Transmit Underrun Errors Register	EMAC_TUND	Read-write	0x0000_0000
0x68	Carrier Sense Errors Register	EMAC_CSE	Read-write	0x0000_0000
0x6C	Receive Resource Errors Register	EMAC_RRE	Read-write	0x0000_0000
0x70	Receive Overrun Errors Register	EMAC_ROV	Read-write	0x0000_0000
0x74	Receive Symbol Errors Register	EMAC_RSE	Read-write	0x0000_0000
0x78	Excessive Length Errors Register	EMAC_ELE	Read-write	0x0000_0000
0x7C	Receive Jabbers Register	EMAC_RJA	Read-write	0x0000_0000
0x80	Undersize Frames Register	EMAC_USF	Read-write	0x0000_0000
0x84	SQE Test Errors Register	EMAC_STE	Read-write	0x0000_0000
0x88	Received Length Field Mismatch Register	EMAC_RLE	Read-write	0x0000_0000

Table 43-6. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0x90	Hash Register Bottom [31:0] Register	EMAC_HRB	Read-write	0x0000_0000
0x94	Hash Register Top [63:32] Register	EMAC_HRT	Read-write	0x0000_0000
0x98	Specific Address 1 Bottom Register	EMAC_SA1B	Read-write	0x0000_0000
0x9C	Specific Address 1 Top Register	EMAC_SA1T	Read-write	0x0000_0000
0xA0	Specific Address 2 Bottom Register	EMAC_SA2B	Read-write	0x0000_0000
0xA4	Specific Address 2 Top Register	EMAC_SA2T	Read-write	0x0000_0000
0xA8	Specific Address 3 Bottom Register	EMAC_SA3B	Read-write	0x0000_0000
0xAC	Specific Address 3 Top Register	EMAC_SA3T	Read-write	0x0000_0000
0xB0	Specific Address 4 Bottom Register	EMAC_SA4B	Read-write	0x0000_0000
0xB4	Specific Address 4 Top Register	EMAC_SA4T	Read-write	0x0000_0000
0xB8	Type ID Checking Register	EMAC_TID	Read-write	0x0000_0000
0xC0	User Input/Output Register	EMAC_USRIO	Read-write	0x0000_0000
0xC4	Wake on LAN Register	EMAC_WOL	Read-write	0x0000_0000
0xC8 - 0xFC	Reserved	–	–	–

43.5.1 Network Control Register

Name: EMAC_NCR

Address: 0xFFFBC000

Access: 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
–	–	–	–	–	THALT	TSTART	BP

7	6	5	4	3	2	1	0
WESTAT	INCSTAT	CLRSTAT	MPE	TE	RE	LLB	LB

- **LB: LoopBack**

Asserts the loopback signal to the PHY.

- **LLB: Loopback local**

Connects `txd` to `rxd`, `tx_en` to `rx_dv`, forces full duplex and drives `rx_clk` and `tx_clk` with `pclk` divided by 4. `rx_clk` and `tx_clk` may glitch as the EMAC is switched into and out of internal loop back. It is important that receive and transmit circuits have already been disabled when making the switch into and out of internal loop back.

- **RE: Receive enable**

When set, enables the EMAC to receive data. When reset, frame reception stops immediately and the receive FIFO is cleared. The receive queue pointer register is unaffected.

- **TE: Transmit enable**

When set, enables the Ethernet transmitter to send data. When reset transmission, stops immediately, the transmit FIFO and control registers are cleared and the transmit queue pointer register resets to point to the start of the transmit descriptor list.

- **MPE: Management port enable**

Set to one to enable the management port. When zero, forces MDIO to high impedance state and MDC low.

- **CLRSTAT: Clear statistics registers**

This bit is write only. Writing a one clears the statistics registers.

- **INCSTAT: Increment statistics registers**

This bit is write only. Writing a one increments all the statistics registers by one for test purposes.

- **WESTAT: Write enable for statistics registers**

Setting this bit to one makes the statistics registers writable for functional test purposes.

- **BP: Back pressure**

If set in half duplex mode, forces collisions on all received frames.

- **TSTART: Start transmission**

Writing one to this bit starts transmission.

- **THALT: Transmit halt**

Writing one to this bit halts transmission as soon as any ongoing frame transmission ends.

43.5.2 Network Configuration Register

Name: EMAC_NCFG

Address: 0xFFFBC004

Access: Read-write

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

23	22	21	20	19	18	17	16
–	–	–	–	IRXFCS	EFRHD	DRFCS	RLCE

15	14	13	12	11	10	9	8
RBOF		PAE	RTY	CLK		–	BIG

7	6	5	4	3	2	1	0
UNI	MTI	NBC	CAF	JFRAME	–	FD	SPD

- **SPD: Speed**

Set to 1 to indicate 100 Mbit/s operation, 0 for 10 Mbit/s. The value of this pin is reflected on the `speed` pin.

- **FD: Full Duplex**

If set to 1, the transmit block ignores the state of collision and carrier sense and allows receive while transmitting. Also controls the `half_duplex` pin.

- **CAF: Copy All Frames**

When set to 1, all valid frames are received.

- **JFRAME: Jumbo Frames**

Set to one to enable jumbo frames of up to 10240 bytes to be accepted.

- **NBC: No Broadcast**

When set to 1, frames addressed to the broadcast address of all ones are not received.

- **MTI: Multicast Hash Enable**

When set, multicast frames are received when the 6-bit hash function of the destination address points to a bit that is set in the hash register.

- **UNI: Unicast Hash Enable**

When set, unicast frames are received when the 6-bit hash function of the destination address points to a bit that is set in the hash register.

- **BIG: Receive 1536 bytes frames**

Setting this bit means the EMAC receives frames up to 1536 bytes in length. Normally, the EMAC would reject any frame above 1518 bytes.

- **CLK: MDC Clock Divider**

Set according to system clock speed. This determines by what number system clock is divided to generate MDC. For conformance with 802.3, MDC must not exceed 2.5MHz (MDC is only active during MDIO read and write operations).

CLK	MDC
00	MCK divided by 8 (MCK up to 20 MHz)
01	MCK divided by 16 (MCK up to 40 MHz)
10	MCK divided by 32 (MCK up to 80 MHz)
11	MCK divided by 64 (MCK up to 160 MHz)

- **RTY: Retry Test**

Must be set to zero for normal operation. If set to one, the back off between collisions is always one slot time. Setting this bit to one helps testing the too many retries condition. Also used in the pause frame tests to reduce the pause counters decrement time from 512 bit times, to every `rx_clk` cycle.

- **PAE: Pause Enable**

When set, transmission pauses when a valid pause frame is received.

- **RBOF: Receive Buffer Offset**

Indicates the number of bytes by which the received data is offset from the start of the first receive buffer.

RBOF	Offset
00	No offset from start of receive buffer
01	One-byte offset from start of receive buffer
10	Two-byte offset from start of receive buffer
11	Three-byte offset from start of receive buffer

- **RLCE: Receive Length field Checking Enable**

When set, frames with measured lengths shorter than their length fields are discarded. Frames containing a type ID in bytes 13 and 14 — length/type ID = 0600 — are not be counted as length errors.

- **DRFCS: Discard Receive FCS**

When set, the FCS field of received frames are not be copied to memory.

- **EFRHD:**

Enable Frames to be received in half-duplex mode while transmitting.

- **IRXFCS: Ignore RX FCS**

When set, frames with FCS/CRC errors are not rejected and no FCS error statistics are counted. For normal operation, this bit must be set to 0.

43.5.3 Network Status Register

Name: EMAC_NSR

Address: 0xFFFBC008

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	IDLE	MDIO	–

- **MDIO**

Returns status of the mdio_in pin. Use the PHY maintenance register for reading managed frames rather than this bit.

- **IDLE**

0 = The PHY logic is running.

1 = The PHY management logic is idle (i.e., has completed).

43.5.4 Transmit Status Register

Name: EMAC_TSR

Address: 0xFFFBC014

Access: Read-write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	UND	COMP	BEX	TGO	RLE	COL	UBR

This register, when read, provides details of the status of a transmit. Once read, individual bits may be cleared by writing 1 to them. It is not possible to set a bit to 1 by writing to the register.

- **UBR: Used Bit Read**

Set when a transmit buffer descriptor is read with its used bit set. Cleared by writing a one to this bit.

- **COL: Collision Occurred**

Set by the assertion of collision. Cleared by writing a one to this bit.

- **RLE: Retry Limit exceeded**

Cleared by writing a one to this bit.

- **TGO: Transmit Go**

If high transmit is active.

- **BEX: Buffers exhausted mid frame**

If the buffers run out during transmission of a frame, then transmission stops, FCS shall be bad and tx_er asserted. Cleared by writing a one to this bit.

- **COMP: Transmit Complete**

Set when a frame has been transmitted. Cleared by writing a one to this bit.

- **UND: Transmit Underrun**

Set when transmit DMA was not able to read data from memory, either because the bus was not granted in time, because a not OK `hresp(bus error)` was returned or because a used bit was read midway through frame transmission. If this occurs, the transmitter forces `bad` CRC. Cleared by writing a one to this bit.

43.5.5 Receive Buffer Queue Pointer Register

Name: EMAC_RBQP

Address: 0xFFFBC018

Access: Read-write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR						–	–

This register points to the entry in the receive buffer queue (descriptor list) currently being used. It is written with the start location of the receive buffer descriptor list. The lower order bits increment as buffers are used up and wrap to their original values after either 1024 buffers or when the wrap bit of the entry is set.

Reading this register returns the location of the descriptor currently being accessed. This value increments as buffers are used. Software should not use this register for determining where to remove received frames from the queue as it constantly changes as new frames are received. Software should instead work its way through the buffer descriptor queue checking the used bits.

Receive buffer writes also comprise bursts of two words and, as with transmit buffer reads, it is recommended that bit 2 is always written with zero to prevent a burst crossing a 1K boundary, in violation of section 3.6 of the AMBA specification.

- **ADDR: Receive Buffer Queue Pointer Address**

Written with the address of the start of the receive queue, reads as a pointer to the current buffer being used.

43.5.6 Transmit Buffer Queue Pointer Register

Name: EMAC_TBQP

Address: 0xFFFBC01C

Access: Read-write

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

This register points to the entry in the transmit buffer queue (descriptor list) currently being used. It is written with the start location of the transmit buffer descriptor list. The lower order bits increment as buffers are used up and wrap to their original values after either 1024 buffers or when the wrap bit of the entry is set. This register can only be written when bit 3 in the transmit status register is low.

As transmit buffer reads consist of bursts of two words, it is recommended that bit 2 is always written with zero to prevent a burst crossing a 1K boundary, in violation of section 3.6 of the AMBA specification.

- **ADDR: Transmit Buffer Queue Pointer Address**

Written with the address of the start of the transmit queue, reads as a pointer to the first buffer of the frame being transmitted or about to be transmitted.

43.5.7 Receive Status Register

Name: EMAC_RSR

Address: 0xFFFBC020

Access: Read-write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	OVR	REC	BNA

This register, when read, provides details of the status of a receive. Once read, individual bits may be cleared by writing 1 to them. It is not possible to set a bit to 1 by writing to the register.

- **BNA: Buffer Not Available**

An attempt was made to get a new buffer and the pointer indicated that it was owned by the processor. The DMA rereads the pointer each time a new frame starts until a valid pointer is found. This bit is set at each attempt that fails even if it has not had a successful pointer read since it has been cleared.

Cleared by writing a one to this bit.

- **REC: Frame Received**

One or more frames have been received and placed in memory. Cleared by writing a one to this bit.

- **OVR: Receive Overrun**

The DMA block was unable to store the receive frame to memory, either because the bus was not granted in time or because a not OK `hresp(bus error)` was returned. The buffer is recovered if this happens.

Cleared by writing a one to this bit.

43.5.8 Interrupt Status Register

Name: EMAC_ISR

Address: 0xFFFBC024

Access: 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
–	WOL	PTZ	PFR	HRESP	ROVR	–	–

7	6	5	4	3	2	1	0
TCOMP	TXERR	RLE	TUND	TXUBR	RXUBR	RCOMP	MFD

- **MFD: Management Frame Done**

The PHY maintenance register has completed its operation. Cleared on read.

- **RCOMP: Receive Complete**

A frame has been stored in memory. Cleared on read.

- **RXUBR: Receive Used Bit Read**

Set when a receive buffer descriptor is read with its used bit set. Cleared on read.

- **TXUBR: Transmit Used Bit Read**

Set when a transmit buffer descriptor is read with its used bit set. Cleared on read.

- **TUND: Ethernet Transmit Buffer Underrun**

The transmit DMA did not fetch frame data in time for it to be transmitted or `hresp` returned not OK. Also set if a used bit is read mid-frame or when a new transmit queue pointer is written. Cleared on read.

- **RLE: Retry Limit Exceeded**

Cleared on read.

- **TXERR: Transmit Error**

Transmit buffers exhausted in mid-frame - transmit error. Cleared on read.

- **TCOMP: Transmit Complete**

Set when a frame has been transmitted. Cleared on read.

- **ROVR: Receive Overrun**

Set when the receive overrun status bit gets set. Cleared on read.

- **HRESP: Hresp not OK**

Set when the DMA block sees `a bus error`. Cleared on read.

- **PFR: Pause Frame Received**

Indicates a valid pause has been received. Cleared on a read.

- **PTZ: Pause Time Zero**

Set when the pause time register, 0x38 decrements to zero. Cleared on a read.

- **WOL: Wake On LAN**

Set when a WOL event has been triggered (This flag can be set even if the EMAC is not clocked). Cleared on a read.

43.5.9 Interrupt Enable Register

Name: EMAC_IER

Address: 0xFFFBC028

Access: Write-only

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

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

15	14	13	12	11	10	9	8
–	WOL	PTZ	PFR	HRESP	ROVR	–	–

7	6	5	4	3	2	1	0
TCOMP	TXERR	RLE	TUND	TXUBR	RXUBR	RCOMP	MFD

- **MFD: Management Frame sent**

Enable management done interrupt.

- **RCOMP: Receive Complete**

Enable receive complete interrupt.

- **RXUBR: Receive Used Bit Read**

Enable receive used bit read interrupt.

- **TXUBR: Transmit Used Bit Read**

Enable transmit used bit read interrupt.

- **TUND: Ethernet Transmit Buffer Underrun**

Enable transmit underrun interrupt.

- **RLE: Retry Limit Exceeded**

Enable retry limit exceeded interrupt.

- **TXERR**

Enable transmit buffers exhausted in mid-frame interrupt.

- **TCOMP: Transmit Complete**

Enable transmit complete interrupt.

- **ROVR: Receive Overrun**

Enable receive overrun interrupt.

- **HRESP: Hresp not OK**

Enable Hresp not OK interrupt.

- **PFR: Pause Frame Received**

Enable pause frame received interrupt.

- **PTZ: Pause Time Zero**

Enable pause time zero interrupt.

- **WOL: Wake On LAN**

Enable Wake On LAN interrupt.

43.5.10 Interrupt Disable Register

Name: EMAC_IDR

Address: 0xFFFBC02C

Access: Write-only

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

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

15	14	13	12	11	10	9	8
–	WOL	PTZ	PFR	HRESP	ROVR	–	–

7	6	5	4	3	2	1	0
TCOMP	TXERR	RLE	TUND	TXUBR	RXUBR	RCOMP	MFD

- **MFD: Management Frame sent**

Disable management done interrupt.

- **RCOMP: Receive Complete**

Disable receive complete interrupt.

- **RXUBR: Receive Used Bit Read**

Disable receive used bit read interrupt.

- **TXUBR: Transmit Used Bit Read**

Disable transmit used bit read interrupt.

- **TUND: Ethernet Transmit Buffer Underrun**

Disable transmit underrun interrupt.

- **RLE: Retry Limit Exceeded**

Disable retry limit exceeded interrupt.

- **TXERR**

Disable transmit buffers exhausted in mid-frame interrupt.

- **TCOMP: Transmit Complete**

Disable transmit complete interrupt.

- **ROVR: Receive Overrun**

Disable receive overrun interrupt.

- **HRESP: Hresp not OK**

Disable Hresp not OK interrupt.

- **PFR: Pause Frame Received**

Disable pause frame received interrupt.

- **PTZ: Pause Time Zero**

Disable pause time zero interrupt.

- **WOL: Wake On LAN**

Disable Wake On LAN interrupt.

43.5.11 Interrupt Mask Register

Name: EMAC_IMR

Address: 0xFFFBC030

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	WOL	PTZ	PFR	HRESP	ROVR	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TCOMP	TXERR	RLE	TUND	TXUBR	RXUBR	RCOMP	MFD

- **MFD: Management Frame sent**

Management done interrupt masked.

- **RCOMP: Receive Complete**

Receive complete interrupt masked.

- **RXUBR: Receive Used Bit Read**

Receive used bit read interrupt masked.

- **TXUBR: Transmit Used Bit Read**

Transmit used bit read interrupt masked.

- **TUND: Ethernet Transmit Buffer Underrun**

Transmit underrun interrupt masked.

- **RLE: Retry Limit Exceeded**

Retry limit exceeded interrupt masked.

- **TXERR**

Transmit buffers exhausted in mid-frame interrupt masked.

- **TCOMP: Transmit Complete**

Transmit complete interrupt masked.

- **ROVR: Receive Overrun**

Receive overrun interrupt masked.

- **HRESP: Hresp not OK**

Hresp not OK interrupt masked.

- **PFR: Pause Frame Received**

Pause frame received interrupt masked.

- **PTZ: Pause Time Zero**

Pause time zero interrupt masked.

- **WOL: Wake On LAN**

Wake On LAN interrupt masked.

43.5.12 PHY Maintenance Register

Name: EMAC_MAN

Address: 0xFFFBC034

Access: Read-write

31	30	29	28	27	26	25	24
SOF		RW		PHYA			

23	22	21	20	19	18	17	16
PHYA	REGA					CODE	

15	14	13	12	11	10	9	8
DATA							

7	6	5	4	3	2	1	0
DATA							

- **DATA**

For a write operation this is written with the data to be written to the PHY.

After a read operation this contains the data read from the PHY.

- **CODE:**

Must be written to 10. Reads as written.

- **REGA: Register Address**

Specifies the register in the PHY to access.

- **PHYA: PHY Address**

- **RW: Read-write**

10 is read; 01 is write. Any other value is an invalid PHY management frame

- **SOF: Start of Frame**

Must be written 01 for a valid frame.

43.5.13 Pause Time Register

Name: EMAC_PTR

Address: 0xFFFBC038

Access: 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
PTIME							

7	6	5	4	3	2	1	0
PTIME							

- **PTIME: Pause Time**

Stores the current value of the pause time register which is decremented every 512 bit times.

43.5.14 Hash Register Bottom

Name: EMAC_HRB

Address: 0xFFFBC090

Access: Read-write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR:**

Bits 31:0 of the hash address register. See "Hash Addressing" on page 856.

43.5.15 Hash Register Top

Name: EMAC_HRT

Address: 0xFFFBC094

Access: Read-write

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

- **ADDR:**

Bits 63:32 of the hash address register. See "Hash Addressing" on page 856.

43.5.16 Specific Address 1 Bottom Register

Name: EMAC_SA1B

Address: 0xFFFBC098

Access: Read-write

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

- **ADDR**

Least significant bits of the destination address. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

43.5.17 Specific Address 1 Top Register

Name: EMAC_SA1T

Address: 0xFFFBC09C

Access: Read-write

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

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

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR**

The most significant bits of the destination address, that is bits 47 to 32.

43.5.18 Specific Address 2 Bottom Register

Name: EMAC_SA2B

Address: 0xFFFBC0A0

Access: Read-write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR**

Least significant bits of the destination address. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

43.5.19 Specific Address 2 Top Register

Name: EMAC_SA2T

Address: 0xFFFBC0A4

Access: Read-write

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

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

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR**

The most significant bits of the destination address, that is bits 47 to 32.

43.5.20 Specific Address 3 Bottom Register

Name: EMAC_SA3B

Address: 0xFFFBC0A8

Access: Read-write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR**

Least significant bits of the destination address. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

43.5.21 Specific Address 3 Top Register

Name: EMAC_SA3T

Address: 0xFFFBC0AC

Access: Read-write

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

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

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR**

The most significant bits of the destination address, that is bits 47 to 32.

43.5.22 Specific Address 4 Bottom Register

Name: EMAC_SA4B

Address: 0xFFFBC0B0

Access: Read-write

31	30	29	28	27	26	25	24
ADDR							

23	22	21	20	19	18	17	16
ADDR							

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR**

Least significant bits of the destination address. Bit zero indicates whether the address is multicast or unicast and corresponds to the least significant bit of the first byte received.

43.5.23 Specific Address 4 Top Register

Name: EMAC_SA4T

Address: 0xFFFBC0B4

Access: Read-write

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

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

15	14	13	12	11	10	9	8
ADDR							

7	6	5	4	3	2	1	0
ADDR							

- **ADDR**

The most significant bits of the destination address, that is bits 47 to 32.

43.5.24 Type ID Checking Register

Name: EMAC_TID

Address: 0xFFFBC0B8

Access: 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
TID							

7	6	5	4	3	2	1	0
TID							

- **TID: Type ID checking**

For use in comparisons with received frames TypeID/Length field.

43.5.25 User Input/Output Register

Name: EMAC_USRIO

Address: 0xFFFBC0C0

Access: 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
–	–	–	–	–	–	CLKEN	RMII

- **RMII**

When set, this bit enables the RMII operation mode. When reset, it selects the MII mode.

- **CLKEN**

When set, this bit enables the transceiver input clock.

Setting this bit to 0 reduces power consumption when the treasurer is not used.

43.5.26 Wake-on-LAN Register

Name: EMAC_WOL

Address: 0xFFFBC0C4

Access: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	MTI	SA1	ARP	MAG
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
IP							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IP							

- **IP: ARP request IP address**

Written to define the least significant 16 bits of the target IP address that is matched to generate a Wake-on-LAN event. A value of zero does not generate an event, even if this is matched by the received frame.

- **MAG: Magic packet event enable**

When set, magic packet events causes the `wol` output to be asserted.

- **ARP: ARP request event enable**

When set, ARP request events causes the `wol` output to be asserted.

- **SA1: Specific address register 1 event enable**

When set, specific address 1 events causes the `wol` output to be asserted.

- **MTI: Multicast hash event enable**

When set, multicast hash events causes the `wol` output to be asserted.

43.5.27 EMAC Statistic Registers

These registers reset to zero on a read and stick at all ones when they count to their maximum value. They should be read frequently enough to prevent loss of data. The receive statistics registers are only incremented when the receive enable bit is set in the network control register. To write to these registers, bit 7 must be set in the network control register. The statistics register block contains the following registers.

43.5.27.1 Pause Frames Received Register

Name: EMAC_PFR

Address: 0xFFFBC03C

Access: 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**
FROK							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FROK							

- **FROK: Pause Frames received OK**

A 16-bit register counting the number of good pause frames received. A good frame has a length of 64 to 1518 (1536 if bit 8 set in network configuration register) and has no FCS, alignment or receive symbol errors.

43.5.27.2 Frames Transmitted OK Register

Name: EMAC_FTO

Address: 0xFFFBC040

Access: Read-write

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

- **FTOK: Frames Transmitted OK**

A 24-bit register counting the number of frames successfully transmitted, i.e., no underrun and not too many retries.

43.5.27.3 *Single Collision Frames Register*

Name: EMAC_SCF

Address: 0xFFFBC044

Access: 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
SCF							

7	6	5	4	3	2	1	0
SCF							

- **SCF: Single Collision Frames**

A 16-bit register counting the number of frames experiencing a single collision before being successfully transmitted, i.e., no underrun.

43.5.27.4 *Multicollision Frames Register*

Name: EMAC_MCF

Address: 0xFFFBC048

Access: 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
MCF							

7	6	5	4	3	2	1	0
MCF							

- **MCF: Multicollision Frames**

A 16-bit register counting the number of frames experiencing between two and fifteen collisions prior to being successfully transmitted, i.e., no underrun and not too many retries.

43.5.27.5 Frames Received OK Register

Name: EMAC_FRO

Address: 0xFFFBC04C

Access: Read-write

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

- **FROK: Frames Received OK**

A 24-bit register counting the number of good frames received, i.e., address recognized and successfully copied to memory. A good frame is of length 64 to 1518 bytes (1536 if bit 8 set in network configuration register) and has no FCS, alignment or receive symbol errors.

43.5.27.6 Frames Check Sequence Errors Register

Name: EMAC_FCSE

Address: 0xFFFBC050

Access: 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**
FCSE							

- **FCSE: Frame Check Sequence Errors**

An 8-bit register counting frames that are an integral number of bytes, have bad CRC and are between 64 and 1518 bytes in length (1536 if bit 8 set in network configuration register). This register is also incremented if a symbol error is detected and the frame is of valid length and has an integral number of bytes.

43.5.27.7 *Alignment Errors Register*

Name: EMAC_ALE

Address: 0xFFFBC054

Access: 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**
ALE							

- **ALE: Alignment Errors**

An 8-bit register counting frames that are not an integral number of bytes long and have bad CRC when their length is truncated to an integral number of bytes and are between 64 and 1518 bytes in length (1536 if bit 8 set in network configuration register). This register is also incremented if a symbol error is detected and the frame is of valid length and does not have an integral number of bytes.

43.5.27.8 *Deferred Transmission Frames Register*

Name: EMAC_DTF

Address: 0xFFFBC058

Access: 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**
DTF							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DTF							

- **DTF: Deferred Transmission Frames**

A 16-bit register counting the number of frames experiencing deferral due to carrier sense being active on their first attempt at transmission. Frames involved in any collision are not counted nor are frames that experienced a transmit underrun.

43.5.27.9 Late Collisions Register

Name: EMAC_LCOL

Address: 0xFFFBC05C

Access: 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
LCOL							

- **LCOL: Late Collisions**

An 8-bit register counting the number of frames that experience a collision after the slot time (512 bits) has expired. A late collision is counted twice; i.e., both as a collision and a late collision.

43.5.27.10 Excessive Collisions Register

Name: EMAC_ECOL

Address: 0xFFFBC060

Access: 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
EXCOL							

- **EXCOL: Excessive Collisions**

An 8-bit register counting the number of frames that failed to be transmitted because they experienced 16 collisions.

43.5.27.11 Transmit Underrun Errors Register

Name: EMAC_TUND

Address: 0xFFFBC064

Access: 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**
TUND							

- **TUND: Transmit Underruns**

An 8-bit register counting the number of frames not transmitted due to a transmit DMA underrun. If this register is incremented, then no other statistics register is incremented.

43.5.27.12 Carrier Sense Errors Register

Name: EMAC_CSE

Address: 0xFFFBC068

Access: 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**
CSE							

- **CSE: Carrier Sense Errors**

An 8-bit register counting the number of frames transmitted where carrier sense was not seen during transmission or where carrier sense was deasserted after being asserted in a transmit frame without collision (no underrun). Only incremented in half-duplex mode. The only effect of a carrier sense error is to increment this register. The behavior of the other statistics registers is unaffected by the detection of a carrier sense error.

43.5.27.13 Receive Resource Errors Register

Name: EMAC_RRE

Address: 0xFFFBC06C

Access: 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**
RRE							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RRE							

- **RRE: Receive Resource Errors**

A 16-bit register counting the number of frames that were address matched but could not be copied to memory because no receive buffer was available.

43.5.27.14 Receive Overrun Errors Register

Name: EMAC_ROV

Address: 0xFFFBC070

Access: 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**
ROVR							

- **ROVR: Receive Overrun**

An 8-bit register counting the number of frames that are address recognized but were not copied to memory due to a receive DMA overrun.

43.5.27.15 *Receive Symbol Errors Register*

Name: EMAC_RSE

Address: 0xFFFBC074

Access: 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
RSE							

- **RSE: Receive Symbol Errors**

An 8-bit register counting the number of frames that had `rx_er` asserted during reception. Receive symbol errors are also counted as an FCS or alignment error if the frame is between 64 and 1518 bytes in length (1536 if bit 8 is set in the network configuration register). If the frame is larger, it is recorded as a jabber error.

43.5.27.16 *Excessive Length Errors Register*

Name: EMAC_ELE

Address: 0xFFFBC078

Access: 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
EXL							

- **EXL: Excessive Length Errors**

An 8-bit register counting the number of frames received exceeding 1518 bytes (1536 if bit 8 set in network configuration register) in length but do not have either a CRC error, an alignment error nor a receive symbol error.

43.5.27.17 Receive Jabbers Register

Name: EMAC_RJA

Address: 0xFFFBC07C

Access: 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
RJB							

- **RJB: Receive Jabbers**

An 8-bit register counting the number of frames received exceeding 1518 bytes (1536 if bit 8 set in network configuration register) in length and have either a CRC error, an alignment error or a receive symbol error.

43.5.27.18 Undersize Frames Register

Name: EMAC_USF

Address: 0xFFFBC080

Access: 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
USF							

- **USF: Undersize Frames**

An 8-bit register counting the number of frames received less than 64 bytes in length but do not have either a CRC error, an alignment error or a receive symbol error.

43.5.27.19 SQE Test Errors Register

Name: EMAC_STE

Address: 0xFFFBC084

Access: 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**
SQER							

- **SQER: SQE Test Errors**

An 8-bit register counting the number of frames where `col` was not asserted within 96 bit times (an interframe gap) of `tx_en` being deasserted in half duplex mode.

43.5.27.20 Received Length Field Mismatch Register

Name: EMAC_RLE

Address: 0xFFFBC088

Access: 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**
RLFM							

- **RLFM: Receive Length Field Mismatch**

An 8-bit register counting the number of frames received that have a measured length shorter than that extracted from its length field. Checking is enabled through bit 16 of the network configuration register. Frames containing a type ID in bytes 13 and 14 (i.e., length/type ID 0x0600) are not counted as length field errors, neither are excessive length frames.

44. USB Host Port (UHP)

44.1 Description

The USB Host Port (UHP) interfaces the USB with the host application. It handles Open HCI protocol (Open Host Controller Interface) as well as USB v2.0 Full-speed and Low-speed protocols.

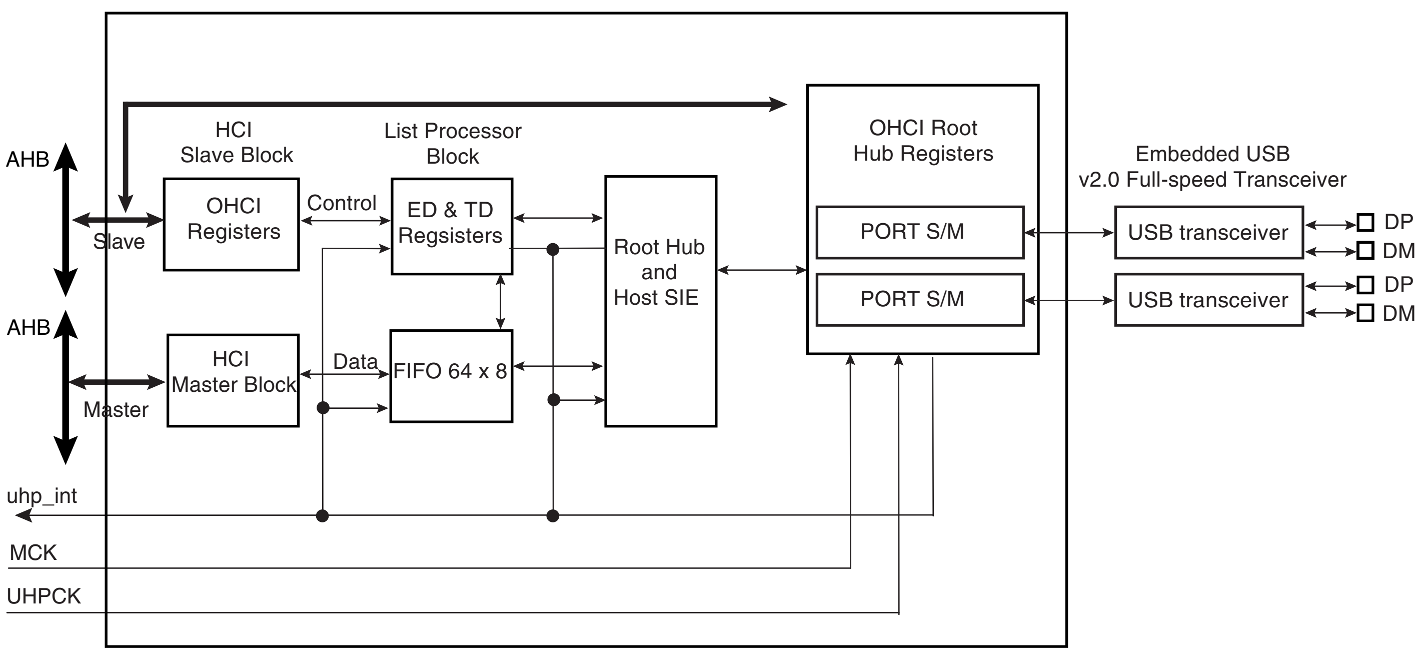
The USB Host Port integrates a root hub and transceivers on downstream ports. It provides several high-speed half-duplex serial communication ports at a baud rate of 12 Mbit/s. Up to 127 USB devices (printer, camera, mouse, keyboard, disk, etc.) and the USB hub can be connected to the USB host in the USB “tiered star” topology.

The USB Host Port controller is fully compliant with the OpenHCI specification. The USB Host Port User Interface (registers description) can be found in the Open HCI Rev 1.0 Specification available on http://h18000.www1.hp.com/productinfo/development/openhci.html. The standard OHCI USB stack driver can be easily ported to ATMEL’s architecture in the same way all existing class drivers run without hardware specialization.

This means that all standard class devices are automatically detected and available to the user application. As an example, integrating an HID (Human Interface Device) class driver provides a plug & play feature for all USB keyboards and mouses.

44.2 Block Diagram

Figure 44-1. Block Diagram



Access to the USB host operational registers is achieved through the AHB bus slave interface. The OpenHCI host controller initializes master DMA transfers through the ASB bus master interface as follows:

- Fetches endpoint descriptors and transfer descriptors
- Access to endpoint data from system memory

- Access to the HC communication area
- Write status and retire transfer Descriptor

Memory access errors (abort, misalignment) lead to an "Unrecoverable Error" indicated by the corresponding flag in the host controller operational registers.

The USB root hub is integrated in the USB host. Several USB downstream ports are available. The number of downstream ports can be determined by the software driver reading the root hub's operational registers. Device connection is automatically detected by the USB host port logic.

USB physical transceivers are integrated in the product and driven by the root hub's ports.

Over current protection on ports can be activated by the USB host controller. Atmel's standard product does not dedicate pads to external over current protection.

44.3 Product Dependencies

44.3.1 I/O Lines

DPs and DMs are not controlled by any PIO controllers. The embedded USB physical transceivers are controlled by the USB host controller.

44.3.2 Power Management

The USB host controller requires a 48 MHz clock. This clock must be generated by a PLL with a correct accuracy of ± 0.25%.

Thus the USB device peripheral receives two clocks from the Power Management Controller (PMC): the master clock MCK used to drive the peripheral user interface (MCK domain) and the UHPCLK 48 MHz clock used to interface with the bus USB signals (Recovered 12 MHz domain).

44.3.3 Interrupt

The USB host interface has an interrupt line connected to the Advanced Interrupt Controller (AIC).

Handling USB host interrupts requires programming the AIC before configuring the UHP.

44.4 Functional Description

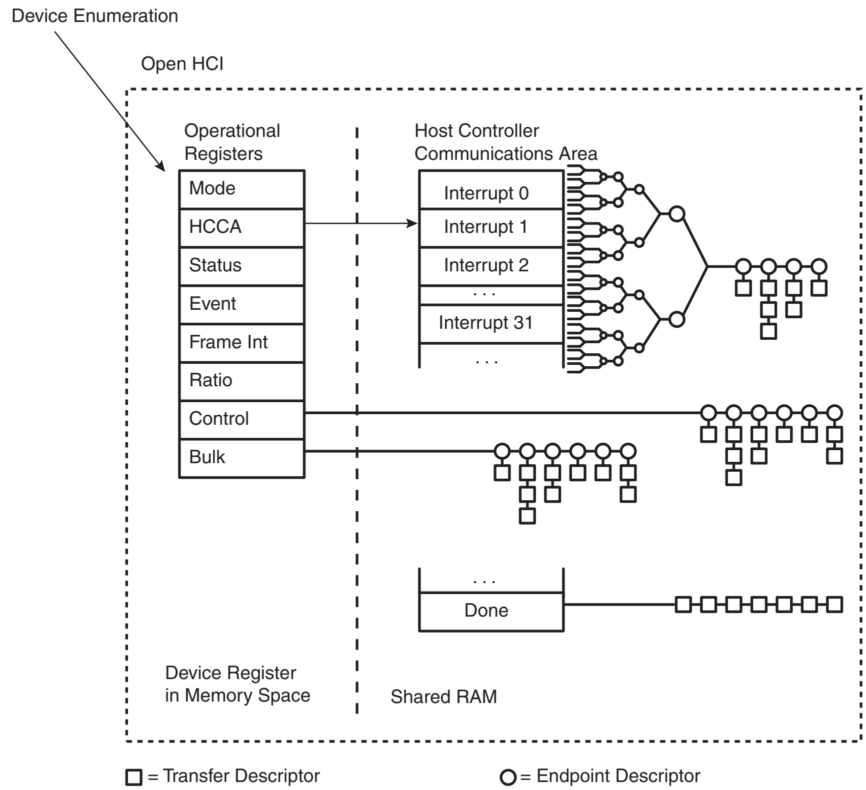
Please refer to the Open Host Controller Interface Specification for USB Release 1.0.a.

44.4.1 Host Controller Interface

There are two communication channels between the Host Controller and the Host Controller Driver. The first channel uses a set of operational registers located on the USB Host Controller. The Host Controller is the target for all communications on this channel. The operational registers contain control, status and list pointer registers. They are mapped in the memory mapped area. Within the operational register set there is a pointer to a location in the processor address space named the Host Controller Communication Area (HCCA). The HCCA is the second communication channel. The host controller is the master for all communication on this channel. The HCCA contains the head pointers to the interrupt Endpoint Descriptor lists, the head pointer to the done queue and status information associated with start-of-frame processing.

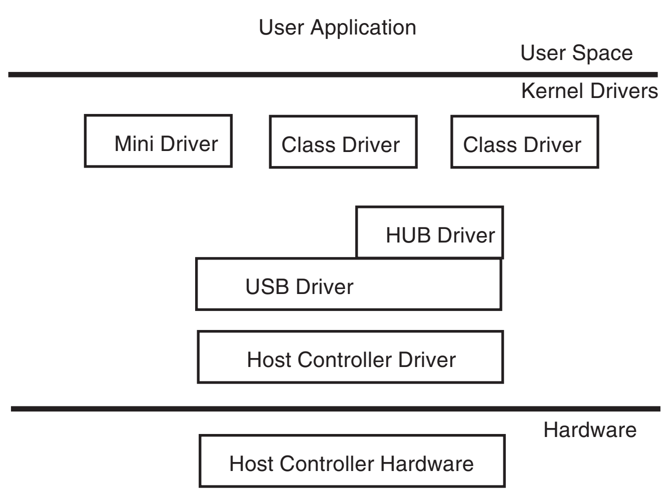
The basic building blocks for communication across the interface are Endpoint Descriptors (ED, 4 double words) and Transfer Descriptors (TD, 4 or 8 double words). The host controller assigns an Endpoint Descriptor to each endpoint in the system. A queue of Transfer Descriptors is linked to the Endpoint Descriptor for the specific endpoint.

Figure 44-2. USB Host Communication Channels



44.4.2 Host Controller Driver

Figure 44-3. USB Host Drivers

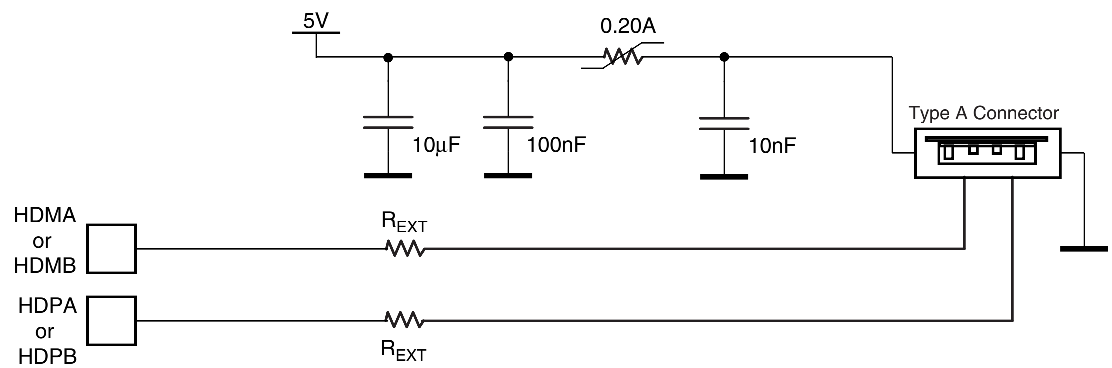


USB Handling is done through several layers as follows:

- Host controller hardware and serial engine: Transmits and receives USB data on the bus.
- Host controller driver: Drives the Host controller hardware and handles the USB protocol.
- USB Bus driver and hub driver: Handles USB commands and enumeration. Offers a hardware independent interface.
- Mini driver: Handles device specific commands.
- Class driver: Handles standard devices. This acts as a generic driver for a class of devices, for example the HID driver.

44.5 Typical Connection

Figure 44-4. Board Schematic to Interface UHP Device Controller



A termination serial resistor must be connected to HDP and HDM. The resistor value is defined in the electrical specification of the product (R_{EXT}).

45. USB High Speed Device Port (UDPHS)

45.1 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 45-1. 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	3	Y	Y	1024	Ctrl/Bulk/Iso[1]/Interrupt
2	EPT_2	3	Y	Y	1024	Ctrl/Bulk/Iso[1]/Interrupt
3	EPT_3	2	Y	N	1024	Ctrl/Bulk/Iso[1]/Interrupt
4	EPT_4	2	Y	N	1024	Ctrl/Bulk/Iso[1]/Interrupt
5	EPT_5	2	Y	N	1024	Ctrl/Bulk/Iso[1]/Interrupt
6	EPT_6	2	Y	N	1024	Ctrl/Bulk/Iso[1]/Interrupt
7	EPT_7	2	N	N	1024	Ctrl/Bulk/Iso[1]/Interrupt

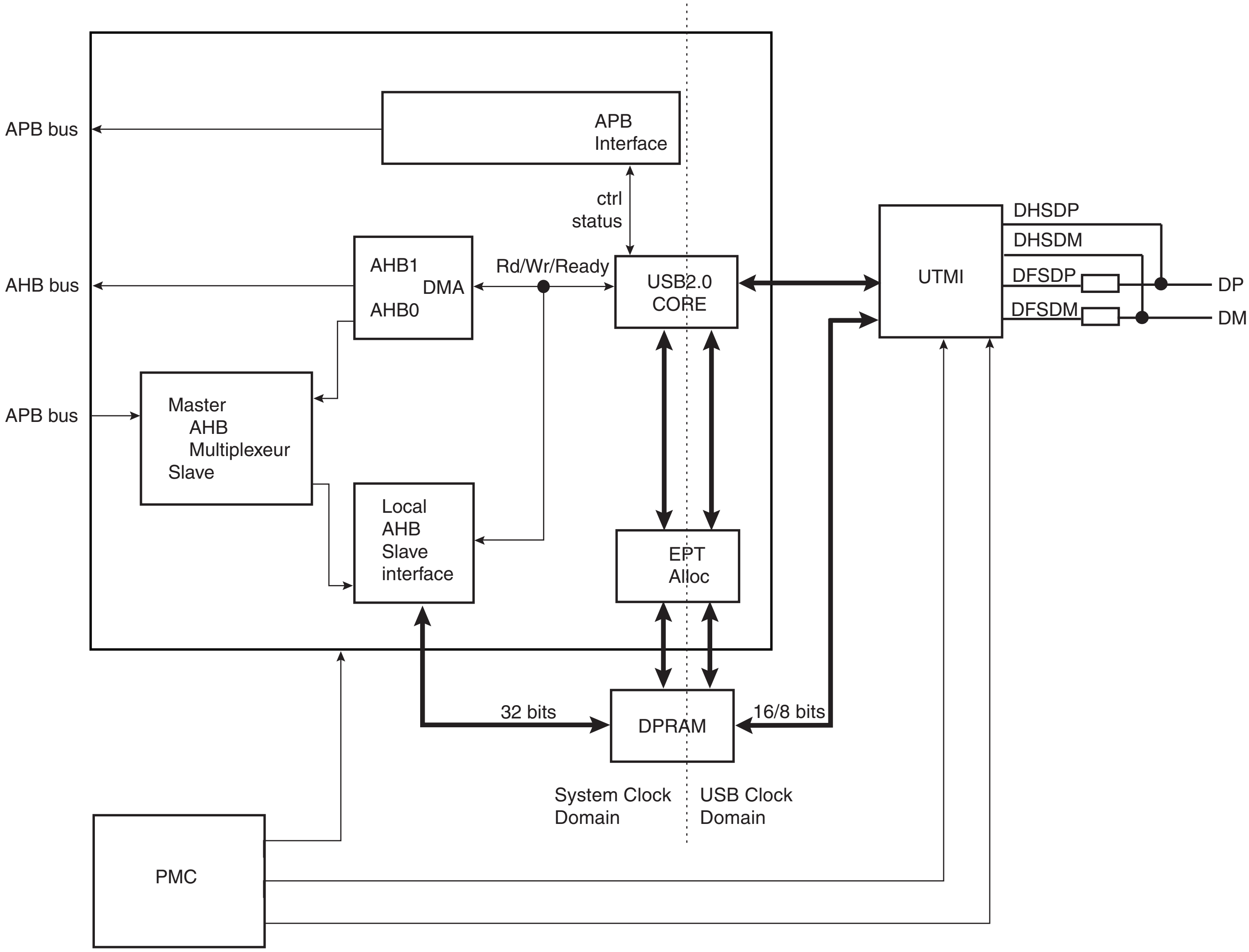
Note: 1. In Isochronous Mode (Iso), it is preferable that High Band Width capability is available.

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.

45.2 Block Diagram

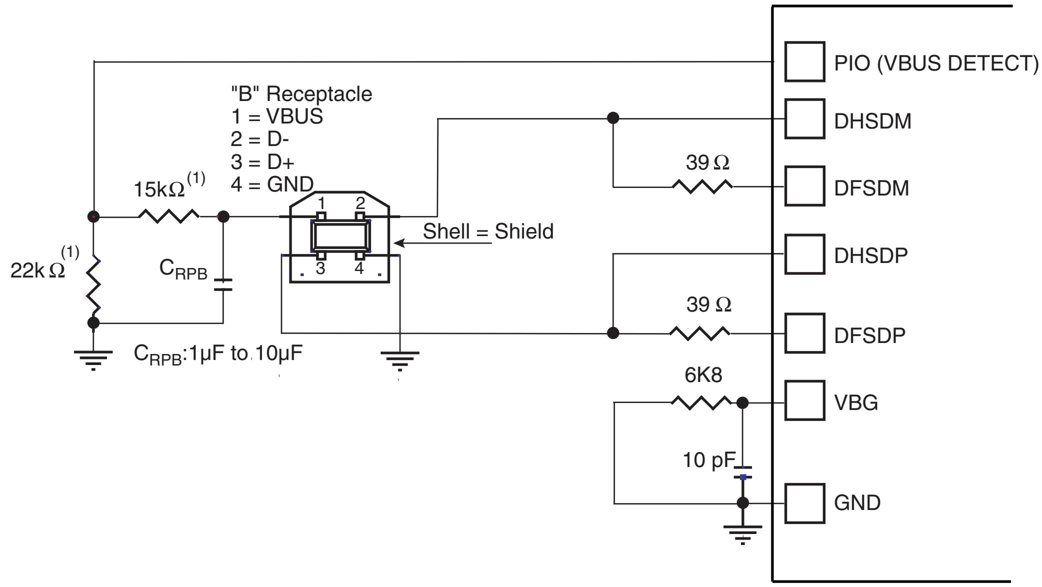
Figure 45-1. Block Diagram



Notes: 1. System clock, bit (1 << AT91C_ID_UDPHS) in PMC_PCER register.
2. Enable UDPHS clock (peripheral clock) bit AT91C_CKGR_UPLLEN in PMC_UCKR register.
3. Enable BIAS bit AT91C_CKGR_BIASEN in PMC_UCKR register.

45.3 Typical Connection

Figure 45-2. Board Schematic



Notes: 1. The values shown on the 22kΩ and 15kΩ resistors are only valid with 3V3 supplied PIOs.

45.4 Functional Description

45.4.1 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.

45.4.2 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 45-2. 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

45.4.3 USB Transfer Event Definitions

A transfer is composed of one or several transactions;

Table 45-3. USB Transfer Events

CONTROL (bidirectional)	Control Transfers [(1)]	• Setup transaction ∅ Data IN transactions ∅Status OUT transaction • Setup transaction ∅ Data OUT transactions ∅Status IN transaction • Setup transaction ∅ Status IN transaction
IN (device toward host)	Bulk IN Transfer	• Data IN transaction ∅ Data IN transaction
	Interrupt IN Transfer	• Data IN transaction ∅ Data IN transaction
	Isochronous IN Transfer [(2)]	• Data IN transaction ∅ Data IN transaction

Table 45-3. USB Transfer Events (Continued)

CONTROL (bidirectional)	Control Transfers [1]	• Setup transaction ∅ Data IN transactions ∅Status OUT transaction • Setup transaction ∅ Data OUT transactions ∅Status IN transaction • Setup transaction ∅ Status IN transaction
OUT (host toward device)	Bulk OUT Transfer	• Data OUT transaction ∅ Data OUT transaction
	Interrupt OUT Transfer	• Data OUT transaction ∅ Data OUT transaction
	Isochronous OUT Transfer [2]	• 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.

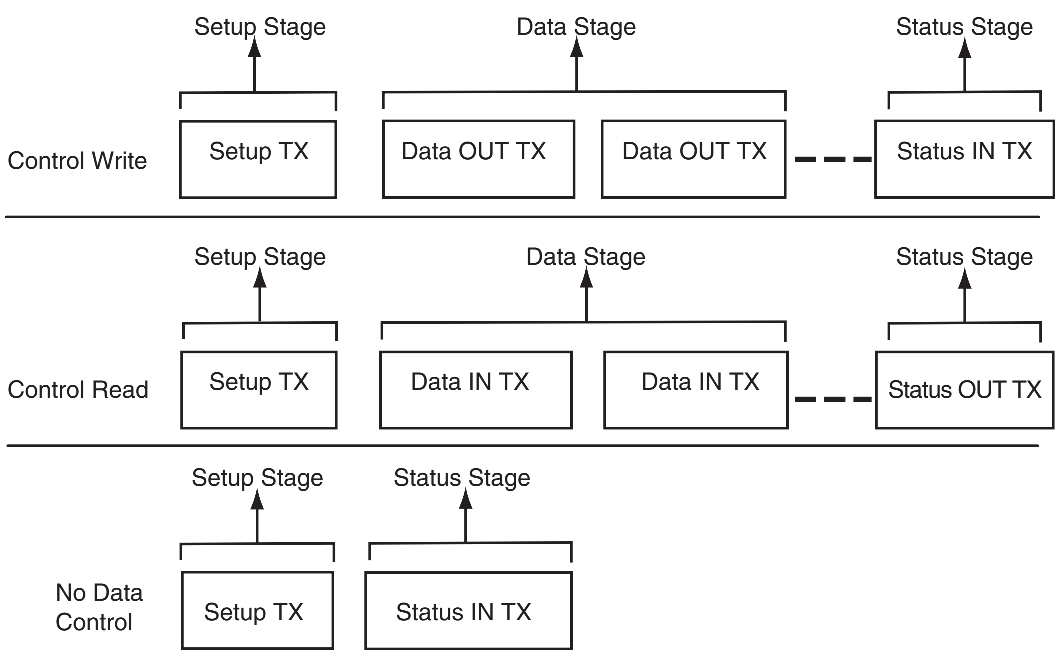
45.4.4 USB V2.0 High Speed BUS Transactions

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 45-3. Control Read and Write Sequences



A status IN or OUT transaction is identical to a data IN or OUT transaction.

45.4.5 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 954.

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 45-1. 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.

SIZE_DPRAM = 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

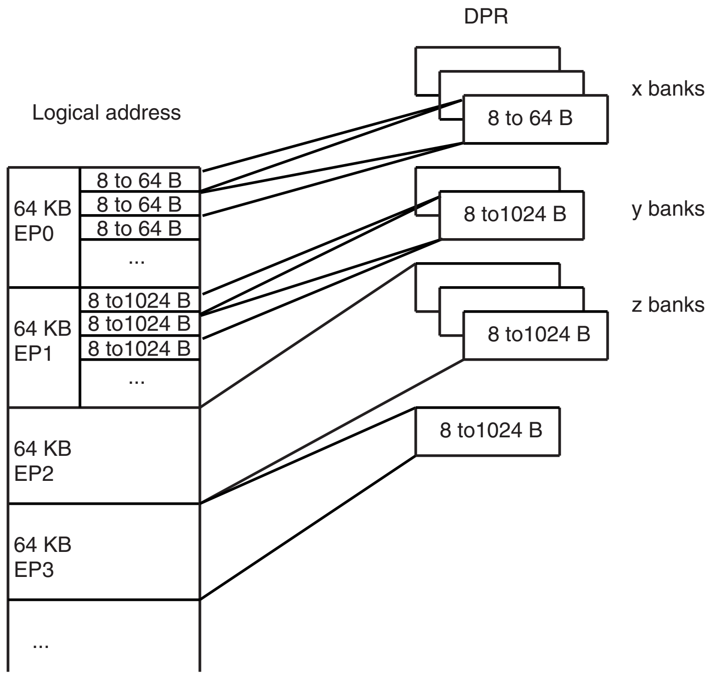
+... (refer to 45.5.11 UDPHS Endpoint Configuration Register)

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 45-4. 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.

45.4.6 Transfer With DMA

USB packets of any length may be transferred when required by the UDPHS Device. These transfers always feature sequential addressing.

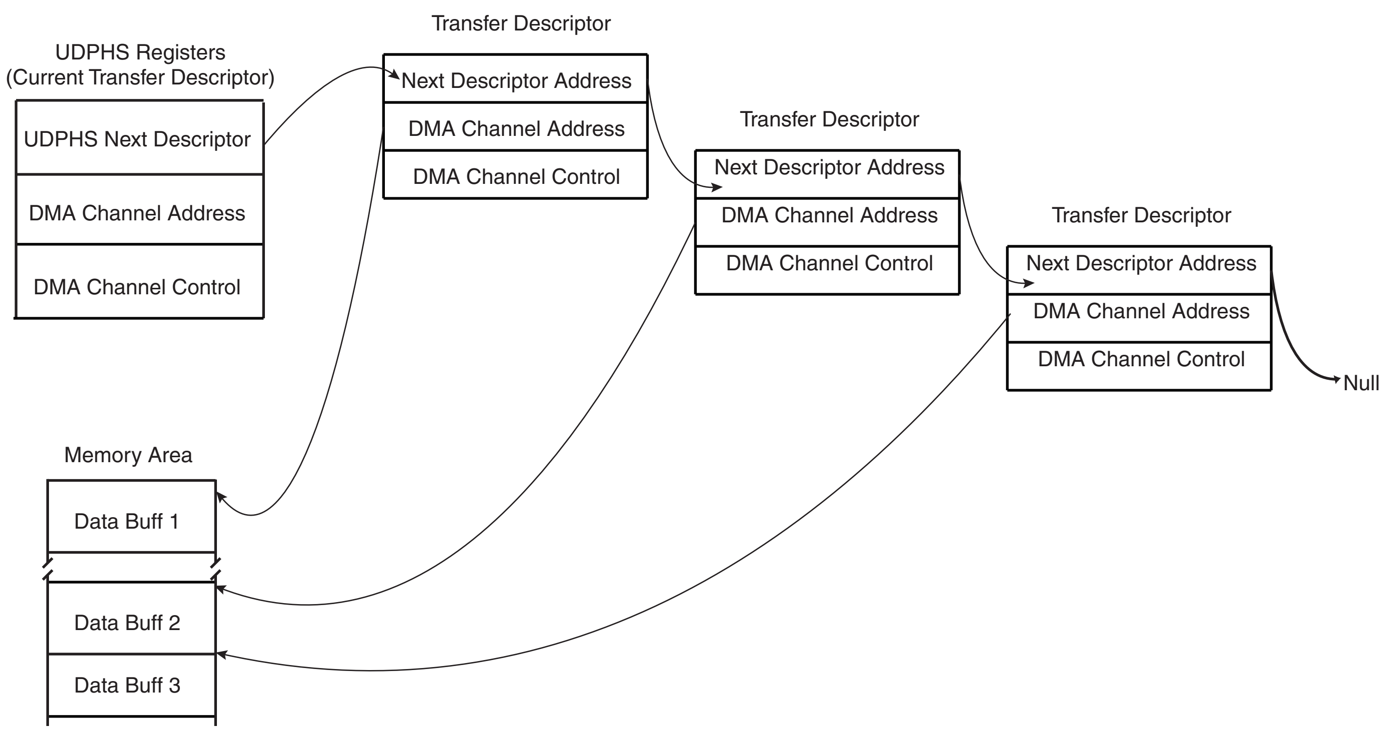
Packet data AHB bursts may be locked on a DMA buffer basis for drastic overall AHB bus bandwidth performance boost with paged memories. These clock-cycle consuming memory row (or bank) changes will then likely not occur, or occur only once instead of dozens times, during a single big USB packet DMA transfer 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 965.

Note: In case of debug, be careful to address the DMA to an SRAM address even if a remap is done.

Figure 45-5. Example of DMA Chained List:



45.4.7 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;
}
```

45.4.8 Handling Transactions with USB V2.0 Device Peripheral

45.4.8.1 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 45.4.8.15 “STALL” on page 925).

45.4.8.2 NYET

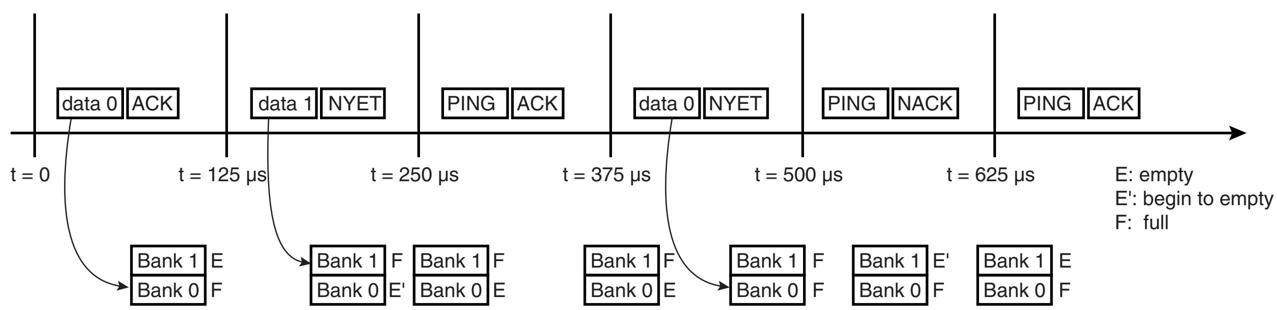
NYET is a High Speed only handshake. It is returned by a High Speed endpoint as part of the PING protocol.

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 45-6. NYET Example with Two Endpoint Banks



45.4.8.3 Data IN

45.4.8.4 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 45.4.8.5 below)
- 64 KB (see 45.4.8.5 below)
- DMA (see 45.4.8.6 below)

45.4.8.5 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.

45.4.8.6 *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 954 and Figure 45-11. 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 965). 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 45-7. Data IN Transfer for Endpoint with One Bank

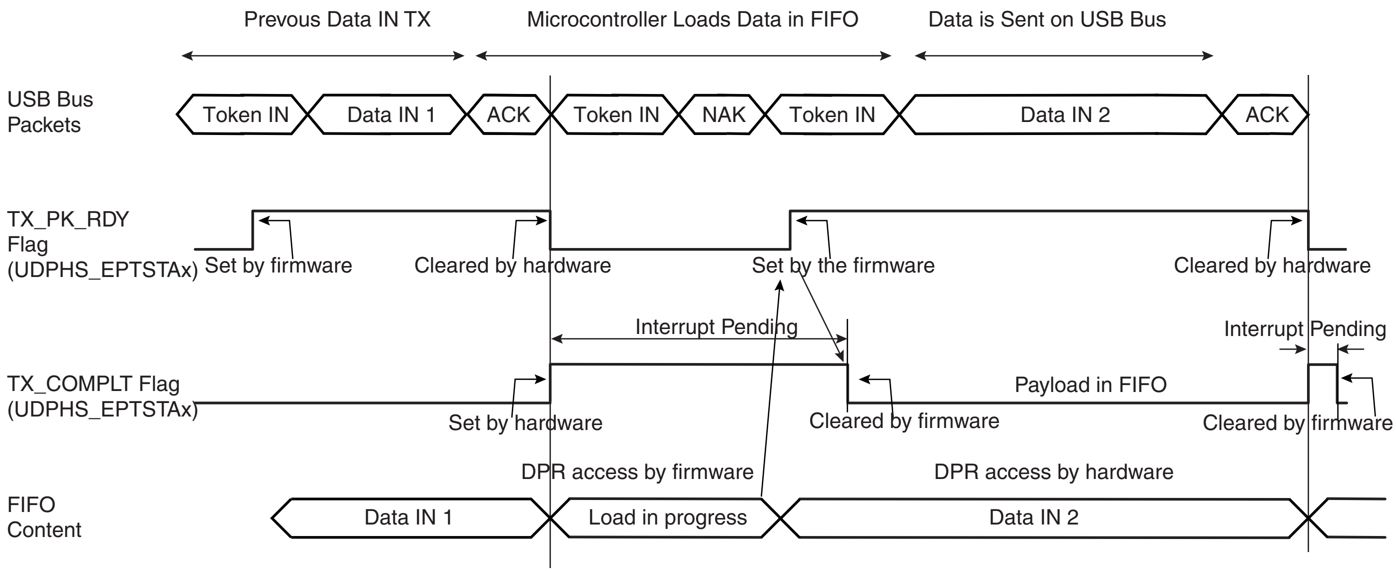


Figure 45-8. Data IN Transfer for Endpoint with Two Banks

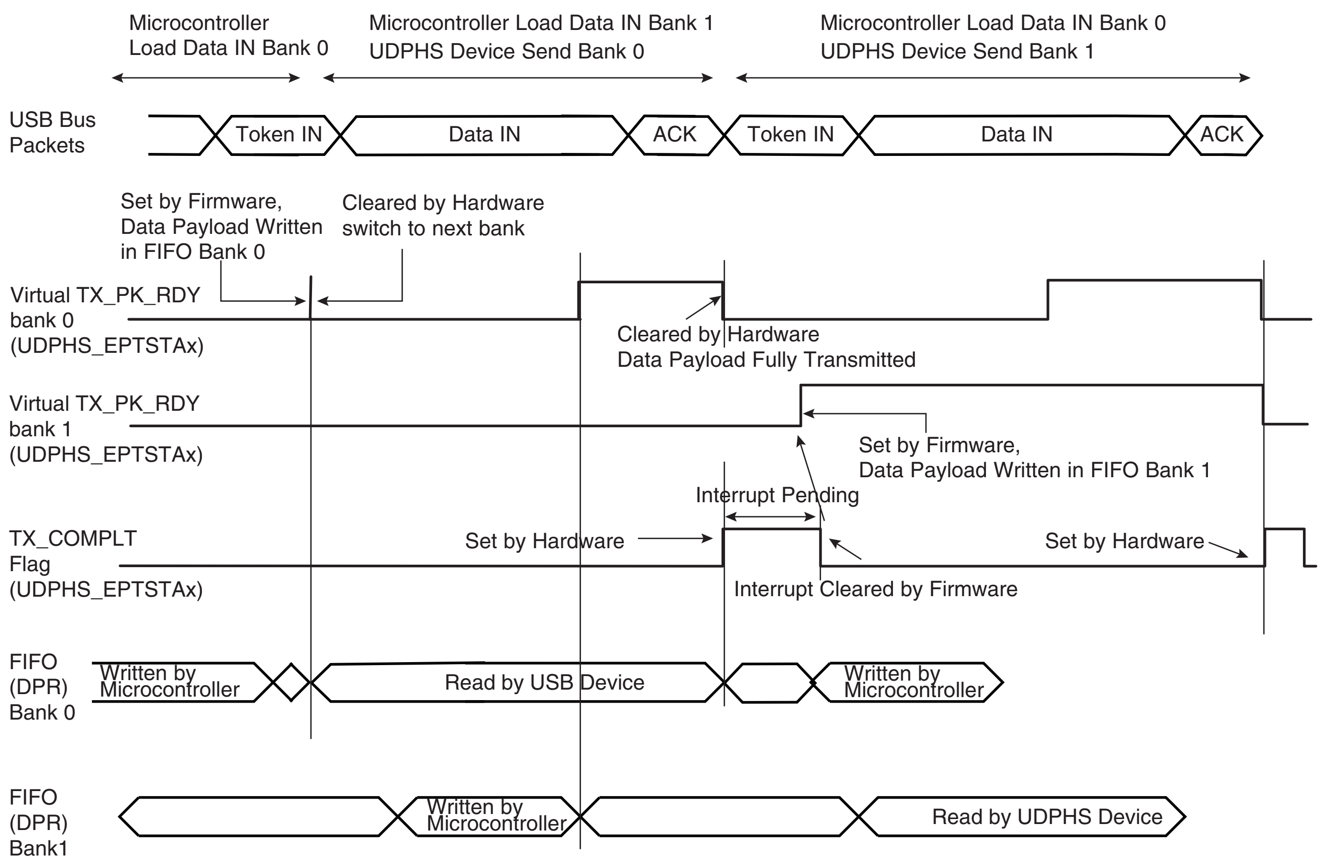
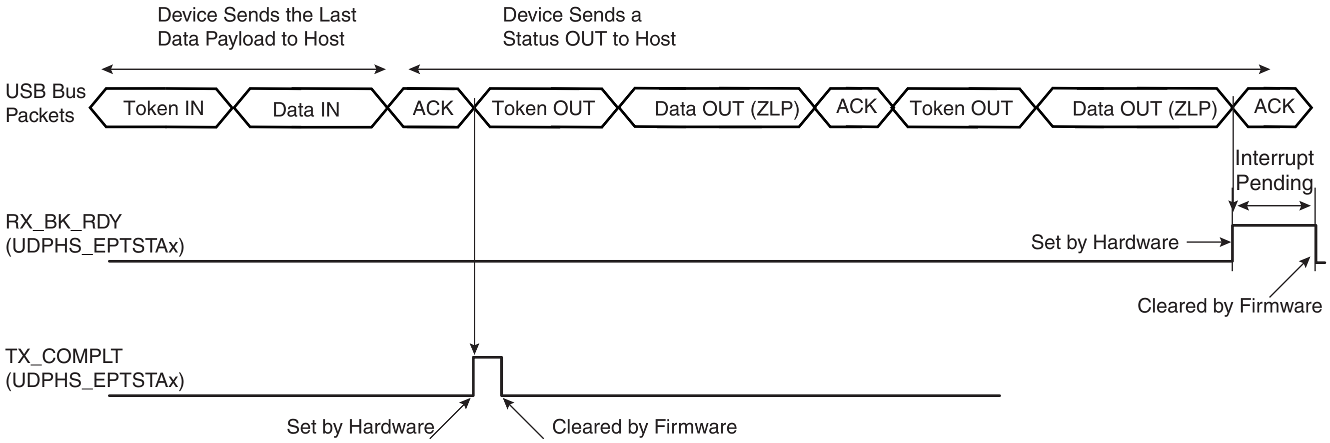
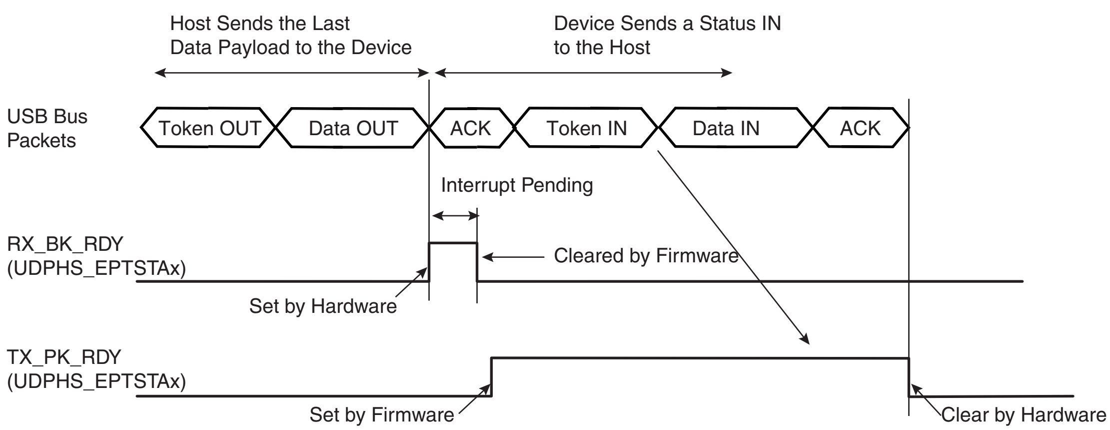


Figure 45-9. Data IN Followed By Status OUT Transfer at the End of a Control Transfer



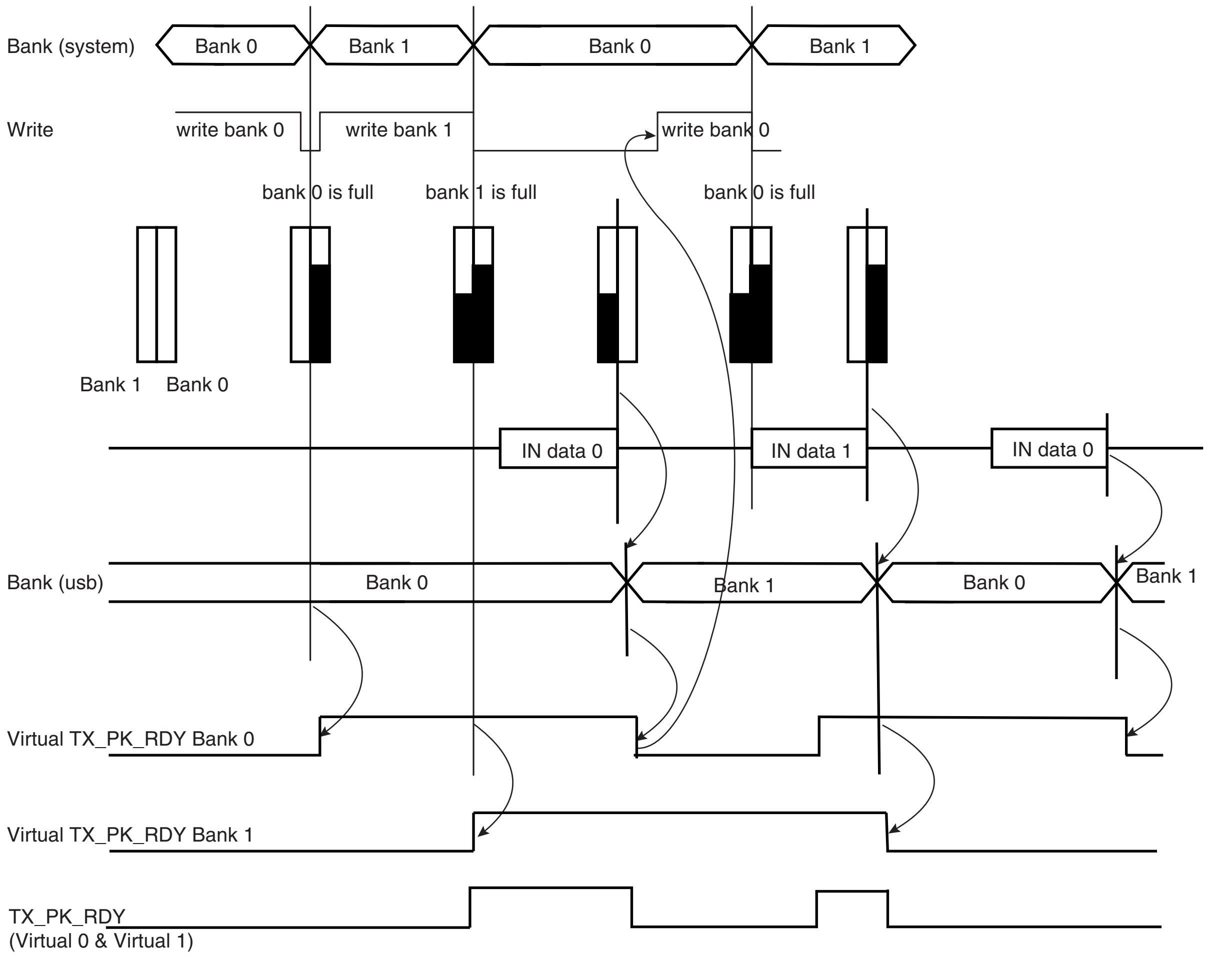
Note: A NAK handshake is always generated at the first status stage token.

Figure 45-10. 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 45-11. 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.

45.4.8.7 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.

45.4.8.8 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 DATAx 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.

45.4.8.9 Data OUT

45.4.8.10 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.

45.4.8.11 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.

45.4.8.12 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 45.4.8.6 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 45-12. Data OUT Transfer for Endpoint with One Bank

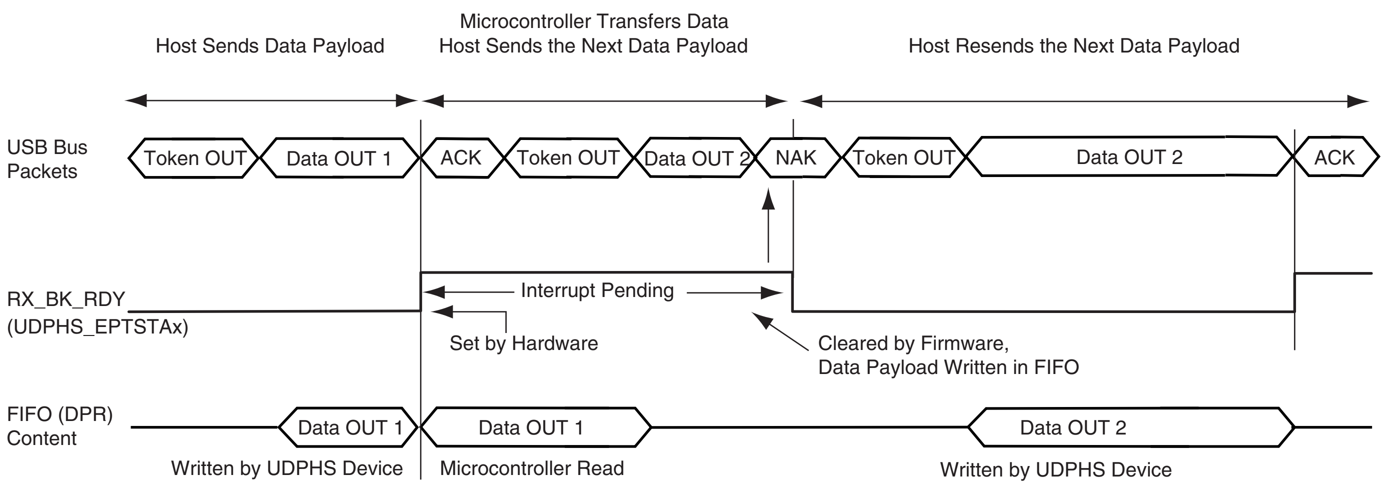
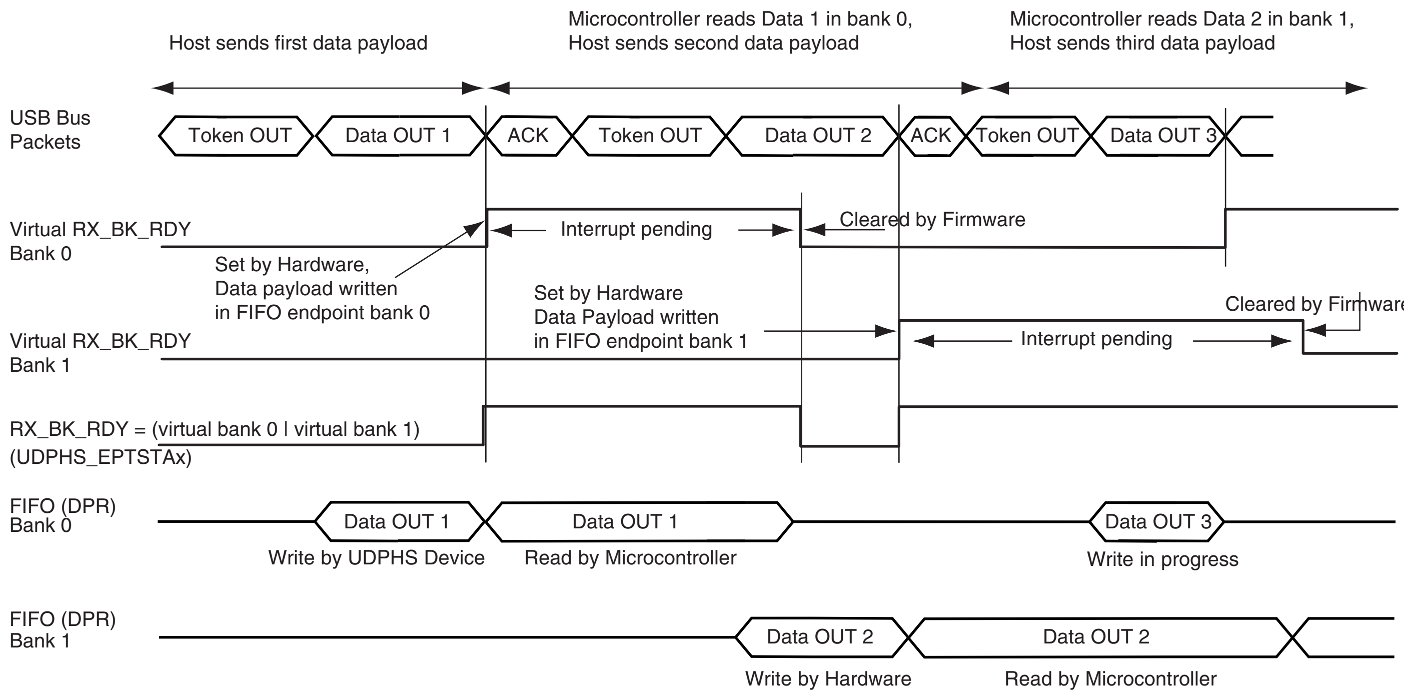
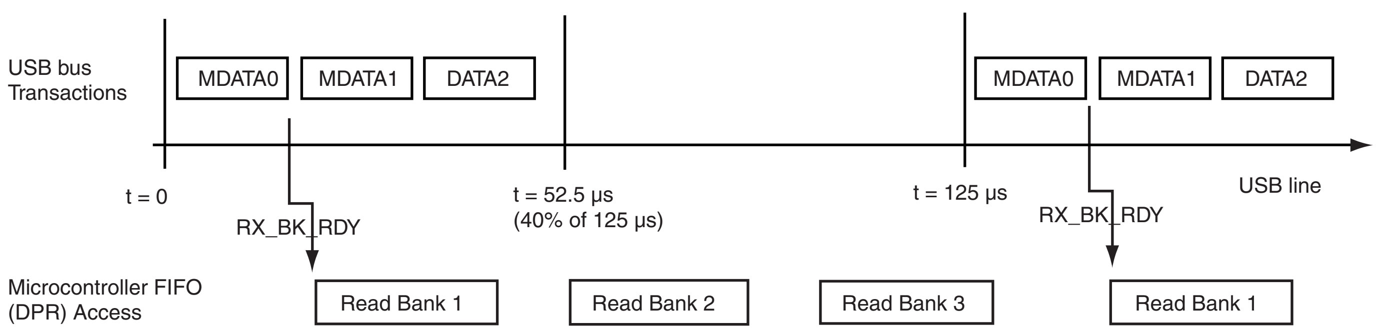


Figure 45-13. Data OUT Transfer for an Endpoint with Two Banks



45.4.8.13 High Bandwidth Isochronous Endpoint OUT

Figure 45-14. 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

45.4.8.14 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.

45.4.8.15 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 45-15. Stall Handshake Data OUT Transfer

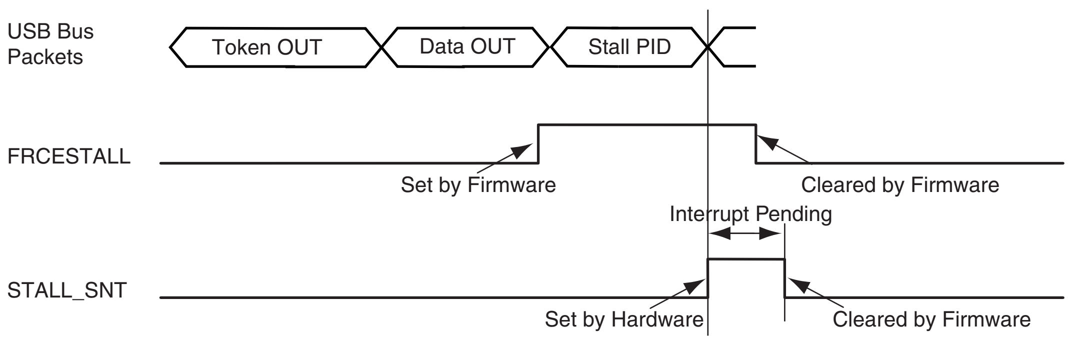
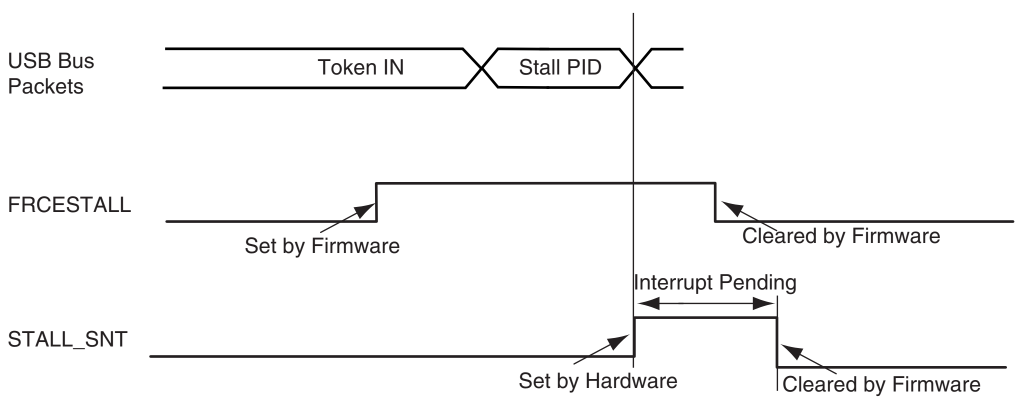


Figure 45-16. Stall Handshake Data IN Transfer



45.4.9 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.

45.4.10 USB V2.0 High Speed Global Interrupt

Interrupts are defined in Section 45.5.3 "UDPHS Interrupt Enable Register" (UDPHS_IEN) and in Section 45.5.4 "UDPHS Interrupt Status Register" (UDPHS_INTSTA).

45.4.11 Endpoint Interrupts

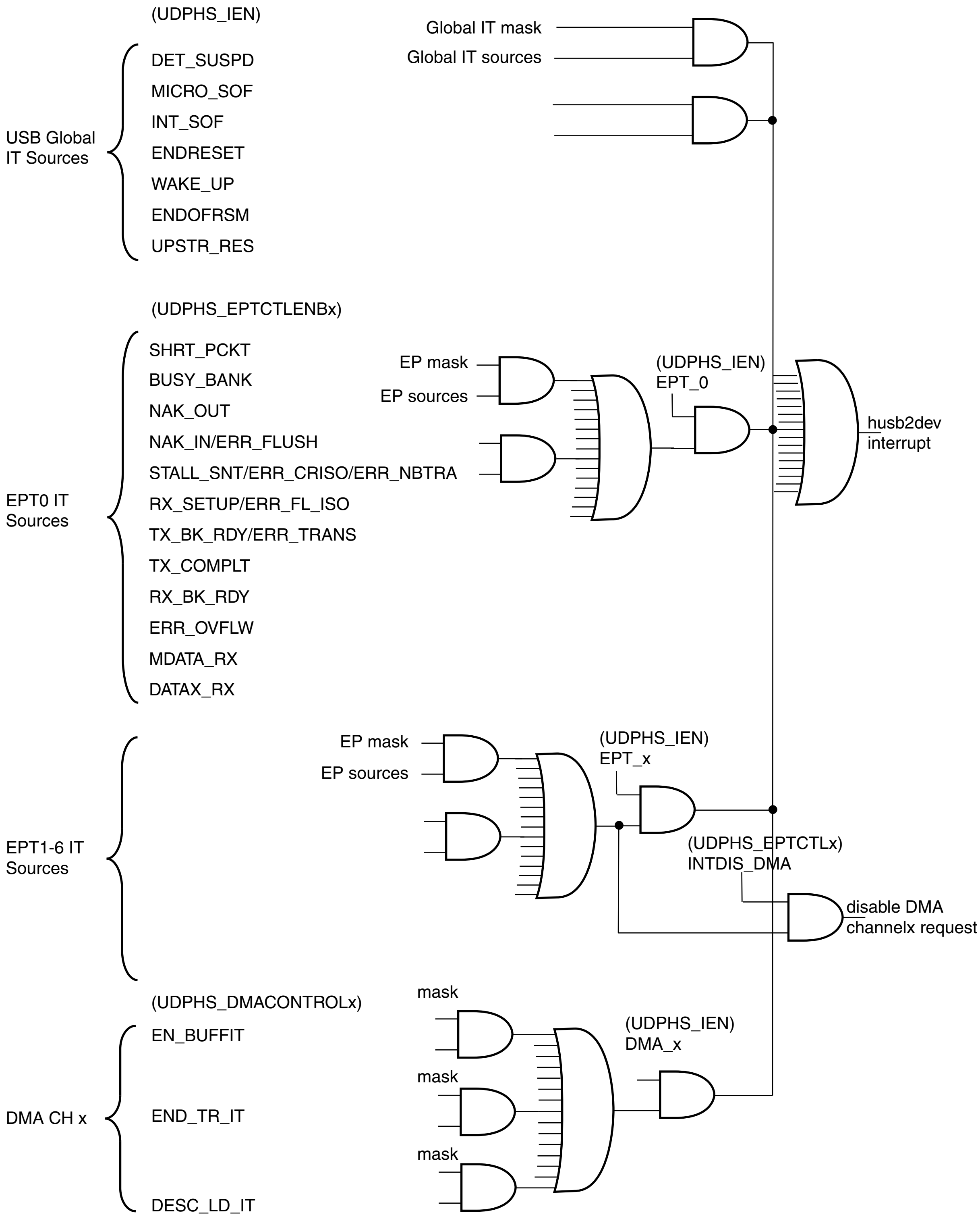
Interrupts are enabled in UDPHS_IEN (see Section 45.5.3 "UDPHS Interrupt Enable Register") and individually masked in UDPHS_EPTCTLENBx (see Section 45.5.12 "UDPHS Endpoint Control Enable Register").

.

Table 45-4. 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
DATAX_RX	DATAx Interrupt

Figure 45-17. UDPHS Interrupt Control Interface

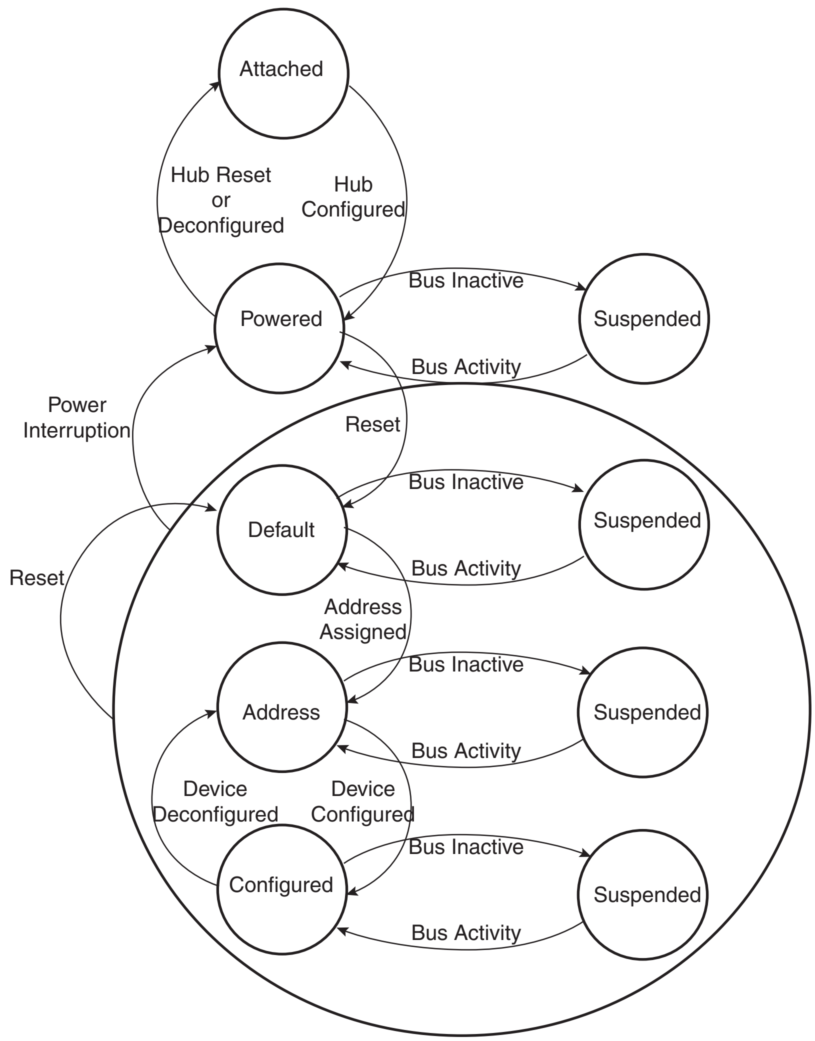


45.4.12 Power Modes

45.4.12.1 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 45-18. 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 µA on the USB bus.

While in Suspend Mode, the host may wake up a device by sending a resume signal (bus activity) or a USB device may send a wake-up request to the host, e.g., waking up a PC by moving a USB mouse.

The wake-up feature is not mandatory for all devices and must be negotiated with the host.

45.4.12.2 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 FSDP lines are tied to GND pull-downs integrated in the hub downstream ports.

45.4.12.3 Entering Attached State

When no device is connected, the USB FSDP and FSDM signals are tied to GND by 15 KΩ pull-downs integrated in the hub downstream ports. When a device is attached to an hub downstream port, the device connects a 1.5 KΩ pull-up on FSDP. The USB bus line goes into IDLE state, FSDP is pulled-up by the device 1.5 KΩ resistor to 3.3V and FSDM is pulled-down by the 15 KΩ 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.

45.4.12.4 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_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.

45.4.12.5 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.

45.4.12.6 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.

45.4.12.7 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 µA 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.

45.4.12.8 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.

45.4.12.9 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.

45.4.13 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 45.5.7 "UDPHS Test Register" on page 943 for definitions of each test mode.)

```
const char test_packet_buffer[] = {
  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,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
};
```

45.5 USB High Speed Device Port (UDPHS) User Interface

Table 45-5. Register Mapping

Offset	Register	Name	Access	Reset
0x00	UDPHS Control Register	UDPHS_CTRL	Read-write	0x0000_0200
0x04	UDPHS Frame Number Register	UDPHS_FNUM	Read	0x0000_0000
0x08 - 0x0C	Reserved	–	–	–
0x10	UDPHS Interrupt Enable Register	UDPHS_IEN	Read-write	0x0000_0010
0x14	UDPHS Interrupt Status Register	UDPHS_INTSTA	Read	0x0000_0000
0x18	UDPHS Clear Interrupt Register	UDPHS_CLRINT	Write	–
0x1C	UDPHS Endpoints Reset Register	UDPHS_EPTRST	Write	–
0x20 - 0xCC	Reserved	–	–	–
0xE0	UDPHS Test Register	UDPHS_TST	Read-write	0x0000_0000
0xE4 - 0xE8	Reserved	–	–	–
0xF0	UDPHS Name1 Register	UDPHS_IPNAME1	Read	0x4855_5342
0xF4	UDPHS Name2 Register	UDPHS_IPNAME2	Read	0x3244_4556
0xF8	UDPHS Features Register	UDPHS_IPFEATURES	Read	
0x100 + endpoint * 0x20 + 0x00	UDPHS Endpoint Configuration Register	UDPHS_EPTCFG	Read-write	0x0000_0000
0x100 + endpoint * 0x20 + 0x04	UDPHS Endpoint Control Enable Register	UDPHS_EPTCTLENB	Write	–
0x100 + endpoint * 0x20 + 0x08	UDPHS Endpoint Control Disable Register	UDPHS_EPTCTLDIS	Write	–
0x100 + endpoint * 0x20 + 0x0C	UDPHS Endpoint Control Register	UDPHS_EPTCTL	Read	0x0000_0000(1)
0x100 + endpoint * 0x20 + 0x10	Reserved (for endpoint)	–	–	–
0x100 + endpoint * 0x20 + 0x14	UDPHS Endpoint Set Status Register	UDPHS_EPTSETSTA	Write	–
0x100 + endpoint * 0x20 + 0x18	UDPHS Endpoint Clear Status Register	UDPHS_EPTCLRSTA	Write	–
0x100 + endpoint * 0x20 + 0x1C	UDPHS Endpoint Status Register	UDPHS_EPTSTA	Read	0x0000_0040
0x120 - 0x1FC	UDPHS Endpoint1 to 7 (2) Registers			
0x200 - 0x300	Reserved			
0x300 - 0x30C	Reserved	–	–	–
0x310 + channel * 0x10 + 0x00	UDPHS DMA Next Descriptor Address Register	UDPHS_DMANXTDSC	Read-write	0x0000_0000
0x310 + channel * 0x10 + 0x04	UDPHS DMA Channel Address Register	UDPHS_DMAADDRESS	Read-write	0x0000_0000
0x310 + channel * 0x10 + 0x08	UDPHS DMA Channel Control Register	UDPHS_DMACONTROL	Read-write	0x0000_0000
0x310 + channel * 0x10 + 0x0C	UDPHS DMA Channel Status Register	UDPHS_DMASTATUS	Read-write	0x0000_0000
0x320 - 0x36C	DMA Channel2 to 6 (3) 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 0x1FC.

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 0x36C.

45.5.1 UDPHS Control Register

Name: UDPHS_CTRL

Address: 0xFFF78000

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 UTMI transceiver. The UTMI may disable the pullup.

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).

(See PULLD_DIS description below.)

- **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.

(See DETACH description above.)

DETACH	PULLD_DIS	DP	DM	Condition
0	0	Pull up	Pull down	not recommended
0	1	Pull up	High impedance state	VBUS present
1	0	Pull down	Pull down	No VBUS
1	1	High impedance state	High impedance state	VBUS present & software disconnect

45.5.2 UDPHS Frame Number Register

Name: UDPHS_FNUM

Address: 0xFFF78004

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.

45.5.3 UDPHS Interrupt Enable Register

Name: UDPHS_IEN

Address: 0xFFF78010

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_7	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.

- **ENDOFRSM: 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_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.

45.5.4 UDPHS Interrupt Status Register

Name: UDPHS_INTSTA

Address: 0xFFF78014

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_7	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_x bit in UDPHS_IEN.

- **DMA_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_x bit in UDPHS_IEN.

45.5.5 UDPHS Clear Interrupt Register

Name: UDPHS_CLRINT

Address: 0xFFF78018

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.

45.5.6 UDPHS Endpoints Reset Register

Name: UDPHS_EPTRST

Address: 0xFFF7801C

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_7	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.

45.5.7 UDPHS Test Register

Name: UDPHS_TST

Address: 0xFFF780E0

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.

45.5.8 UDPHS Name1 Register

Name: UDPHS_IPNAME1

Address: 0xFFF780F0

Access Type: Read-only

31	30	29	28	27	26	25	24
IP_NAME1							

23	22	21	20	19	18	17	16
IP_NAME1							

15	14	13	12	11	10	9	8
IP_NAME1							

7	6	5	4	3	2	1	0
IP_NAME1							

- **IP_NAME1**

ASCII string “HUSB”

45.5.9 UDPHS Name2 Register

Name: UDPHS_IPNAME2

Address: 0xFFF780F4

Access Type: Read-only

31	30	29	28	27	26	25	24
IP_NAME2							

23	22	21	20	19	18	17	16
IP_NAME2							

15	14	13	12	11	10	9	8
IP_NAME2							

7	6	5	4	3	2	1	0
IP_NAME2							

- **IP_NAME2**

ASCII string “2DEV”

45.5.10 UDPHS Features Register

Name: UDPHS_IPFEATURES

Address: 0xFFF780F8

Access Type: Read-only

31	30	29	28	27	26	25	24
ISO_EPT_15	ISO_EPT_14	ISO_EPT_13	ISO_EPT_12	ISO_EPT_11	ISO_EPT_10	ISO_EPT_9	ISO_EPT_8
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ISO_EPT_7	ISO_EPT_6	ISO_EPT_5	ISO_EPT_4	ISO_EPT_3	ISO_EPT_2	ISO_EPT_1	DATAB16_8
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
BW_DPRAM	FIFO_MAX_SIZE			DMA_FIFO_WORD_DEPTH			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DMA_B_SIZ	DMA_CHANNEL_NBR			EPT_NBR_MAX			

- **EPT_NBR_MAX: Max Number of Endpoints**

Give the max number of endpoints.

0 = if 16 endpoints are hardware implemented.

1 = if 1 endpoint is hardware implemented.

2 = if 2 endpoints are hardware implemented.

...

15 = if 15 endpoints are hardware implemented.

- **DMA_CHANNEL_NBR: Number of DMA Channels**

Give the number of DMA channels.

1 = if 1 DMA channel is hardware implemented.

2 = if 2 DMA channels are hardware implemented.

...

7 = if 7 DMA channels are hardware implemented.

- **DMA_B_SIZ: DMA Buffer Size**

0 = if the DMA Buffer size is 16 bits.

1 = if the DMA Buffer size is 24 bits.

- **DMA_FIFO_WORD_DEPTH: DMA FIFO Depth in Words**

0 = if FIFO is 16 words deep.

1 = if FIFO is 1 word deep.

2 = if FIFO is 2 words deep.

...

15 = if FIFO is 15 words deep.

- **FIFO_MAX_SIZE: DPRAM Size**

0 = if DPRAM is 128 bytes deep.

1 = if DPRAM is 256 bytes deep.

2 = if DPRAM is 512 bytes deep.

3 = if DPRAM is 1024 bytes deep.

4 = if DPRAM is 2048 bytes deep.

5 = if DPRAM is 4096 bytes deep.

6 = if DPRAM is 8192 bytes deep.

7 = if DPRAM is 16384 bytes deep.

- **BW_DPRAM: DPRAM Byte Write Capability**

0 = if DPRAM Write Data Shadow logic is implemented.

1 = if DPRAM is byte write capable.

- **DATAB16_8: UTMI DataBus16_8**

0 = if the UTMI uses an 8-bit parallel data interface (60 MHz, unidirectional).

1 = if the UTMI uses a 16-bit parallel data interface (30 MHz, bidirectional).

- **ISO_EPT_x: Endpointx High Bandwidth Isochronous Capability**

0 = if the endpoint does not have isochronous High Bandwidth Capability.

1 = if the endpoint has isochronous High Bandwidth Capability.

45.5.11 UDPHS Endpoint Configuration Register

Name: UDPHS_EPTCFGx [x=0..7]

Addresses: 0xFFF78100 [0], 0xFFF78120 [1], 0xFFF78140 [2], 0xFFF78160 [3], 0xFFF78180 [4], 0xFFF781A0 [5], 0xFFF781C0 [6], 0xFFF781E0 [7]

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 45.4.5 ”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[(1)]

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 45.4.5 ”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 45.4.5 "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

45.5.12 UDPHS Endpoint Control Enable Register

Name: UDPHS_EPTCTLENBx [x=0..7]

Addresses: 0xFFF78104 [0], 0xFFF78124 [1], 0xFFF78144 [2], 0xFFF78164 [3], 0xFFF78184 [4], 0xFFF781A4 [5], 0xFFF781C4 [6], 0xFFF781E4 [7]

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	DATAX_RX	–	NYET_DIS	INTDIS_DMA	–	AUTO_VALID	EPT_ENABL

For additional Information, see "UDPHS Endpoint Control Register" on page 954.

- **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.

- **DATAX_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.

45.5.13 UDPHS Endpoint Control Disable Register

Name: UDPHS_EPTCTLDISx [x=0..7]

Addresses: 0xFFF78108 [0], 0xFFF78128 [1], 0xFFF78148 [2], 0xFFF78168 [3], 0xFFF78188 [4], 0xFFF781A8 [5], 0xFFF781C8 [6], 0xFFF781E8 [7]

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	DATAX_RX	–	NYET_DIS	INTDIS_DMA	–	AUTO_VALID	EPT_DISABL

For additional Information, see "UDPHS Endpoint Control Register" on page 954.

- **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.

- **DATAX_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.

45.5.14 UDPHS Endpoint Control Register

Name: UDPHS_EPTCTLx [x=0..7]

Addresses: 0xFFF7810C [0], 0xFFF7812C [1], 0xFFF7814C [2], 0xFFF7816C [3], 0xFFF7818C [4], 0xFFF781AC [5], 0xFFF781CC [6], 0xFFF781EC [7]

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	DATAX_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_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.

- **DATAX_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.

45.5.15 UDPHS Endpoint Set Status Register

Name: UDPHS_EPTSETSTAx [x=0..7]

Addresses: 0xFFF78114 [0], 0xFFF78134 [1], 0xFFF78154 [2], 0xFFF78174 [3], 0xFFF78194 [4], 0xFFF781B4 [5], 0xFFF781D4 [6], 0xFFF781F4 [7]

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.

45.5.16 UDPHS Endpoint Clear Status Register

Name: UDPHS_EPTCLRSTAx [x=0..7]

Addresses: 0xFFF78118 [0], 0xFFF78138 [1], 0xFFF78158 [2], 0xFFF78178 [3], 0xFFF78198 [4], 0xFFF781B8 [5], 0xFFF781D8 [6], 0xFFF781F8 [7]

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.

45.5.17 UDPHS Endpoint Status Register

Name: UDPHS_EPTSTAx [x=0..7]

Addresses: 0xFFF7811C [0], 0xFFF7813C [1], 0xFFF7815C [2], 0xFFF7817C [3], 0xFFF7819C [4], 0xFFF781BC [5], 0xFFF781DC [6], 0xFFF781FC [7]

Access Type: Read-only

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td>SHRT_PCKT</td><td colspan="7">BYTE_COUNT</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="4">BYTE_COUNT</td><td colspan="2">BUSY_BANK_STA</td><td colspan="2">CURRENT_BANK/
CONTROL_DIR</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td>NAK_OUT</td><td>NAK_IN/
ERR_FLUSH</td><td>STALL_SNT/
ERR_CRISO/
ERR_NBTRA</td><td>RX_SETUP/
ERR_FL_ISO</td><td>TX_PK_RDY/
ERR_TRANS</td><td>TX_COMPLT</td><td>RX_BK_RDY/
KILL_BANK</td><td>ERR_OVFLW</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">TOGGLESQ_STA</td><td>FRCESTALL</td><td>–</td><td>–</td><td>–</td><td>–</td><td>–</td></tr>
</table>

- **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 963) 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).

45.5.18 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.

45.5.19 UDPHS DMA Next Descriptor Address Register

Name: UDPHS_DMANXTDSCx [x = 1..6]

Addresses: 0xFFF78320 [1], 0xFFF78330 [2], 0xFFF78340 [3], 0xFFF78350 [4], 0xFFF78360 [5], 0xFFF78370 [6]

Access Type: Read-write

31	30	29	28	27	26	25	24
NXT_DSC_ADD							

23	22	21	20	19	18	17	16
NXT_DSC_ADD							

15	14	13	12	11	10	9	8
NXT_DSC_ADD							

7	6	5	4	3	2	1	0
NXT_DSC_ADD							

- **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.

45.5.20 UDPHS DMA Channel Address Register

Name: UDPHS_DMAADDRESSx [x = 1..6]

Addresses: 0xFFF78324 [1], 0xFFF78334 [2], 0xFFF78344 [3], 0xFFF78354 [4], 0xFFF78364 [5], 0xFFF78374 [6]

Access Type: Read-write

31	30	29	28	27	26	25	24
BUFF_ADD							

23	22	21	20	19	18	17	16
BUFF_ADD							

15	14	13	12	11	10	9	8
BUFF_ADD							

7	6	5	4	3	2	1	0
BUFF_ADD							

- **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.

45.5.21 UDPHS DMA Channel Control Register

Name: UDPHS_DMACONTROLx [x = 1..6]

Addresses: 0xFFF78328 [1], 0xFFF78338 [2], 0xFFF78348 [3], 0xFFF78358 [4], 0xFFF78368 [5], 0xFFF78378 [6]

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.

DMA Channel Control Command Summary

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".

45.5.22 UDPHS DMA Channel Status Register

Name: UDPHS_DMASTATUSx [x = 1..6]

Addresses: 0xFFF7832C [1], 0xFFF7833C [2], 0xFFF7834C [3], 0xFFF7835C [4], 0xFFF7836C [5], 0xFFF7837C [6]

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.

46. LCD Controller (LCDC)

46.1 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.

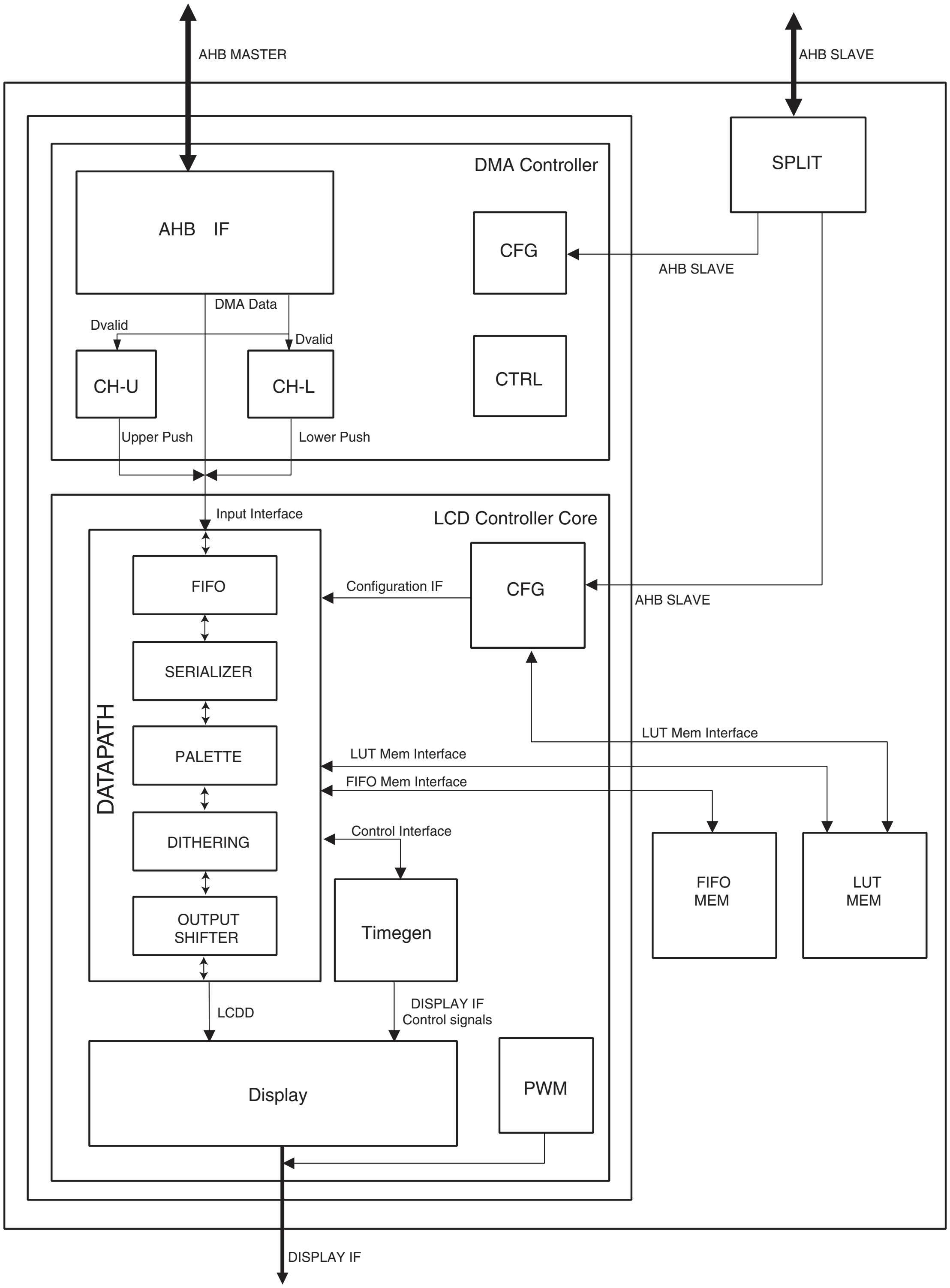
The LCD Controller has a display input buffer (FIFO) to allow a flexible connection of the external AHB master interface, and a lookup table to allow palletized display configurations.

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.

46.2 Block Diagram

Figure 46-1. LCD Macrocell Block Diagram



46.3 I/O Lines Description

Table 46-1. I/O Lines Description

Name	Description	Type
LCDCC	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

46.4 Product Dependencies

46.4.1 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.

Table 46-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
LCDC	LCDCC	PB9	B
LCDC	LCDDEN	PC3	A
LCDC	LCDDOTCK	PC2	A
LCDC	LCDD0	PC4	A
LCDC	LCDD1	PC5	A
LCDC	LCDD2	PC6	A
LCDC	LCDD3	PC4	B
LCDC	LCDD3	PC7	A
LCDC	LCDD4	PC5	B
LCDC	LCDD4	PC8	A
LCDC	LCDD5	PC6	B
LCDC	LCDD5	PC9	A
LCDC	LCDD6	PC7	B
LCDC	LCDD6	PC10	A
LCDC	LCDD7	PC8	B
LCDC	LCDD7	PC11	A
LCDC	LCDD8	PC12	A
LCDC	LCDD9	PC13	A
LCDC	LCDD10	PC9	B
LCDC	LCDD10	PC14	A

Table 46-2. I/O Lines (Continued)

LCDC	LCDD11	PC10	B
LCDC	LCDD11	PC15	A
LCDC	LCDD12	PC11	B
LCDC	LCDD12	PC16	A
LCDC	LCDD13	PC12	B
LCDC	LCDD13	PC17	A
LCDC	LCDD14	PC13	B
LCDC	LCDD14	PC18	A
LCDC	LCDD15	PC14	B
LCDC	LCDD15	PC19	A
LCDC	LCDD16	PC20	A
LCDC	LCDD17	PC21	A
LCDC	LCDD18	PC22	A
LCDC	LCDD19	PC15	B
LCDC	LCDD19	PC23	A
LCDC	LCDD20	PC16	B
LCDC	LCDD20	PC24	A
LCDC	LCDD21	PC17	B
LCDC	LCDD21	PC25	A
LCDC	LCDD22	PC18	B
LCDC	LCDD22	PC26	A
LCDC	LCDD23	PC19	B
LCDC	LCDD23	PC27	A
LCDC	LCDHSYNC	PC1	A
LCDC	LCDVSYNC	PC0	A

46.4.2 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).

46.4.3 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.

Table 46-3. Peripheral IDs

Instance	ID
LCDC	26

46.5 Functional Description

The LCD Controller consists of two main blocks (Figure 46-1 on page 974), 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.

46.5.1 DMA Controller

46.5.1.1 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 1002.

46.5.1.2 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 1007.

46.5.1.3 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:

$$\text{Frame\_size} = \left\lceil \frac{\text{X\_size*Y\_size}}{32} \right\rceil$$

where:

X_size = ((LINESIZE+1)*Bpp+PIXELOFF)/32

Y_size = (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 999), 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.

46.5.1.4 Channel-L

This block has the same functionality as Channel-U, but for the Lower Panel in dual scan mode only.

46.5.1.5 Control

This block receives the request signals from the LCDC core and generates the requests for the channels.

46.5.2 LCD Controller Core

46.5.2.1 Configuration Block

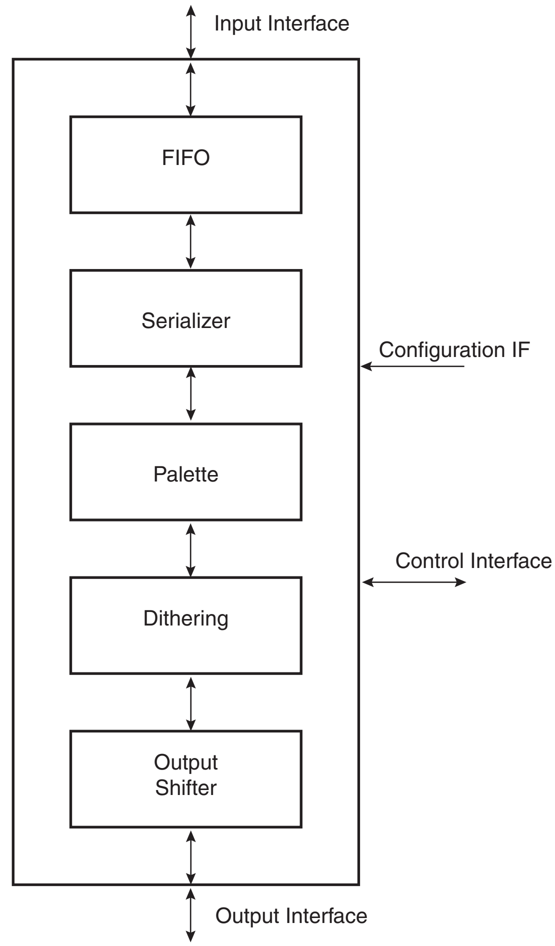
The configuration block is a set of programmable registers that are used to configure the LCDC core operation. These registers are written via the AHB slave interface. Only word access is allowed.

The description of the configuration registers can be found in "LCD Controller (LCDC) User Interface" on page 1002.

46.5.2.2 Datapath

The datapath block contains five submodules: FIFO, Serializer, Palette, Dithering and Shifter. The structure of the datapath is shown in Figure 46-2.

Figure 46-2. 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 46-4.

Table 46-4. 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

46.5.2.3 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:

FIFOTH = 2048 - (2 x 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

46.5.2.4 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 46-5 and Table 7-4.

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 999.).

Table 46-6. 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 46-7. 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															

Table 46-7. Big Endian Memory Organization (Continued)

Mem Addr	0x3	0x2	0x1	0x0
Pixel 24bpp packed	2	3		
Pixel 24bpp packed	4			5
Pixel 24bpp unpacked	not used	0		

Table 46-8. 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																							

46.5.2.5 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 46-9. 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 46-9. Palette Configurations

Configuration		
DISTYPE	PIXELSIZE	Palette
TFT	1, 2, 4, 8	Palletized
TFT	16, 24	Non-palletized
STN Mono	1, 2	Palletized

Table 46-9. Palette Configurations (Continued)

Configuration		
DISTYPE	**PIXELSIZE**	**Palette**
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 46-16 on page 1002.

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 46-10.

Table 46-10. 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]).

46.5.2.6 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 46-11 shows the correspondences between the gray levels and the duty cycle.

Table 46-11. 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
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 $(3\text{-}m)^{th}$ 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 46-12.

Table 46-12. 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

Table 46-12. Dithering Algorithm for Monochrome Mode (Continued)

Frame Number	Pattern	Pixel a	Pixel b	Pixel c	Pixel d
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 46-13 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 46-13. 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.

46.5.2.7 *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 LCDCON2 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 1002.

For a more detailed description of the LCD Interface, see "LCD Interface" on page 991.

46.5.2.8 *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 LCDDOTCK signal depends on the configuration. This information can be found in Table 46-14.

$$f_{LCDDOTCK} = \frac{f_{LCDC\_clock}}{2 \times CLKVAL}$$

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 46-14. Minimum LCDDOTCK Period in LCDC Core Clock Cycles

Configuration			
DISTYPE	**SCAN**	**IFWIDTH**	**LCDDOTCK Period**
TFT			1
STN Mono	Single	4	4
STN Mono	Single	8	8
STN Mono	Dual	8	8
STN Mono	Dual	16	16

Table 46-14. Minimum LCDDOTCK Period in LCDC Core Clock Cycles (Continued)

Configuration			
DISTYPE	**SCAN**	**IFWIDTH**	**LCDDOTCK Period**
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+2) 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 46-4 on page 980. This limitation is given by the following formula:

46.5.2.9 *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 46-4 on page 980

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 46-3, Figure 46-4 and Figure 46-5 show the timing of LCDDOTCK, LCDDEN, LCDHSYNC and LCDVSYNC signals:

Figure 46-3. STN Panel Timing, CLKMOD 0

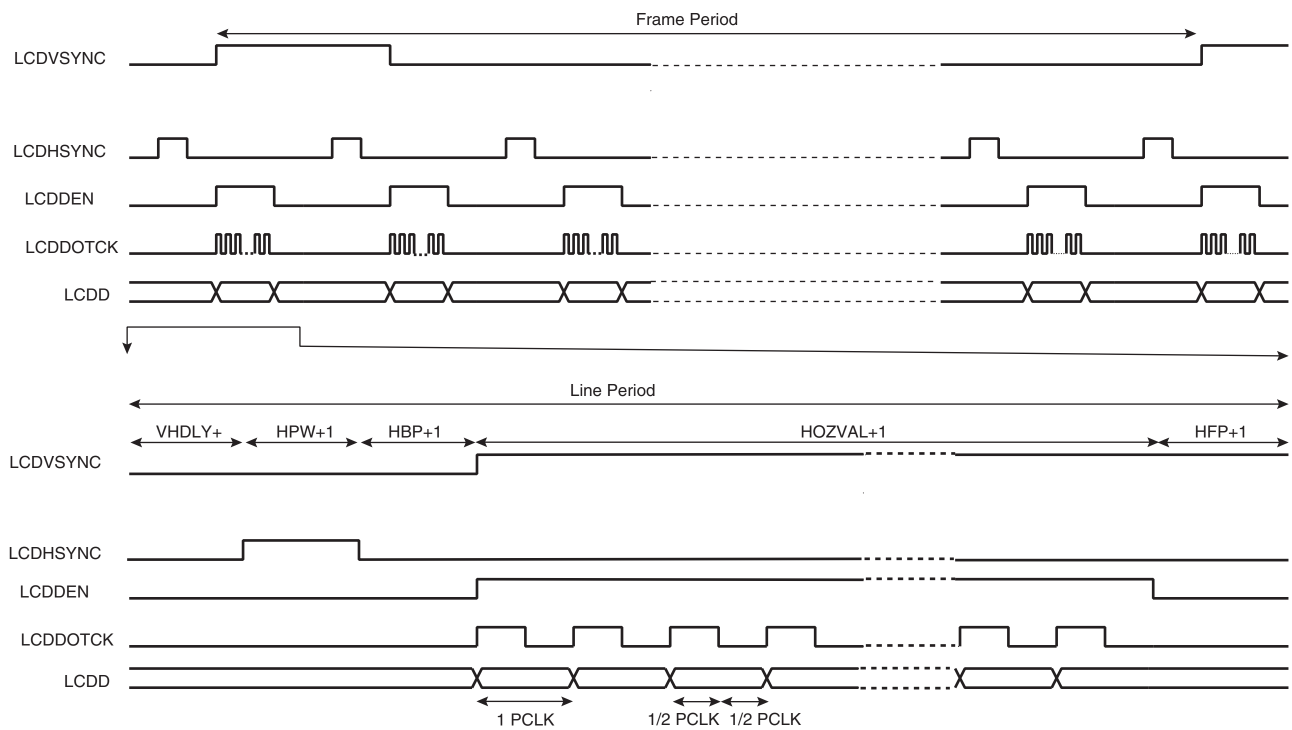


Figure 46-4. TFT Panel Timing, CLKMOD = 0, VPW = 2, VBP = 2, VFP = 1

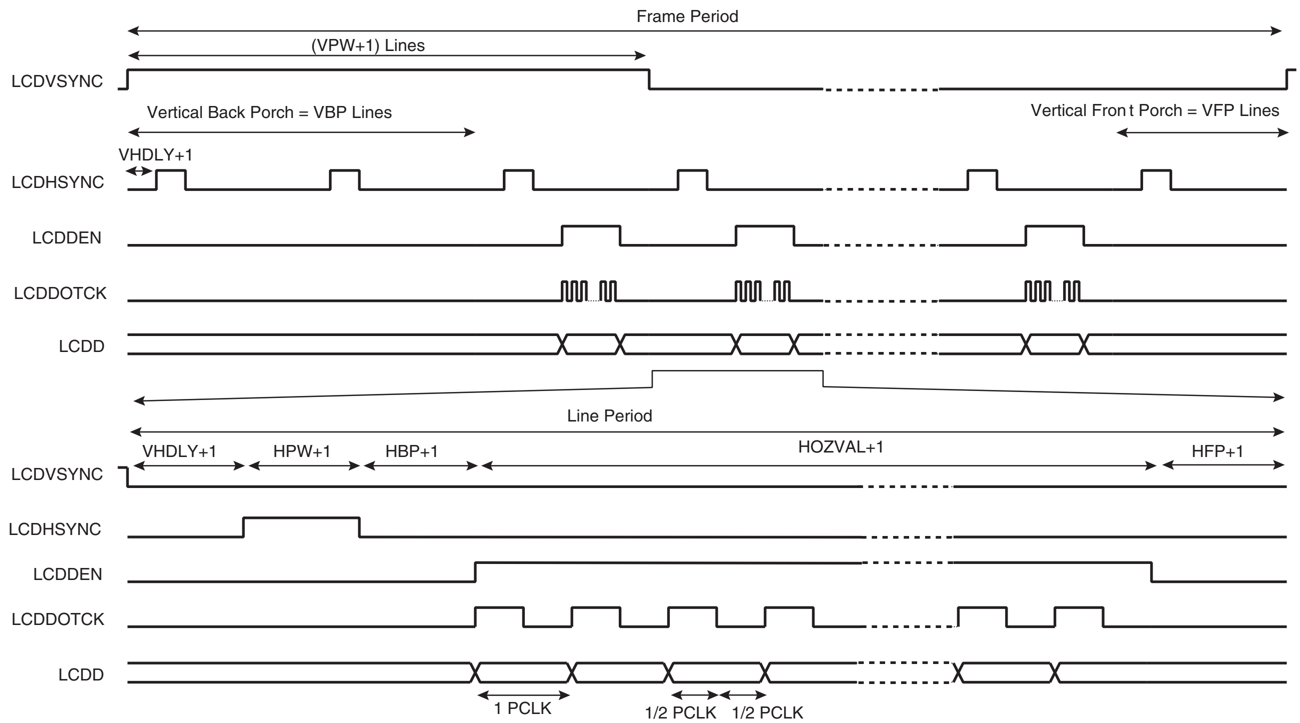
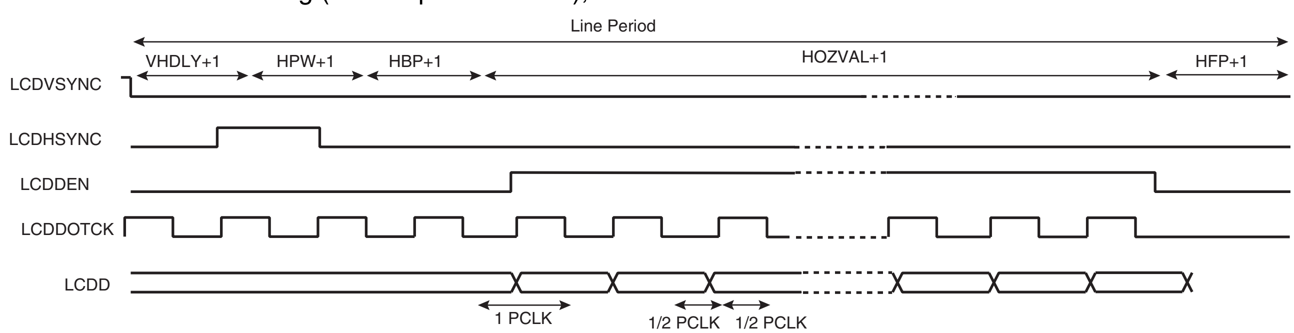


Figure 46-5. TFT Panel Timing (Line Expanded View), CLKMOD=1



Usually the LCD_FRM rate is about 70 Hz to 75 Hz. It is given by the following equation:

$$\frac{1}{f_{LCDVSYNC}} = \left(\frac{VHDLY + HPW + HBP + HOZVAL + HFP + 5}{f_{LCDDOTCK}}\right)(VBP + LINEVAL + VFP + 1)$$

where:

- HOZVAL determines de number of LCDDOTCK cycles per line
- LINEVAL determines the number of LCDHSYNC cycles per frame, according to the expressions shown below:

In STN Mode:

$$HOZVAL = \frac{Horizontal\_display\_size}{Number\_data\_lines} - 1$$

$$LINEVAL = Vertical\_display\_size - 1$$

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:

$$HOZVAL = Horizontal\_display\_size - 1$$

$$LINEVAL = Vertical\_display\_size - 1$$

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:

$$f_{lcd\_pclk} = (HOZVAL + 5) \times (f_{lcd\_vsync} \times (LINEVAL + 1))$$

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 987.

Finally, the frame rate is adjusted to 70 Hz - 75 Hz with the HFP value:

$$HFP = f_{LCDDOTCK} \times \left[\frac{1}{f_{LCDVSYNC} \times (LINEVAL + VBP + VFP + 1)}\right] - (VHDLY + VPW + VBP + HOZVAL + 5)$$

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.

46.5.2.10 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.

46.5.2.11 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 (CONTRAST_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) × VDD can be obtained (for the positive polarity case, or between (1/256) × 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.

46.5.3 LCD Interface

The LCD Controller interfaces with the LCD Module through the LCD Interface (Table 46-15 on page 996). 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 46-6 to Figure 46-10. Figure 46-6 on page 992 shows the 24-bit single scan TFT display timing; Figure 46-7 on page 992 shows the 4-bit single scan STN display timing for monochrome and color modes; Figure 46-8 on page 993 shows the 8-bit single scan STN display timing for monochrome and color modes; Figure 46-9 on page 994 shows the 8-bit Dual Scan STN display timing for monochrome and color modes; Figure 46-10 on page 995 shows the 16-bit Dual Scan STN display timing for monochrome and color modes.

Figure 46-6. TFT Timing (First Line Expanded View)

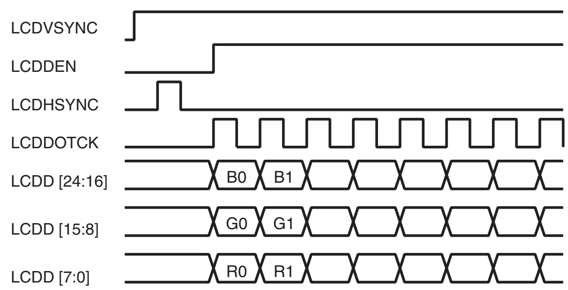


Figure 46-7. Single Scan Monochrome and Color 4-bit Panel Timing (First Line Expanded View)

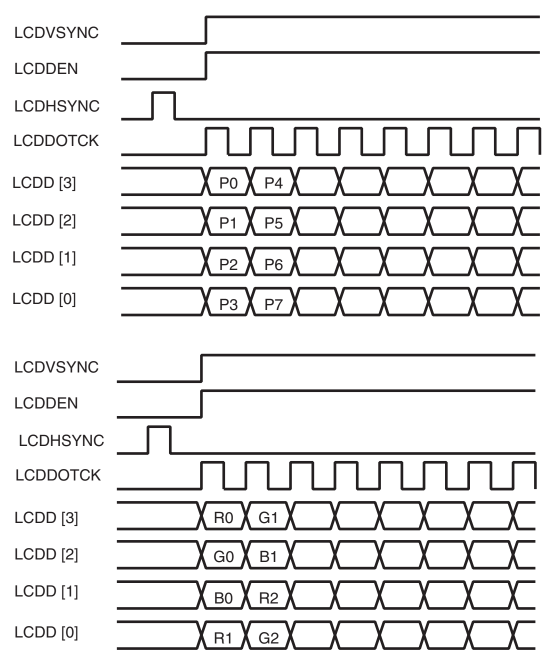


Figure 46-8. Single Scan Monochrome and Color 8-bit Panel Timing (First Line Expanded View)

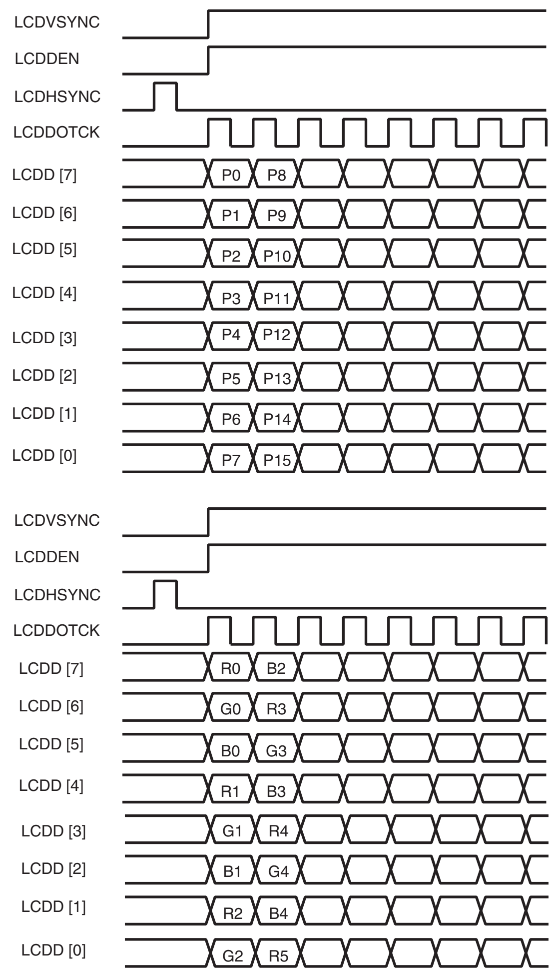


Figure 46-9. Dual Scan Monochrome and Color 8-bit Panel Timing (First Line Expanded View)

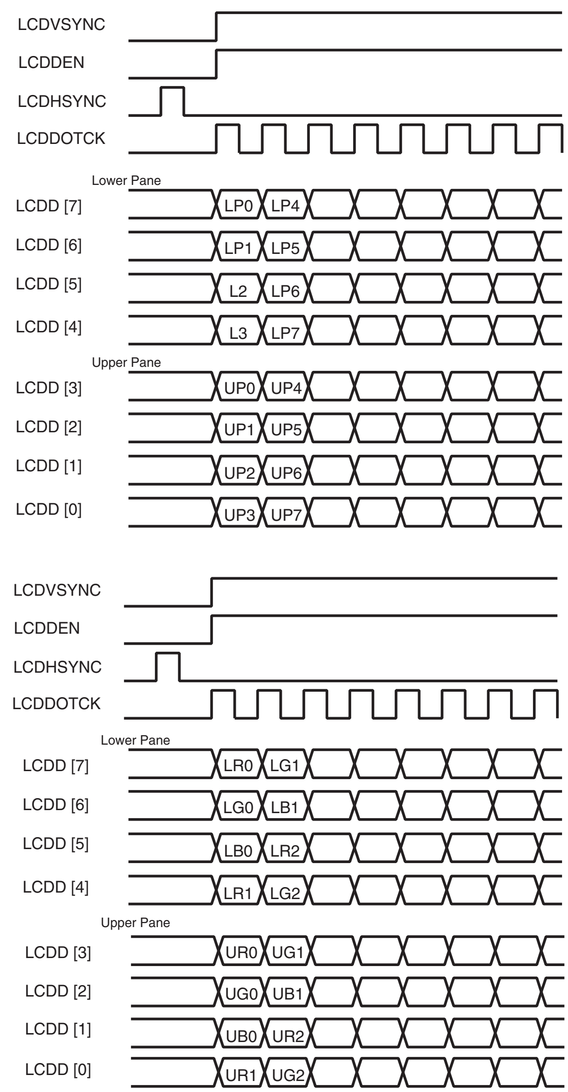
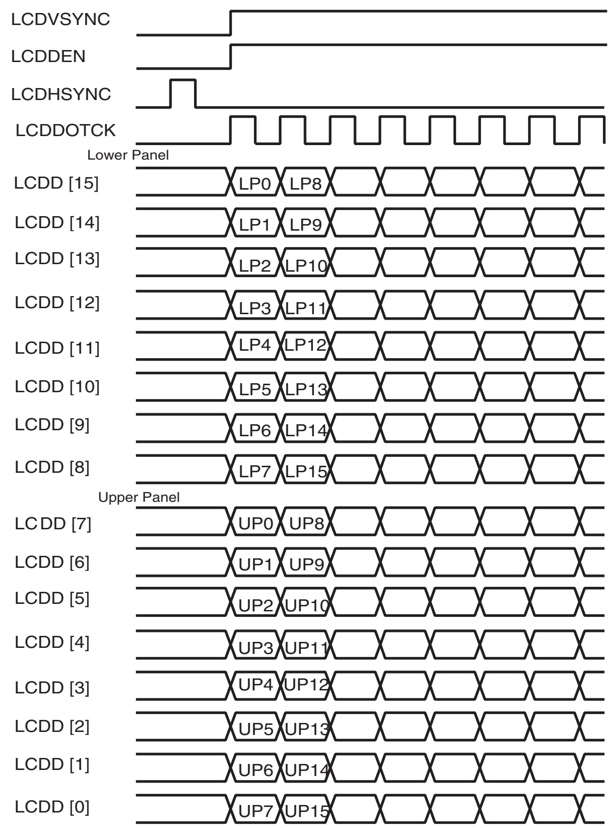


Figure 46-10. Dual Scan Monochrome and Color 16-bit Panel Timing (First Line Expanded View)



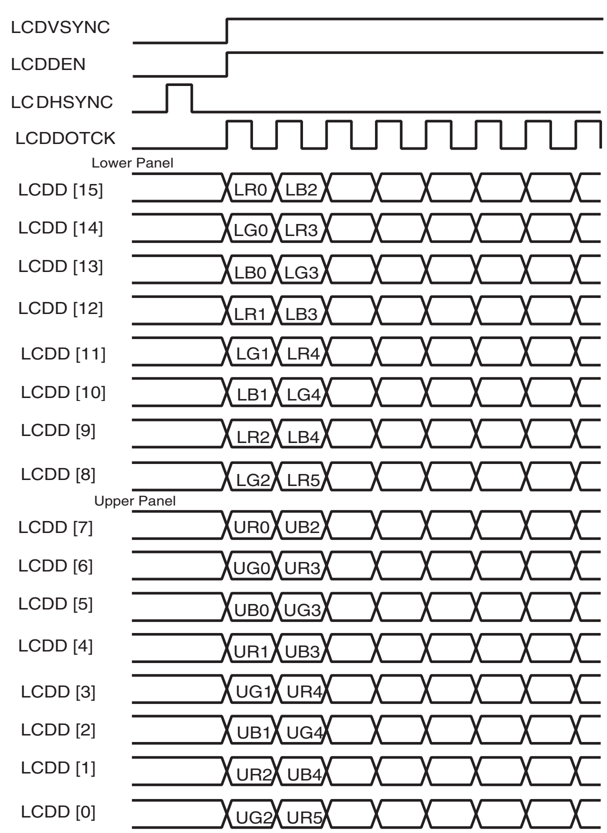


Table 46-15. LCD Signal Multiplexing

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	

46.6 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 998.
- 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 LDC_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 1002.

46.7 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 shows 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 982.
- Configure the LCD Controller Core without enabling it:
  - LCDCON1 register: Program the C*LKVAL* 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 46-14 on page 986. 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 986

– 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 LCDC.

– LCDFRMCFG register: program the dimensions of the LCD module used.

– LCDFIFO register: To program it, use the formula in section "FIFO" on page 980

– 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 999).

- 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.

46.8 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.

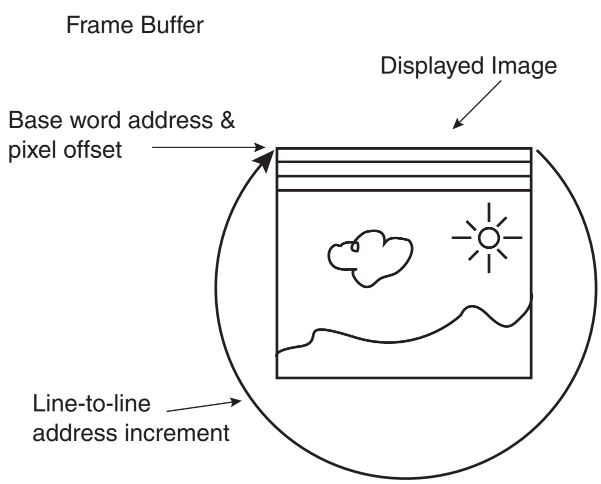
Once the host has updated the Frame Base Address Registers and the next DMA end of frame IRQ arrives, the backbuffer and the primary buffer are swapped and the host can work with the new backbuffer.

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.

46.9 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 46-11. 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.).

46.10 Register Configuration Guide

Program the PIO Controller to enable LCD signals.

Enable the LCD controller clock in the Power Management Controller.

46.10.1 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 MHz/ (2*2.016 MHz)) - 1= 14

LCDCON1 = CLKVAL << 12

LCDCON2 = LITTLEENDIAN | SINGLESCAN | STNCOLOR | DISP8BIT| PS8BPP;

LCDTIM1 = 0;

LCDTIM2 = 10 | (10 << 21);

LCDFRMCFG = (HOZVAL << 21) | LINEVAL;

DMAFRMCFG = (7 << 24) + (320 * 240 * 8) / 32;

46.10.2 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 = [21 MHz..29 MHz] with a typical value = 25.175 MHz.

The Master clock must be (2*(n + 1))*pixel clock frequency

HOZVAL = 640 - 1

LINEVAL = 480 - 1

If Master clock is 50 MHz

CLKVAL = (50 MHz/ (2*25.175 MHz)) - 1= 0

VFP = (12 -1), VBP = (31-1), VPW = (2-1), VHDLY= (2-1)

HFP = (16-2), HBP = (48 -1), HPW = (64-1)

LCDCON1= CLKVAL << 12

LCDCON2 = LITTLEENDIAN | CLKMOD | INVERT_CLK | INVERT_LINE | INVERT_FRM | PS16BPP | SINGLESCAN | TFT

LCDTIM1 = VFP | (VBP << 8) | (VPW << 16) | (VHDLY << 24)

LCDTIM2 = HBP | (HPW << 8) | (HFP << 21)

LCDFRMCFG = (HOZVAL << 21) | LINEVAL

DMAFRMCFG = (7 << 24) + (640 * 480* 16) / 32;

46.11 LCD Controller (LCDC) User Interface

Table 46-16. Register Mapping

Offset	Register	Name	Access	Reset
0x0	DMA Base Address Register 1	DMABADDR1	Read-write	0x00000000
0x4	DMA Base Address Register 2	DMABADDR2	Read-write	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	Read-write	0x00000000
0x1C	DMA Control Register	DMACON	Read-write	0x00000000
0x20	DMA Control Register	DMA2DCFG	Read-write	0x00000000
0x800	LCD Control Register 1	LCDCON1	Read-write	0x00002000
0x804	LCD Control Register 2	LCDCON2	Read-write	0x00000000
0x808	LCD Timing Register 1	LCDTIM1	Read-write	0x00000000
0x80C	LCD Timing Register 2	LCDTIM2	Read-write	0x00000000
0x810	LCD Frame Configuration Register	LCDFRMCFG	Read-write	0x00000000
0x814	LCD FIFO Register	LCDFIFO	Read-write	0x00000000
0x818	Reserved	–	–	–
0x81C	Dithering Pattern DP1_2	DP1_2	Read-write	0xA5
0x820	Dithering Pattern DP4_7	DP4_7	Read-write	0x5AF0FA5
0x824	Dithering Pattern DP3_5	DP3_5	Read-write	0xA5A5F
0x828	Dithering Pattern DP2_3	DP2_3	Read-write	0xA5F
0x82C	Dithering Pattern DP5_7	DP5_7	Read-write	0xFAF5FA5
0x830	Dithering Pattern DP3_4	DP3_4	Read-write	0xFAF5
0x834	Dithering Pattern DP4_5	DP4_5	Read-write	0xFAF5F
0x838	Dithering Pattern DP6_7	DP6_7	Read-write	0xF5FFAFF
0x83C	Power Control Register	PWRCON	Read-write	0x0000000e
0x840	Contrast Control Register	CONTRAST_CTR	Read-write	0x00000000
0x844	Contrast Value Register	CONTRAST_VAL	Read-write	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
0xC00	Palette entry 0	LUT ENTRY 0	Read-write	

Table 46-16. Register Mapping (Continued)

Offset	Register	Name	Access	Reset
0xC04	Palette entry 1	LUT ENTRY 1	Read-write	
0xC08	Palette entry 2	LUT ENTRY 2	Read-write	
0xC0C	Palette entry 3	LUT ENTRY 3	Read-write	
...		...		
0xFFC	Palette entry 255	LUT ENTRY 255	Read-write	

46.11.1 DMA Base Address Register 1

Name: DMABADDR1

Address: 0x00500000

Access: Read-write

Reset: 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						0	0

- **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.

46.11.2 DMA Base Address Register 2

Name: DMABADDR2

Address: 0x00500004

Access: Read-write

Reset: 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.

46.11.3 DMA Frame Pointer Register 1

Name: DMAFRMPT1

Address: 0x00500008

Access: Read-only

Reset: 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.

46.11.4 DMA Frame Pointer Register 2

Name: DMAFRMPT2

Address: 0x0050000C

Access: Read-only

Reset: 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.

46.11.5 DMA Frame Address Register 1

Name: DMAFRMADD1

Address: 0x00500010

Access: Read-only

Reset: 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.

46.11.6 DMA Frame Address Register 2

Name: DMAFRMADD2

Address: 0x00500014

Access: Read-only

Reset: 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.

46.11.7 DMA Frame Configuration Register

Name: DMAFRMCFG

Address: 0x00500018

Access: Read-write

Reset: 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

46.11.8 DMA Control Register

Name: DMACON

Address: 0x0050001C

Access: Read-write

Reset: 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.

46.11.9 LCD DMA 2D Addressing Register

Name: DMA2DCFG

Address: 0x00500020

Access: Read-write

Reset: 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.

46.11.10 LCD Control Register 1

Name: LCDCON1

Address: 0x00500800

Access: Read-write, except LINECNT: Read-only

Reset: 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.

46.11.11 LCD Control Register 2

Name: LCDCON2

Address: 0x00500804

Access: Read-write

Reset: 0x0000000

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	INVLINE	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)

- **INVLINE: 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

46.11.12 LCD Timing Configuration Register 1

Name: LCDTIM1

Address: 0x00500808

Access: Read-write

Reset: 0x0000000

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.

46.11.13 LCD Timing Configuration Register 2

Name: LCDTIM2

Address: 0x0050080C

Access: Read-write

Reset: 0x0000000

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+2) LCDDOTCK cycles.

46.11.14 LCD Frame Configuration Register

Name: LCDFRMCFG

Address: 0x00500810

Access: Read-write

Reset: 0x0000000

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

- **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**

46.11.15 LCD FIFO Register

Name: LCDFIFO

Address: 0x00500814

Access: Read-write

Reset: 0x0000000

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:

FIFOTH = 2048 - (2 x 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 1007.

46.11.16 Dithering Pattern DP1_2 Register

Name: DP1_2

Address: 0x0050081C

Access: Read-write

Reset: 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 ½ duty cycle**

46.11.17 Dithering Pattern DP4_7 Register

Name: DP4_7

Address: 0x00500820

Access: Read-write

Reset: 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**

46.11.18 Dithering Pattern DP3_5 Register

Name: DP3_5

Address: 0x00500824

Access: Read-write

Reset: 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**

46.11.19 Dithering Pattern DP2_3 Register

Name: DP2_3: Dithering Pattern DP2_3 Register

Address: 0x00500828

Access: Read-write

Reset: 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**

46.11.20 Dithering Pattern DP5_7 Register

Name: DP5_7:

Address: 0x0050082C

Access: Read-write

Reset: 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**

46.11.21 Dithering Pattern DP3_4 Register

Name: DP3_4

Address: 0x00500830

Access: Read-write

Reset: 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**

46.11.22 Dithering Pattern DP4_5 Register

Name: DP4_5

Address: 0x00500834

Access: Read-write

Reset: 0xFAF5FF

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**

46.11.23 Dithering Pattern DP6_7 Register

Name: DP6_7

Address: 0x00500838

Access: Read-write

Reset: 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**

46.11.24 Power Control Register

Name: PWRCON

Address: 0x0050083C

Access: Read-write

Reset: 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).

46.11.25 Contrast Control Register

Name: CONTRAST_CTR

Address: 0x00500840

Access: Read-write

Reset: 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 fCOUNTER = fLCDC_CLOCK.
0	1	The counter advances at a rate of fCOUNTER = fLCDC_CLOCK/2.
1	0	The counter advances at a rate of fCOUNTER = fLCDC_CLOCK/4.
1	1	The counter advances at a rate of fCOUNTER = fLCDC_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.

46.11.26 Contrast Value Register

Name: CONTRAST_VAL

Address: 0x00500844

Access: Read-write

Reset: 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.

46.11.27 LCD Interrupt Enable Register

Name: LCD_IER

Address: 0x00500848

Access: Write-only

Reset: 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

46.11.28 LCD Interrupt Disable Register

Name: LCD_IDR

Address: 0x0050084C

Access: Write-only

Reset: 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

46.11.29 LCD Interrupt Mask Register

Name: LCD_IMR

Address: 0x00500850

Access: Read-only

Reset: 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

46.11.30 LCD Interrupt Status Register

Name: LCD_ISR

Address: 0x00500854

Access: Read-only

Reset: 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

46.11.31 LCD Interrupt Clear Register

Name: LCD_ICR

Address: 0x00500858

Access: Write-only

Reset: 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

46.11.32 LCD Interrupt Test Register

Name: LCD_ITR

Address: 0x00500860

Access: Write-only

Reset: 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

46.11.33 LCD Interrupt Raw Status Register

Name: LCD_IRR

Address: 0x00500864

Access: Write-only

Reset: 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

47. Image Sensor Interface (ISI)

47.1 Overview

The Image Sensor Interface (ISI) connects a CMOS-type image sensor to the processor and provides image capture in various formats. It does data conversion, if necessary, before the storage in memory through DMA.

The ISI supports color CMOS image sensor and grayscale image sensors with a reduced set of functionalities.

In grayscale mode, the data stream is stored in memory without any processing and so is not compatible with the LCD controller.

Internal FIFOs on the preview and codec paths are used to store the incoming data. The RGB output on the preview path is compatible with the LCD controller. This module outputs the data in RGB format (LCD compatible) and has scaling capabilities to make it compliant to the LCD display resolution (See Table 47-3 on page 1034).

Several input formats such as preprocessed RGB or YCbCr are supported through the data bus interface.

It supports two modes of synchronization:

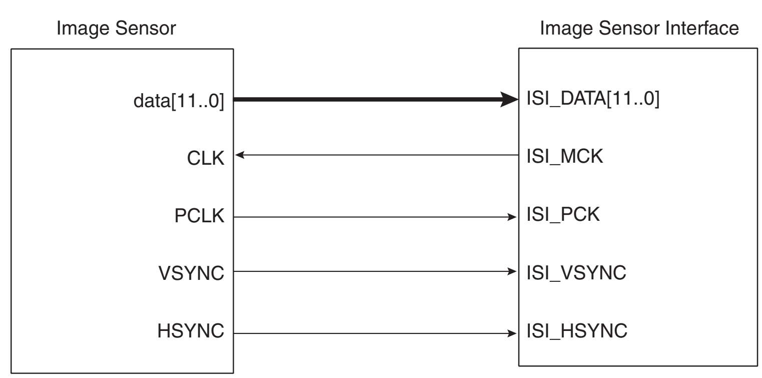
1. The hardware with ISI_VSYNC and ISI_HSYNC signals
2. The International Telecommunication Union Recommendation *ITU-R BT.656-4* Start-of-Active-Video (SAV) and End-of-Active-Video (EAV) synchronization sequence.

Using EAV/SAV for synchronization reduces the pin count (ISI_VSYNC, ISI_HSYNC not used). The polarity of the synchronization pulse is programmable to comply with the sensor signals.

Table 47-1. I/O Description

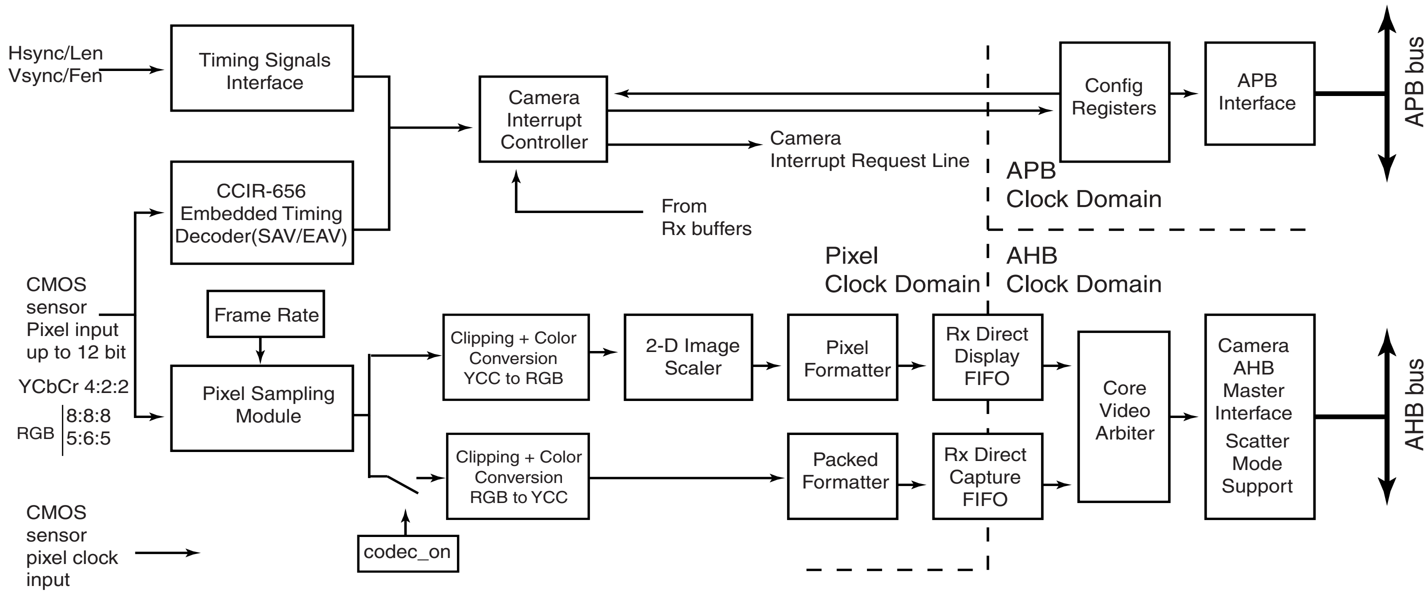
Signal	Dir	Description
ISI_VSYNC	IN	Vertical Synchronization
ISI_HSYNC	IN	Horizontal Synchronization
ISI_DATA[11..0]	IN	Sensor Pixel Data
ISI_MCK	OUT	Master Clock Provided to the Image Sensor
ISI_PCK	IN	Pixel Clock Provided by the Image Sensor

Figure 47-1. ISI Connection Example



47.2 Block Diagram

Figure 47-2. Image Sensor Interface Block Diagram



47.3 Functional Description

The Image Sensor Interface (ISI) supports direct connection to the ITU-R BT. 601/656 8-bit mode compliant sensors and up to 12-bit grayscale sensors. It receives the image data stream from the image sensor on the 12-bit data bus.

This module receives up to 12 bits for data, the horizontal and vertical synchronizations and the pixel clock. The reduced pin count alternative for synchronization is supported for sensors that embed SAV (start of active video) and EAV (end of active video) delimiters in the data stream.

The Image Sensor Interface interrupt line is generally connected to the Advanced Interrupt Controller and can trigger an interrupt at the beginning of each frame and at the end of a DMA frame transfer. If the SAV/EAV synchronization is used, an interrupt can be triggered on each delimiter event.

For 8-bit color sensors, the data stream received can be in several possible formats: YCbCr 4:2:2, RGB 8:8:8, RGB 5:6:5 and may be processed before the storage in memory. The data stream may be sent on both preview path and codec path if the bit CODEC_ON in the ISI_CR1 is one. To optimize the bandwidth, the codec path should be enabled only when a capture is required.

In grayscale mode, the input data stream is stored in memory without any processing. The 12-bit data, which represent the grayscale level for the pixel, is stored in memory one or two pixels per word, depending on the GS_MODE bit in the ISI_CR2 register. The data is stored via the preview path without any treatment (scaling, color conversion,…). The size of the sensor must be programmed in the fields IM_VSIZE and IM_HSIZE in the ISI_CR2 register.The programming of the preview path register (ISI_PSIZE) is not necessary. The codec datapath is not available when grayscale image is selected.

A frame rate counter allows users to capture all frames or 1 out of every 2 to 8 frames.

47.3.1 Data Timing

The two data timings using horizontal and vertical synchronization and EAV/SAV sequence synchronization are shown in Figure 47-3 and Figure 47-4.

In the VSYNC/HSYNC synchronization, the valid data is captured with the active edge of the pixel clock (ISI_PCK), after SFD lines of vertical blanking and SLD pixel clock periods delay programmed in the control register.

The ITU-RBT.656-4 defines the functional timing for an 8-bit wide interface.

There are two timing reference signals, one at the beginning of each video data block SAV (0xFF000080) and one at the end of each video data block EAV(0xFF00009D). Only data sent between EAV and SAV is captured. Horizontal blanking and vertical blanking are ignored. Use of the SAV and EAV synchronization eliminates the ISI_VSYNC and ISI_HSYNC signals from the interface, thereby reducing the pin count. In order to retrieve both frame and line synchronization properly, at least one line of vertical blanking is mandatory.

Figure 47-3. HSYNC and VSYNC Synchronization

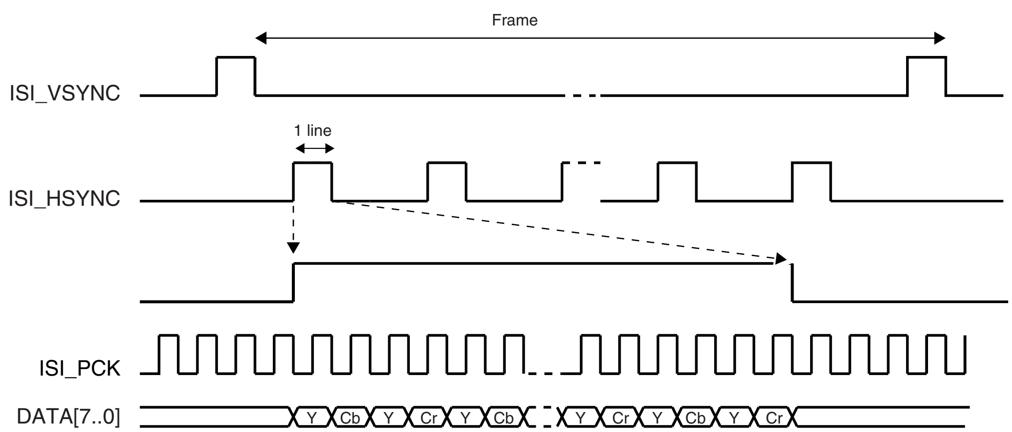
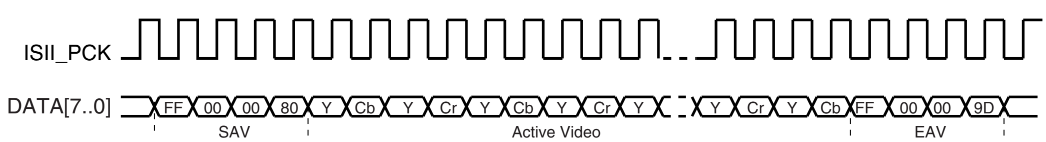


Figure 47-4. SAV and EAV Sequence Synchronization



47.3.2 Data Ordering

The RGB color space format is required for viewing images on a display screen preview, and the YCbCr color space format is required for encoding.

All the sensors do not output the YCbCr or RGB components in the same order. The ISI allows the user to program the same component order as the sensor, reducing software treatments to restore the right format.

Table 47-2. Data Ordering in YCbCr Mode

Mode	Byte 0	Byte 1	Byte 2	Byte 3
Default	Cb(i)	Y(i)	Cr(i)	Y(i+1)
Mode1	Cr(i)	Y(i)	Cb(i)	Y(i+1)
Mode2	Y(i)	Cb(i)	Y(i+1)	Cr(i)
Mode3	Y(i)	Cr(i)	Y(i+1)	Cb(i)

Table 47-3. RGB Format in Default Mode, RGB_CFG = 00, No Swap

Mode	Byte	D7	D6	D5	D4	D3	D2	D1	D0
RGB 8:8:8	Byte 0	R7(i)	R6(i)	R5(i)	R4(i)	R3(i)	R2(i)	R1(i)	R0(i)
	Byte 1	G7(i)	G6(i)	G5(i)	G4(i)	G3(i)	G2(i)	G1(i)	G0(i)
	Byte 2	B7(i)	B6(i)	B5(i)	B4(i)	B3(i)	B2(i)	B1(i)	B0(i)
	Byte 3	R7(i+1)	R6(i+1)	R5(i+1)	R4(i+1)	R3(i+1)	R2(i+1)	R1(i+1)	R0(i+1)
RGB 5:6:5	Byte 0	R4(i)	R3(i)	R2(i)	R1(i)	R0(i)	G5(i)	G4(i)	G3(i)
	Byte 1	G2(i)	G1(i)	G0(i)	B4(i)	B3(i)	B2(i)	B1(i)	B0(i)
	Byte 2	R4(i+1)	R3(i+1)	R2(i+1)	R1(i+1)	R0(i+1)	G5(i+1)	G4(i+1)	G3(i+1)
	Byte 3	G2(i+1)	G1(i+1)	G0(i+1)	B4(i+1)	B3(i+1)	B2(i+1)	B1(i+1)	B0(i+1)

Table 47-4. RGB Format, RGB_CFG = 10 (Mode 2), No Swap

Mode	Byte	D7	D6	D5	D4	D3	D2	D1	D0
RGB 5:6:5	Byte 0	G2(i)	G1(i)	G0(i)	R4(i)	R3(i)	R2(i)	R1(i)	R0(i)
	Byte 1	B4(i)	B3(i)	B2(i)	B1(i)	B0(i)	G5(i)	G4(i)	G3(i)
	Byte 2	G2(i+1)	G1(i+1)	G0(i+1)	R4(i+1)	R3(i+1)	R2(i+1)	R1(i+1)	R0(i+1)
	Byte 3	B4(i+1)	B3(i+1)	B2(i+1)	B1(i+1)	B0(i+1)	G5(i+1)	G4(i+1)	G3(i+1)

Table 47-5. RGB Format in Default Mode, RGB_CFG = 00, Swap Activated

Mode	Byte	D7	D6	D5	D4	D3	D2	D1	D0
RGB 8:8:8	Byte 0	R0(i)	R1(i)	R2(i)	R3(i)	R4(i)	R5(i)	R6(i)	R7(i)
	Byte 1	G0(i)	G1(i)	G2(i)	G3(i)	G4(i)	G5(i)	G6(i)	G7(i)
	Byte 2	B0(i)	B1(i)	B2(i)	B3(i)	B4(i)	B5(i)	B6(i)	B7(i)
	Byte 3	R0(i+1)	R1(i+1)	R2(i+1)	R3(i+1)	R4(i+1)	R5(i+1)	R6(i+1)	R7(i+1)
RGB 5:6:5	Byte 0	G3(i)	G4(i)	G5(i)	R0(i)	R1(i)	R2(i)	R3(i)	R4(i)
	Byte 1	B0(i)	B1(i)	B2(i)	B3(i)	B4(i)	G0(i)	G1(i)	G2(i)
	Byte 2	G3(i+1)	G4(i+1)	G5(i+1)	R0(i+1)	R1(i+1)	R2(i+1)	R3(i+1)	R4(i+1)
	Byte 3	B0(i+1)	B1(i+1)	B2(i+1)	B3(i+1)	B4(i+1)	G0(i+1)	G1(i+1)	G2(i+1)

The RGB 5:6:5 input format is processed to be displayed as RGB 5:5:5 format, compliant with the 16-bit mode of the LCD controller.

47.3.3 Clocks

The sensor master clock (ISI_MCK) can be generated either by the Advanced Power Management Controller (APMC) through a Programmable Clock output or by an external oscillator connected to the sensor.

None of the sensors embeds a power management controller, so providing the clock by the APMC is a simple and efficient way to control power consumption of the system.

Care must be taken when programming the system clock. The ISI has two clock domains, the system bus clock and the pixel clock provided by sensor. The two clock domains are not synchronized, but the system clock must be faster than pixel clock.

47.3.4 Preview Path

47.3.4.1 Scaling, Decimation (Subsampling)

This module resizes captured 8-bit color sensor images to fit the LCD display format. The resize module performs only downscaling. The same ratio is applied for both horizontal and vertical resize, then a fractional decimation algorithm is applied.

The decimation factor is a multiple of 1/16 and values 0 to 15 are forbidden.

Table 47-6. Decimation Factor

Dec value	0->15	16	17	18	19	...	124	125	126	127
Dec Factor	X	1	1.063	1.125	1.188	...	7.750	7.813	7.875	7.938

Table 47-7. Decimation and Scaler Offset Values

OUTPUT / INPUT		352*288	640*480	800*600	1280*1024	1600*1200	2048*1536
VGA 640*480	F	NA	16	20	32	40	51
QVGA 320*240	F	16	32	40	64	80	102
CIF 352*288	F	16	26	33	56	66	85
QCIF 176*144	F	16	53	66	113	133	170

Example:

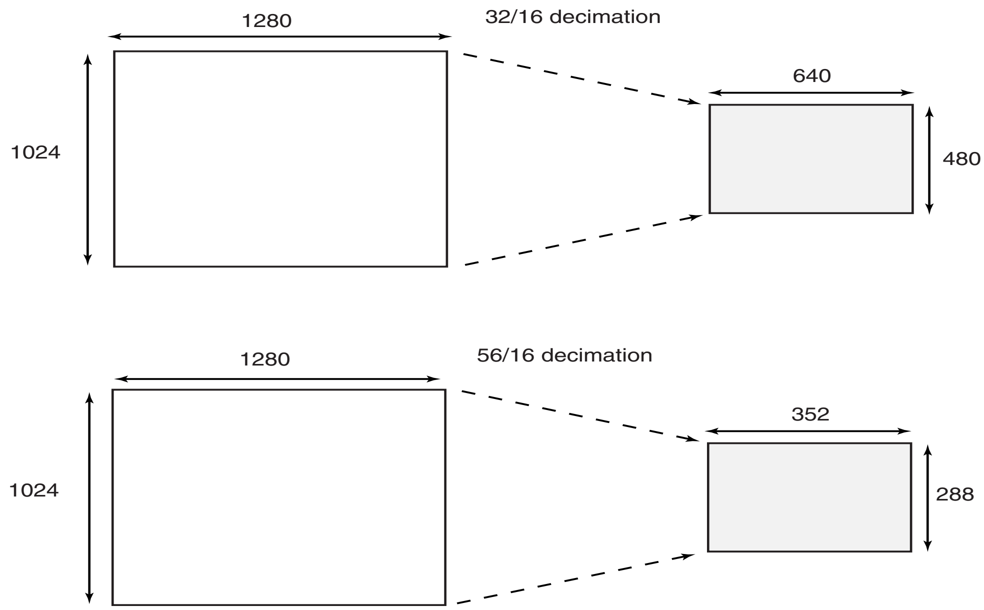
Input 1280*1024 Output=640*480

Hratio = 1280/640 =2

Vratio = 1024/480 =2.1333

The decimation factor is 2 so 32/16.

Figure 47-5. Resize Examples



47.3.4.2 Color Space Conversion

This module converts YCrCb or YUV pixels to RGB color space. Clipping is performed to ensure that the samples value do not exceed the allowable range. The conversion matrix is defined below and is fully programmable:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} C_0 & 0 & C_1 \\ C_0 & -C_2 & -C_3 \\ C_0 & C_4 & 0 \end{bmatrix} \times \begin{bmatrix} Y - Y_{off} \\ C_b - C_{boff} \\ C_r - C_{roff} \end{bmatrix}$$

Example of programmable value to convert YCrCb to RGB:

$$\begin{cases} R = 1{,}164 \cdot (Y - 16) + 1{,}596 \cdot (C_r - 128) \\ G = 1{,}164 \cdot (Y - 16) - 0{,}813 \cdot (C_r - 128) - 0{,}392 \cdot (C_b - 128) \\ B = 1{,}164 \cdot (Y - 16) + 2{,}107 \cdot (C_b - 128) \end{cases}$$

An example of programmable value to convert from YUV to RGB:

$$\begin{cases} R = Y + 1{,}596 \cdot V \\ G = Y - 0{,}394 \cdot U - 0{,}436 \cdot V \\ B = Y + 2{,}032 \cdot U \end{cases}$$

47.3.4.3 Memory Interface

Preview datapath contains a data formatter that converts 8:8:8 pixel to RGB 5:5:5 format compliant with 16-bit format of the LCD controller. In general, when converting from a color channel with more bits to one with fewer bits, formatter module discards the lower-order bits. Example: Converting from RGB 8:8:8 to RGB 5:6:5, it discards the three LSBs from the red and blue channels, and two LSBs from the green channel. When grayscale mode is enabled, two memory format are supported. One mode supports 2 pixels per word, and the other mode supports 1 pixel per word.

Table 47-8. Grayscale Memory Mapping Configuration for 12-bit Data

GS_MODE	DATA[31:24]	DATA[23:16]	DATA[15:8]	DATA[7:0]
0	P_0[11:4]	P_0[3:0], 0000	P_1[11:4]	P_1[3:0], 0000
1	P_0[11:4]	P_0[3:0], 0000	0	0

47.3.4.4 FIFO and DMA Features

Both preview and Codec datapaths contain FIFOs, asynchronous buffers that are used to safely transfer formatted pixels from Pixel clock domain to AHB clock domain. A video arbiter is used to manage FIFO thresholds and triggers a relevant DMA request through the AHB master interface. Thus, depending on FIFO state, a specified length burst is asserted. Regarding AHB master interface, it supports Scatter DMA mode through linked list operation. This mode of operation improves flexibility of image buffer location and allows the user to allocate two or more frame buffers. The destination frame buffers are defined by a series of Frame Buffer Descriptors (FBD). Each FBD controls the transfer of one entire frame and then optionally loads a further FBD to switch the DMA operation at another frame buffer address. The FBD is defined by a series of two words. The first one defines the current frame buffer address, and the second defines the next FBD memory location. This DMA transfer mode is only available for preview datapath and is configured in the ISI_PPFBD register that indicates the memory location of the first FBD.

The primary FBD is programmed into the camera interface controller. The data to be transferred described by an FBD requires several burst access. In the example below, the use of 2 ping-pong frame buffers is described.

47.3.4.5 Example

The first FBD, stored at address 0x30000, defines the location of the first frame buffer.

Destination Address: frame buffer ID0 0x02A000

Next FBD address: 0x30010

Second FBD, stored at address 0x30010, defines the location of the second frame buffer.

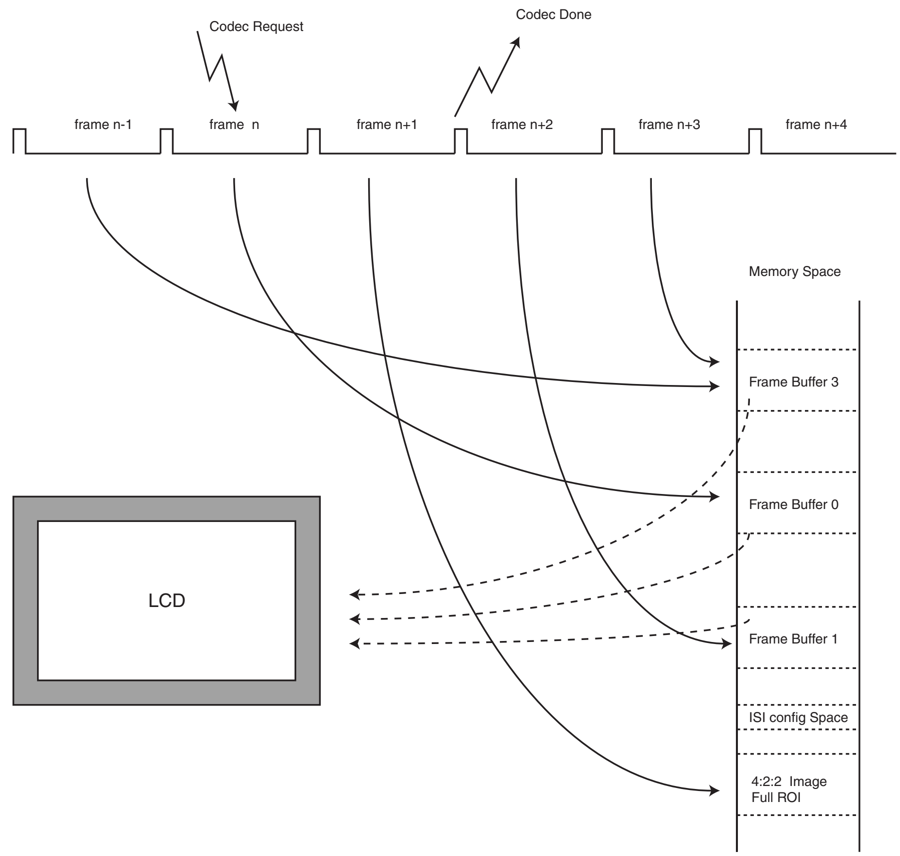
Destination Address: frame buffer ID1 0x3A000

Transfer width: 32 bit

Next FBD address: 0x30000, wrapping to first FBD.

Using this technique, several frame buffers can be configured through the linked list. Figure 47-6 illustrates a typical three frame buffer application. Frame n is mapped to frame buffer 0, frame n+1 is mapped to frame buffer 1, frame n+2 is mapped to Frame buffer 2, further frames wrap. A codec request occurs, and the full-size 4:2:2 encoded frame is stored in a dedicated memory space.

Figure 47-6. Three Frame Buffers Application and Memory Mapping



47.3.5 Codec Path

47.3.5.1 Color Space Conversion

Depending on user selection, this module can be bypassed so that input YCrCb stream is directly connected to the format converter module. If the RGB input stream is selected, this module converts RGB to YCrCb color space with the formulas given below:

$$\begin{bmatrix} Y \\ C_r \\ C_b \end{bmatrix} = \begin{bmatrix} C_0 & C_1 & C_2 \\ C_3 & -C_4 & -C_5 \\ -C_6 & -C_7 & C_8 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} Y_{off} \\ Cr_{off} \\ Cb_{off} \end{bmatrix}$$

An example of coefficients are given below:

$$\begin{cases} Y = 0{,}257 \cdot R + 0{,}504 \cdot G + 0{,}098 \cdot B + 16 \\ C_r = 0{,}439 \cdot R - 0{,}368 \cdot G - 0{,}071 \cdot B + 128 \\ C_b = -0{,}148 \cdot R - 0{,}291 \cdot G + 0{,}439 \cdot B + 128 \end{cases}$$

47.3.5.2 Memory Interface

Dedicated FIFO are used to support packed memory mapping. YCrCb pixel components are sent in a single 32-bit word in a contiguous space (packed). Data is stored in the order of natural scan lines. Planar mode is not supported.

47.3.5.3 DMA Features

Unlike preview datapath, codec datapath DMA mode does not support linked list operation. Only the CODEC_DMA_ADDR register is used to configure the frame buffer base address.

47.4 Image Sensor Interface (ISI) User Interface

Table 47-9. Register Mapping

Offset	Register	Name	Access	Reset
0x00	ISI Control 1 Register	ISI_CR1	Read-write	0x00000002
0x04	ISI Control 2 Register	ISI_CR2	Read-write	0x00000000
0x08	ISI Status Register	ISI_SR	Read	0x00000000
0x0C	ISI Interrupt Enable Register	ISI_IER	Read-write	0x00000000
0x10	ISI Interrupt Disable Register	ISI_IDR	Read-write	0x00000000
0x14	ISI Interrupt Mask Register	ISI_IMR	Read-write	0x00000000
0x18	Reserved	-	-	-
0x1C	Reserved	-	-	-
0x20	ISI Preview Size Register	ISI_PSIZE	Read-write	0x00000000
0x24	ISI Preview Decimation Factor Register	ISI_PDECF	Read-write	0x00000010
0x28	ISI Preview Primary FBD Register	ISI_PPFBD	Read-write	0x00000000
0x2C	ISI Codec DMA Base Address Register	ISI_CDBA	Read-write	0x00000000
0x30	ISI CSC YCrCb To RGB Set 0 Register	ISI_Y2R_SET0	Read-write	0x6832cc95
0x34	ISI CSC YCrCb To RGB Set 1 Register	ISI_Y2R_SET1	Read-write	0x00007102
0x38	ISI CSC RGB To YCrCb Set 0 Register	ISI_R2Y_SET0	Read-write	0x01324145
0x3C	ISI CSC RGB To YCrCb Set 1 Register	ISI_R2Y_SET1	Read-write	0x01245e38
0x40	ISI CSC RGB To YCrCb Set 2 Register	ISI_R2Y_SET2	Read-write	0x01384a4b
0x44-0xF8	Reserved	–	–	–
0xFC	Reserved	–	–	–

Note: Several parts of the ISI controller use the pixel clock provided by the image sensor (ISI_PCK). Thus the user must first program the image sensor to provide this clock (ISI_PCK) before programming the Image Sensor Controller.

47.4.1 ISI Control 1 Register

Name: ISI_CR1

Address: 0xFFFC4000

Access: Read-write

Reset: 0x00000002

31	30	29	28	27	26	25	24
SFD							

23	22	21	20	19	18	17	16
SLD							

15	14	13	12	11	10	9	8
CODEC_ON	THMASK		FULL	-	FRATE		

7	6	5	4	3	2	1	0
CRC_SYNC	EMB_SYNC	-	PIXCLK_POL	VSYNC_POL	HSYNC_POL	ISI_DIS	ISI_RST

- **ISI_RST: Image sensor interface reset**

Write-only. Refer to bit SOFTRST in Section 47.4.3 “ISI Status Register” on page 1046 for soft reset status.

0: No action

1: Resets the image sensor interface.

- **ISI_DIS: Image sensor disable:**

0: Enable the image sensor interface.

1: Finish capturing the current frame and then shut down the module.

- **HSYNC_POL: Horizontal synchronization polarity**

0: HSYNC active high

1: HSYNC active low

- **VSYNC_POL: Vertical synchronization polarity**

0: VSYNC active high

1: VSYNC active low

- **PIXCLK_POL: Pixel clock polarity**

0: Data is sampled on rising edge of pixel clock

1: Data is sampled on falling edge of pixel clock

- **EMB_SYNC: Embedded synchronization**

0: Synchronization by HSYNC, VSYNC

1: Synchronization by embedded synchronization sequence SAV/EAV

- **CRC_SYNC: Embedded synchronization**

0: No CRC correction is performed on embedded synchronization

1: CRC correction is performed. if the correction is not possible, the current frame is discarded and the CRC_ERR is set in the status register.

- **FRATE: Frame rate [0..7]**

0: All the frames are captured, else one frame every FRATE+1 is captured.

- **FULL: Full mode is allowed**

1: Both codec and preview datapaths are working simultaneously

- **THMASK: Threshold mask**

0: 4, 8 and 16 AHB bursts are allowed

1: 8 and 16 AHB bursts are allowed

2: Only 16 AHB bursts are allowed

- **CODEC_ON: Enable the codec path enable bit**

Write-only.

0: The codec path is disabled

1: The codec path is enabled and the next frame is captured. Refer to bit CDC_PND in "ISI Status Register" on page 1046.

- **SLD: Start of Line Delay**

SLD pixel clock periods to wait before the beginning of a line.

- **SFD: Start of Frame Delay**

SFD lines are skipped at the beginning of the frame.

47.4.2 ISI Control 2 Register

Name: ISI_CR2

Address: 0xFFFC4004

Access: Read-write

Reset: 0x0

31	30	29	28	27	26	25	24
RGB_CFG		YCC_SWAP		-	IM_HSIZE		

23	22	21	20	19	18	17	16
IM_HSIZE							

15	14	13	12	11	10	9	8
COL_SPACE	RGB_SWAP	GRAYSCALE	RGB_MODE	GS_MODE	IM_VSIZE		

7	6	5	4	3	2	1	0
IM_VSIZE							

- **IM_VSIZE: Vertical size of the Image sensor [0..2047]**

Vertical size = IM_VSIZE + 1

- **GS_MODE**

0: 2 pixels per word

1: 1 pixel per word

- **RGB_MODE: RGB input mode**

0: RGB 8:8:8 24 bits

1: RGB 5:6:5 16 bits

- **GRAYSCALE**

0: Grayscale mode is disabled

1: Input image is assumed to be grayscale coded

- **RGB_SWAP**

0: D7 -> R7

1: D0 -> R7

The RGB_SWAP has no effect when the grayscale mode is enabled.

- **COL_SPACE: Color space for the image data**

0: YCbCr

1: RGB

- **IM_HSIZE: Horizontal size of the Image sensor [0..2047]**

Horizontal size = IM_HSIZE + 1

- **YCC_SWAP: Defines the YCC image data**

YCC_SWAP	Byte 0	Byte 1	Byte 2	Byte 3
00: Default	Cb(i)	Y(i)	Cr(i)	Y(i+1)
01: Mode1	Cr(i)	Y(i)	Cb(i)	Y(i+1)
10: Mode2	Y(i)	Cb(i)	Y(i+1)	Cr(i)
11: Mode3	Y(i)	Cr(i)	Y(i+1)	Cb(i)

- **RGB_CFG: Defines RGB pattern when RGB_MODE is set to 1**

RGB_CFG	Byte 0	Byte 1	Byte 2	Byte 3
00: Default	R/G(MSB)	G(LSB)/B	R/G(MSB)	G(LSB)/B
01: Mode1	B/G(MSB)	G(LSB)/R	B/G(MSB)	G(LSB)/R
10: Mode2	G(LSB)/R	B/G(MSB)	G(LSB)/R	B/G(MSB)
11: Mode3	G(LSB)/B	R/G(MSB)	G(LSB)/B	R/G(MSB)

If RGB_MODE is set to RGB 8:8:8, then RGB_CFG = 0 implies RGB color sequence, else it implies BGR color sequence.

47.4.3 ISI Status Register

Name: ISI_SR

Address: 0xFFFC4008

Access: Read

Reset: 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
–	–	–	–	–	–	FR_OVR	FO_C_EMP

7	6	5	4	3	2	1	0
FO_P_EMP	FO_P_OVF	FO_C_OVF	CRC_ERR	CDC_PND	SOFTRST	DIS	SOF

- **SOF: Start of frame**

0: No start of frame has been detected.

1: A start of frame has been detected.

- **DIS: Image Sensor Interface disable**

0: The image sensor interface is enabled.

1: The image sensor interface is disabled and stops capturing data. The DMA controller and the core can still read the FIFOs.

- **SOFTRST: Software reset**

0: Software reset not asserted or not completed.

1: Software reset has completed successfully.

- **CDC_PND: Codec request pending**

0: No request asserted.

1: A codec request is pending. If a codec request is asserted during a frame, the CDC_PND bit rises until the start of a new frame. The capture is completed when the flag FO_C_EMP = 1.

- **CRC_ERR: CRC synchronization error**

0: No crc error in the embedded synchronization frame (SAV/EAV)

1: The CRC_SYNC is enabled in the control register and an error has been detected and not corrected. The frame is discarded and the ISI waits for a new one.

- **FO_C_OVF: FIFO codec overflow**

0: No overflow

1: An overrun condition has occurred in input FIFO on the codec path. The overrun happens when the FIFO is full and an attempt is made to write a new sample to the FIFO.

- **FO_P_OVF: FIFO preview overflow**

0: No overflow

1: An overrun condition has occurred in input FIFO on the preview path. The overrun happens when the FIFO is full and an attempt is made to write a new sample to the FIFO.

- **FO_P_EMP**

0:The DMA has not finished transferring all the contents of the preview FIFO.

1:The DMA has finished transferring all the contents of the preview FIFO.

- **FO_C_EMP**

0: The DMA has not finished transferring all the contents of the codec FIFO.

1: The DMA has finished transferring all the contents of the codec FIFO.

- **FR_OVR: Frame rate overrun**

0: No frame overrun.

1: Frame overrun, the current frame is being skipped because a vsync signal has been detected while flushing FIFOs.

47.4.4 Interrupt Enable Register

Name: ISI_IER

Address: 0xFFFC400C

Access: Read-write

Reset: 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
–	–	–	–	–	–	FR_OVR	FO_C_EMP

7	6	5	4	3	2	1	0
FO_P_EMP	FO_P_OVF	FO_C_OVF	CRC_ERR	–	SOFTRST	DIS	SOF

- **SOF: Start of Frame**

1: Enables the Start of Frame interrupt.

- **DIS: Image Sensor Interface disable**

1: Enables the DIS interrupt.

- **SOFTRST: Soft Reset**

1: Enables the Soft Reset Completion interrupt.

- **CRC_ERR: CRC synchronization error**

1: Enables the CRC_SYNC interrupt.

- **FO_C_OVF: FIFO codec Overflow**

1: Enables the codec FIFO overflow interrupt.

- **FO_P_OVF: FIFO preview Overflow**

1: Enables the preview FIFO overflow interrupt.

- **FO_P_EMP**

1: Enables the preview FIFO empty interrupt.

- **FO_C_EMP**

1: Enables the codec FIFO empty interrupt.

- **FR_OVR: Frame overrun**

1: Enables the Frame overrun interrupt.

47.4.5 ISI Interrupt Disable Register

Name: ISI_IDR

Address: 0xFFFC4010

Access: Read-write

Reset: 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
–	–	–	–	–	–	FR_OVR	FO_C_EMP

7	6	5	4	3	2	1	0
FO_P_EMP	FO_P_OVF	FO_C_OVF	CRC_ERR	–	SOFTRST	DIS	SOF

- **SOF: Start of Frame**

1: Disables the Start of Frame interrupt.

- **DIS: Image Sensor Interface disable**

1: Disables the DIS interrupt.

- **SOFTRST**

1: Disables the soft reset completion interrupt.

- **CRC_ERR: CRC synchronization error**

1: Disables the CRC_SYNC interrupt.

- **FO_C_OVF: FIFO codec overflow**

1: Disables the codec FIFO overflow interrupt.

- **FO_P_OVF: FIFO preview overflow**

1: Disables the preview FIFO overflow interrupt.

- **FO_P_EMP**

1: Disables the preview FIFO empty interrupt.

- **FO_C_EMP**

1: Disables the codec FIFO empty interrupt.

- **FR_OVR**

1: Disables frame overrun interrupt.

47.4.6 ISI Interrupt Mask Register

Name: ISI_IMR

Address: 0xFFFC4014

Access: Read-write

Reset: 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
–	–	–	–	–	–	FR_OVR	FO_C_EMP

7	6	5	4	3	2	1	0
FO_P_EMP	FO_P_OVF	FO_C_OVF	CRC_ERR	–	SOFTRST	DIS	SOF

- **SOF: Start of Frame**

0: The Start of Frame interrupt is disabled.

1: The Start of Frame interrupt is enabled.

- **DIS: Image sensor interface disable**

0: The DIS interrupt is disabled.

1: The DIS interrupt is enabled.

- **SOFTRST**

0: The soft reset completion interrupt is enabled.

1: The soft reset completion interrupt is disabled.

- **CRC_ERR: CRC synchronization error**

0: The CRC_SYNC interrupt is disabled.

1: The CRC_SYNC interrupt is enabled.

- **FO_C_OVF: FIFO codec overflow**

0: The codec FIFO overflow interrupt is disabled.

1: The codec FIFO overflow interrupt is enabled.

- **FO_P_OVF: FIFO preview overflow**

0: The preview FIFO overflow interrupt is disabled.

1: The preview FIFO overflow interrupt is enabled.

- **FO_P_EMP**

0: The preview FIFO empty interrupt is disabled.

1: The preview FIFO empty interrupt is enabled.

- **FO_C_EMP**

0: The codec FIFO empty interrupt is disabled.

1: The codec FIFO empty interrupt is enabled.

- **FR_OVR: Frame Rate Overrun**

0: The frame overrun interrupt is disabled.

1: The frame overrun interrupt is enabled.

47.4.7 ISI Preview Register

Name: ISI_PSIZE

Address: 0xFFFC4020

Access: Read-write

Reset: 0x0

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

23	22	21	20	19	18	17	16
PREV_HSIZE							

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

7	6	5	4	3	2	1	0
PREV_VSIZE							

- **PREV_VSIZE: Vertical size for the preview path**

Vertical Preview size = PREV_VSIZE + 1 (480 max only in RGB mode).

- **PREV_HSIZE: Horizontal size for the preview path**

Horizontal Preview size = PREV_HSIZE + 1 (640 max only in RGB mode).

47.4.8 ISI Preview Decimation Factor Register

Name: ISI_PDECF

Address: 0xFFFC4024

Access: Read-write

Reset: 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
DEC_FACTOR							

- **DEC_FACTOR: Decimation factor**

DEC_FACTOR is 8-bit width, range is from 16 to 255. Values from 0 to 16 do not perform any decimation.

47.4.9 ISI Preview Primary FBD Register

Name: ISI_PPFBD

Address: 0xFFFC4028

Access: Read-write

Reset: 0x0

31	30	29	28	27	26	25	24
PREV_FBD_ADDR							

23	22	21	20	19	18	17	16
PREV_FBD_ADDR							

15	14	13	12	11	10	9	8
PREV_FBD_ADDR							

7	6	5	4	3	2	1	0
PREV_FBD_ADDR							

- **PREV_FBD_ADDR: Base address for preview frame buffer descriptor**

Written with the address of the start of the preview frame buffer queue, reads as a pointer to the current buffer being used. The frame buffer is forced to word alignment.

47.4.10 ISI Codec DMA Base Address Register

Name: ISI_CDBA

Address: 0xFFFC402C

Access: Read-write

Reset: 0x0

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

- **CODEC_DMA_ADDR: Base address for codec DMA**

This register contains codec datapath start address of buffer location.

47.4.11 ISI Color Space Conversion YCrCb to RGB Set 0 Register

Name: ISI_Y2R_SET0

Address: 0xFFFC4030

Access: Read-write

Reset: 0x6832cc95

31	30	29	28	27	26	25	24
C3							

23	22	21	20	19	18	17	16
C2							

15	14	13	12	11	10	9	8
C1							

7	6	5	4	3	2	1	0
C0							

- **C0: Color Space Conversion Matrix Coefficient C0**

C0 element, default step is 1/128, ranges from 0 to 1.9921875

- **C1: Color Space Conversion Matrix Coefficient C1**

C1 element, default step is 1/128, ranges from 0 to 1.9921875

- **C2: Color Space Conversion Matrix Coefficient C2**

C2 element, default step is 1/128, ranges from 0 to 1.9921875

- **C3: Color Space Conversion Matrix Coefficient C3**

C3 element default step is 1/128, ranges from 0 to 1.9921875

47.4.12 ISI Color Space Conversion YCrCb to RGB Set 1 Register

Name: ISI_Y2R_SET1

Address: 0xFFFC4034

Access: Read-write

Reset: 0x00007102

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

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

15	14	13	12	11	10	9	8
–	Cboff	Croff	Yoff	–	–	–	C4

7	6	5	4	3	2	1	0
C4							

- **C4: Color Space Conversion Matrix coefficient C4**

C4 element default step is 1/128, ranges from 0 to 3.9921875

- **Yoff: Color Space Conversion Luminance default offset**

0: No offset

1: Offset = 128

- **Croff: Color Space Conversion Red Chrominance default offset**

0: No offset

1: Offset = 16

- **Cboff: Color Space Conversion Blue Chrominance default offset**

0: No offset

1: Offset = 16

47.4.13 ISI Color Space Conversion RGB to YCrCb Set 0 Register

Name: ISI_R2Y_SET0

Address: 0xFFFC4038

Access: Read-write

Reset: 0x01324145

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

23	22	21	20	19	18	17	16
C2							

15	14	13	12	11	10	9	8
C1							

7	6	5	4	3	2	1	0
C0							

- **C0: Color Space Conversion Matrix coefficient C0**

C0 element default step is 1/256, from 0 to 0.49609375

- **C1: Color Space Conversion Matrix coefficient C1**

C1 element default step is 1/128, from 0 to 0.9921875

- **C2: Color Space Conversion Matrix coefficient C2**

C2 element default step is 1/512, from 0 to 0.2480468875

- **Roff: Color Space Conversion Red component offset**

0: No offset

1: Offset = 16

47.4.14 ISI Color Space Conversion RGB to YCrCb Set 1 Register

Name: ISI_R2Y_SET1

Address: 0xFFFC403C

Access: Read-write

Reset: 0x01245e38

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

23	22	21	20	19	18	17	16
C5							

15	14	13	12	11	10	9	8
C4							

7	6	5	4	3	2	1	0
C3							

- **C3: Color Space Conversion Matrix coefficient C3**

C0 element default step is 1/128, ranges from 0 to 0.9921875

- **C4: Color Space Conversion Matrix coefficient C4**

C1 element default step is 1/256, ranges from 0 to 0.49609375

- **C5: Color Space Conversion Matrix coefficient C5**

C1 element default step is 1/512, ranges from 0 to 0.2480468875

- **Goff: Color Space Conversion Green component offset**

0: No offset

1: Offset = 128

47.4.15 ISI Color Space Conversion RGB to YCrCb Set 2 Register

Name: ISI_R2Y_SET2

Address: 0xFFFC4040

Access: Read-write

Reset: 0x01384a4b

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	Boff
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
C8							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
C7							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
C6							

- **C6: Color Space Conversion Matrix coefficient C6**

C6 element default step is 1/512, ranges from 0 to 0.2480468875

- **C7: Color Space Conversion Matrix coefficient C7**

C7 element default step is 1/256, ranges from 0 to 0.49609375

- **C8: Color Space Conversion Matrix coefficient C8**

C8 element default step is 1/128, ranges from 0 to 0.9921875

- **Boff: Color Space Conversion Blue component offset**

0: No offset

1: Offset = 128

48. Analog-to-Digital Converter (ADC)

48.1 Description

The ADC is based on a Successive Approximation Register (SAR) 10-bit Analog-to-Digital Converter (ADC). It also integrates an 8-to-1 analog multiplexer, making possible the analog-to-digital conversions of 8 analog lines. The conversions extend from 0V to ADVREF.

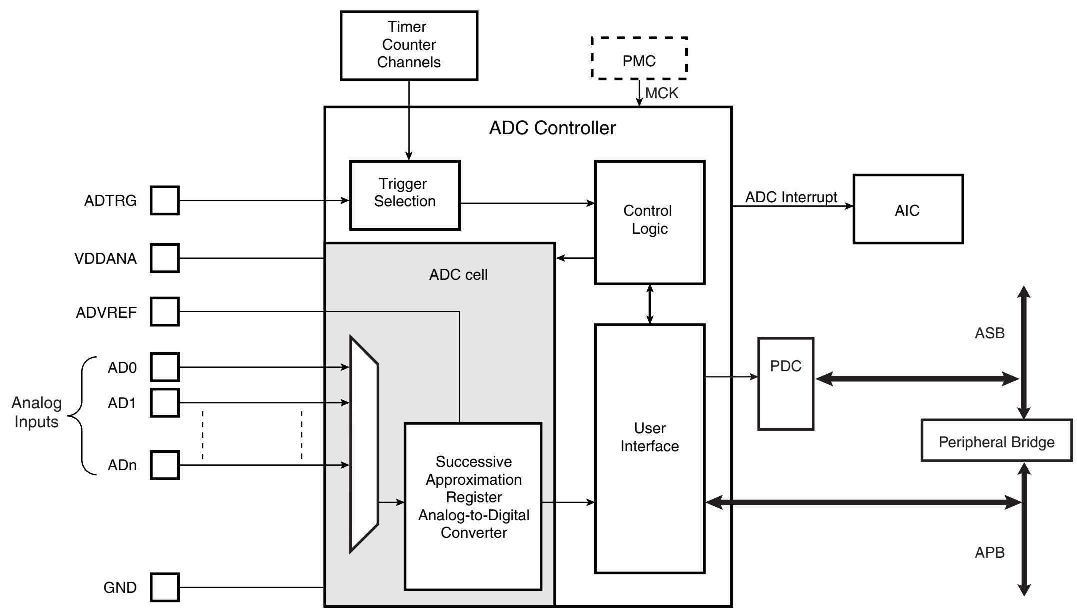
The ADC supports an 8-bit or 10-bit resolution mode, and conversion results are reported in a common register for all channels, as well as in a channel-dedicated register. Software trigger, external trigger on rising edge of the ADTRG pin or internal triggers from Timer Counter output(s) are configurable.

The ADC also integrates a Sleep Mode and a conversion sequencer and connects with a PDC channel. These features reduce both power consumption and processor intervention.

Finally, the user can configure ADC timings, such as Startup Time and Sample & Hold Time.

48.2 Block Diagram

Figure 48-1. Analog-to-Digital Converter Block Diagram



48.3 Signal Description

Table 48-1. ADC Pin Description

Pin Name	Description
VDDANA	Analog power supply
AD0 - AD7	Analog input channels
ADTRG	External trigger

48.4 Product Dependencies

48.4.1 Power Management

The MCK of the ADC Controller is not continuously clocked. The programmer must first enable the ADC Controller MCK in the Power Management Controller (PMC) before using the ADC Controller. However, if the application does not require ADC operations, the ADC Controller clock can be stopped when not needed and restarted when necessary. Configuring the ADC Controller does not require the ADC Controller clock to be enabled.

48.4.2 Interrupt Sources

The ADC interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the ADC interrupt requires the AIC to be programmed first.

Table 48-2. Peripheral IDs

Instance	ID
ADC	24

48.4.3 Analog Inputs

The analog input pins can be multiplexed with PIO lines. In this case, the assignment of the ADC input is automatically done as soon as the corresponding channel is enabled by writing the register ADC_CHER. By default, after reset, the PIO line is configured as input with its pull-up enabled and the ADC input is connected to the GND.

48.4.4 I/O Lines

The pin ADTRG may be shared with other peripheral functions through the PIO Controller. In this case, the PIO Controller should be set accordingly to assign the pin ADTRG to the ADC function.

Table 48-3. I/O Lines

Instance	Signal	I/O Line	Peripheral
ADC	ADTRIG	PB31	A

48.4.5 Timer Triggers

Timer Counters may or may not be used as hardware triggers depending on user requirements. Thus, some or all of the timer counters may be non-connected.

48.4.6 Conversion Performances

For performance and electrical characteristics of the ADC, see the DC Characteristics section.

48.5 Functional Description

48.5.1 Analog-to-digital Conversion

The ADC uses the ADC Clock to perform conversions. Converting a single analog value to a 10-bit digital data requires Sample and Hold Clock cycles as defined in the field SHTIM of the "ADC Mode Register" on page 1070 and 10 ADC Clock cycles. The ADC Clock frequency is selected in the PRESCAL field of the Mode Register (ADC_MR).

The ADC clock range is between MCK/2, if PRESCAL is 0, and MCK/128, if PRESCAL is set to 63 (0x3F). PRESCAL must be programmed in order to provide an ADC clock frequency according to the parameters given in the Product definition section.

48.5.2 Conversion Reference

The conversion is performed on a full range between 0V and the reference voltage pin ADVREF Analog inputs between these voltages convert to values based on a linear conversion.

48.5.3 Conversion Resolution

The ADC supports 8-bit or 10-bit resolutions. The 8-bit selection is performed by setting the bit LOWRES in the ADC Mode Register (ADC_MR). By default, after a reset, the resolution is the highest and the DATA field in the data registers is fully used. By setting the bit LOWRES, the ADC switches in the lowest resolution and the conversion results can be read in the eight lowest significant bits of the data registers. The two highest bits of the DATA field in the corresponding ADC_CDR register and of the LDATA field in the ADC_LCDR register read 0.

Moreover, when a PDC channel is connected to the ADC, 10-bit resolution sets the transfer request sizes to 16-bit. Setting the bit LOWRES automatically switches to 8-bit data transfers. In this case, the destination buffers are optimized.

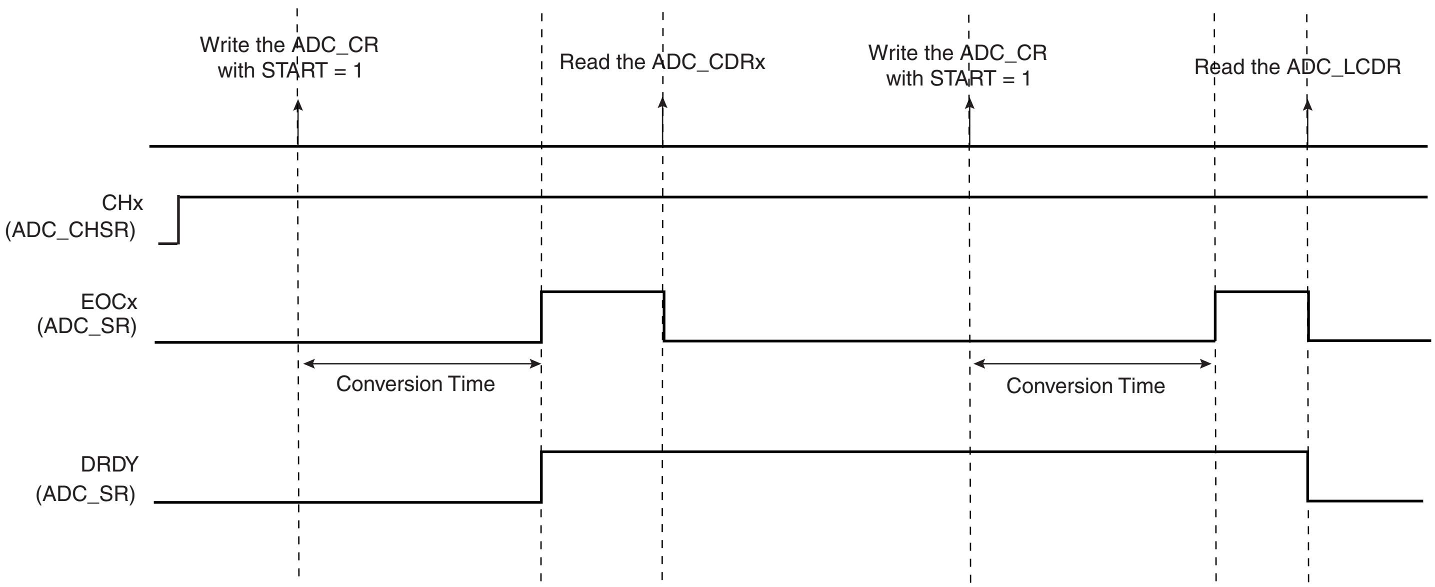
48.5.4 Conversion Results

When a conversion is completed, the resulting 10-bit digital value is stored in the Channel Data Register (ADC_CDR) of the current channel and in the ADC Last Converted Data Register (ADC_LCDR).

The channel EOC bit in the Status Register (ADC_SR) is set and the DRDY is set. In the case of a connected PDC channel, DRDY rising triggers a data transfer request. In any case, either EOC and DRDY can trigger an interrupt.

Reading one of the ADC_CDR registers clears the corresponding EOC bit. Reading ADC_LCDR clears the DRDY bit and the EOC bit corresponding to the last converted channel.

Figure 48-2. EOCx and DRDY Flag Behavior

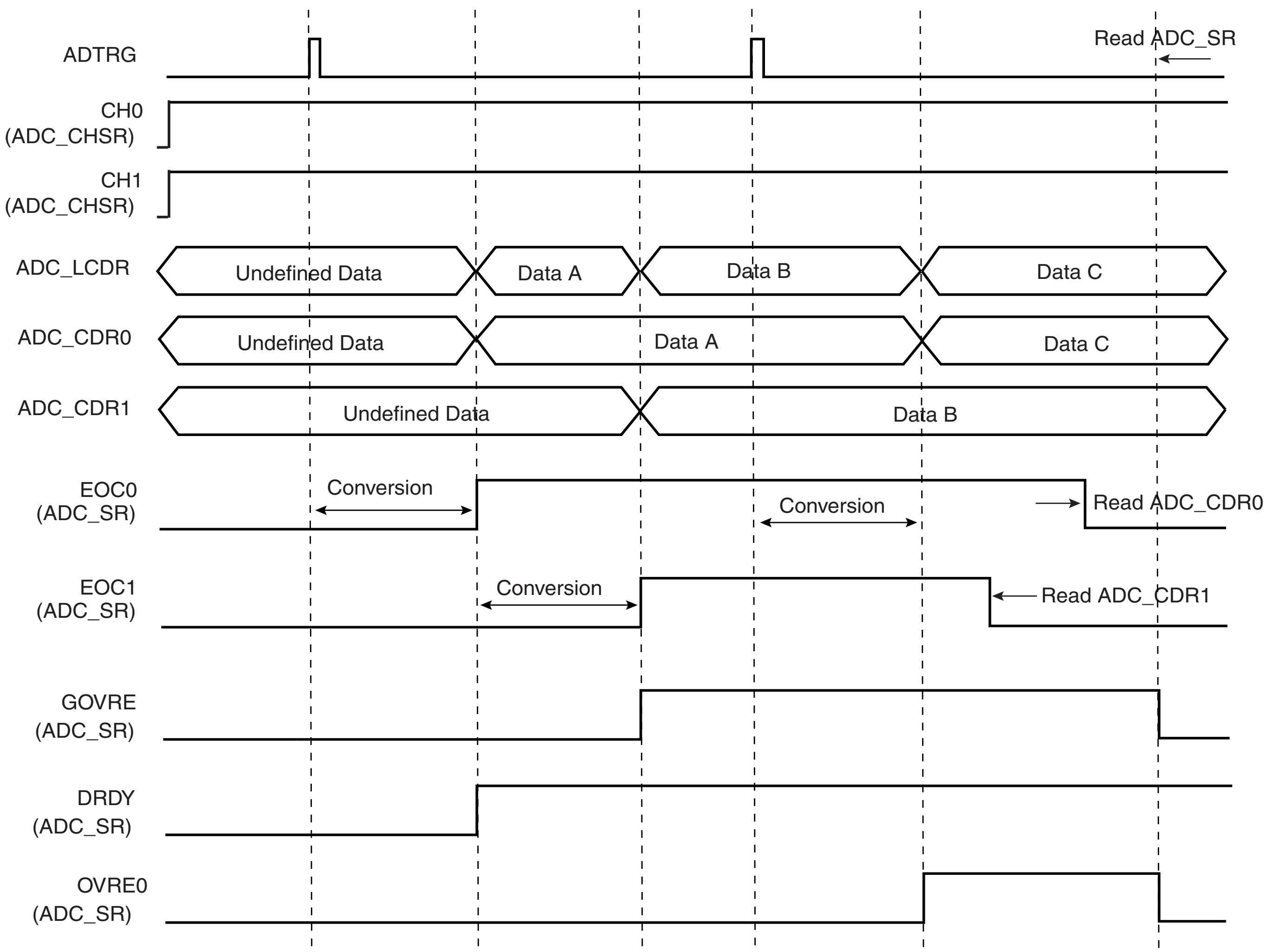


If the ADC_CDR is not read before further incoming data is converted, the corresponding Overrun Error (OVRE) flag is set in the Status Register (ADC_SR).

In the same way, new data converted when DRDY is high sets the bit GOVRE (General Overrun Error) in ADC_SR.

The OVRE and GOVRE flags are automatically cleared when ADC_SR is read.

Figure 48-3. GOVRE and OVREx Flag Behavior



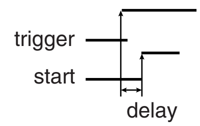
Warning: If the corresponding channel is disabled during a conversion or if it is disabled and then reenabled during a conversion, its associated data and its corresponding EOC and OVRE flags in ADC_SR are unpredictable.

48.5.5 Conversion Triggers

Conversions of the active analog channels are started with a software or a hardware trigger. The software trigger is provided by writing the Control Register (ADC_CR) with the bit START at 1.

The hardware trigger can be one of the TIOA outputs of the Timer Counter channels, or the external trigger input of the ADC (ADTRG). The hardware trigger is selected with the field TRGSEL in the Mode Register (ADC_MR). The selected hardware trigger is enabled with the bit TRGEN in the Mode Register (ADC_MR).

If a hardware trigger is selected, the start of a conversion is triggered after a delay starting at each rising edge of the selected signal.Due to asynchronism handling, the delay may vary in a range of 2 MCK clock periods to 1 ADC clock period.



If one of the TIOA outputs is selected, the corresponding Timer Counter channel must be programmed in Waveform Mode.

Only one start command is necessary to initiate a conversion sequence on all the channels. The ADC hardware logic automatically performs the conversions on the active channels, then waits for a new request. The Channel Enable (ADC_CHER) and Channel Disable (ADC_CHDR) Registers enable the analog channels to be enabled or disabled independently.

If the ADC is used with a PDC, only the transfers of converted data from enabled channels are performed and the resulting data buffers should be interpreted accordingly.

Warning: Enabling hardware triggers does not disable the software trigger functionality. Thus, if a hardware trigger is selected, the start of a conversion can be initiated either by the hardware or the software trigger.

48.5.6 Sleep Mode and Conversion Sequencer

The ADC Sleep Mode maximizes power saving by automatically deactivating the ADC when it is not being used for conversions. Sleep Mode is selected by setting the bit SLEEP in the Mode Register ADC_MR.

The SLEEP mode is automatically managed by a conversion sequencer, which can automatically process the conversions of all channels at lowest power consumption.

When a start conversion request occurs, the ADC is automatically activated. As the analog cell requires a start-up time, the logic waits during this time and starts the conversion on the enabled channels. When all conversions are complete, the ADC is deactivated until the next trigger. Triggers occurring during the sequence are not taken into account.

The conversion sequencer allows automatic processing with minimum processor intervention and optimized power consumption. Conversion sequences can be performed periodically using a Timer/Counter output. The periodic acquisition of several samples can be processed automatically without any intervention of the processor thanks to the PDC.

Note: The reference voltage pins always remain connected in normal mode as in sleep mode.

48.5.7 ADC Timings

Each ADC has its own minimal Startup Time that is programmed through the field STARTUP in the Mode Register ADC_MR.

In the same way, a minimal Sample and Hold Time is necessary for the ADC to guarantee the best converted final value between two channels selection. This time has to be programmed through the bitfield SHTIM in the Mode Register ADC_MR.

Warning: No input buffer amplifier to isolate the source is included in the ADC. This must be taken into consideration to program a precise value in the SHTIM field. See the section, ADC Characteristics in the product datasheet.

48.6 Analog-to-Digital Converter (ADC) User Interface

Table 48-4. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	ADC_CR	Write-only	–
0x04	Mode Register	ADC_MR	Read-write	0x00000000
0x08	Reserved	–	–	–
0x0C	Reserved	–	–	–
0x10	Channel Enable Register	ADC_CHER	Write-only	–
0x14	Channel Disable Register	ADC_CHDR	Write-only	–
0x18	Channel Status Register	ADC_CHSR	Read-only	0x00000000
0x1C	Status Register	ADC_SR	Read-only	0x000C0000
0x20	Last Converted Data Register	ADC_LCDR	Read-only	0x00000000
0x24	Interrupt Enable Register	ADC_IER	Write-only	–
0x28	Interrupt Disable Register	ADC_IDR	Write-only	–
0x2C	Interrupt Mask Register	ADC_IMR	Read-only	0x00000000
0x30	Channel Data Register 0	ADC_CDR0	Read-only	0x00000000
0x34	Channel Data Register 1	ADC_CDR1	Read-only	0x00000000
...	...	...	...	...
0x4C	Channel Data Register 7	ADC_CDR7	Read-only	0x00000000
0x50 - 0xFC	Reserved	–	–	–

48.6.1 ADC Control Register

Name: ADC_CR

Address: 0xFFFC0000

Access: Write-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	START	SWRST

- **SWRST: Software Reset**

0 = No effect.

1 = Resets the ADC simulating a hardware reset.

- **START: Start Conversion**

0 = No effect.

1 = Begins analog-to-digital conversion.

48.6.2 ADC Mode Register

Name: ADC_MR

Address: 0xFFFC0004

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	SHTIM			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	STARTUP						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PRESCAL							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	SLEEP	LOWRES	TRGSEL			TRGEN

- **TRGEN: Trigger Enable**

TRGEN	Selected TRGEN
0	Hardware triggers are disabled. Starting a conversion is only possible by software.
1	Hardware trigger selected by TRGSEL field is enabled.

- **TRGSEL: Trigger Selection**

TRGSEL			Selected TRGSEL
0	0	0	TIOA Output of the Timer Counter 0
0	0	1	TIOA Output of the Timer Counter 1
0	1	0	TIOA Output of the Timer Counter 2
0	1	1	TIOB Output of the Timer Counter 0
1	0	0	TIOB Output of the Timer Counter 1
1	0	1	TIOB Output of the Timer Counter 2
1	1	0	External trigger
1	1	1	Reserved

- **LOWRES: Resolution**

LOWRES	Selected Resolution
0	10-bit resolution
1	8-bit resolution

- **SLEEP: Sleep Mode**

SLEEP	Selected Mode
0	Normal Mode
1	Sleep Mode

- **PRESCAL: Prescaler Rate Selection**
  ADCClock = MCK / ((PRESCAL+1) * 2)

- **STARTUP: Start Up Time**
  Startup Time = (STARTUP+1) * 8 / ADCClock

- **SHTIM: Sample & Hold Time**
  Sample & Hold Time = SHTIM/ADCClock

48.6.3 ADC Channel Enable Register

Name: ADC_CHER

Address: 0xFFFC0010

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Enable**

0 = No effect.

1 = Enables the corresponding channel.

48.6.4 ADC Channel Disable Register

Name: ADC_CHDR

Address: 0xFFFC0014

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Disable**

0 = No effect.

1 = Disables the corresponding channel.

Warning: If the corresponding channel is disabled during a conversion or if it is disabled then reenabled during a conversion, its associated data and its corresponding EOC and OVRE flags in ADC_SR are unpredictable.

48.6.5 ADC Channel Status Register

Name: ADC_CHSR

Address: 0xFFFC0018

Access: Read-only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: Channel x Status**

0 = Corresponding channel is disabled.

1 = Corresponding channel is enabled.

48.6.6 ADC Status Register

Name: ADC_SR

Address: 0xFFFC001C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion x**

0 = Corresponding analog channel is disabled, or the conversion is not finished.

1 = Corresponding analog channel is enabled and conversion is complete.

- **OVREx: Overrun Error x**

0 = No overrun error on the corresponding channel since the last read of ADC_SR.

1 = There has been an overrun error on the corresponding channel since the last read of ADC_SR.

- **DRDY: Data Ready**

0 = No data has been converted since the last read of ADC_LCDR.

1 = At least one data has been converted and is available in ADC_LCDR.

- **GOVRE: General Overrun Error**

0 = No General Overrun Error occurred since the last read of ADC_SR.

1 = At least one General Overrun Error has occurred since the last read of ADC_SR.

- **ENDRX: End of RX Buffer**

0 = The Receive Counter Register has not reached 0 since the last write in ADC_RCR or ADC_RNCR.

1 = The Receive Counter Register has reached 0 since the last write in ADC_RCR or ADC_RNCR.

- **RXBUFF: RX Buffer Full**

0 = ADC_RCR or ADC_RNCR have a value other than 0.

1 = Both ADC_RCR and ADC_RNCR have a value of 0.

48.6.7 ADC Last Converted Data Register

Name: ADC_LCDR

Address: 0xFFFC0020

Access: Read-only

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

- **LDATA: Last Data Converted**

The analog-to-digital conversion data is placed into this register at the end of a conversion and remains until a new conversion is completed.

48.6.8 ADC Interrupt Enable Register

Name: ADC_IER

Address: 0xFFFC0024

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion Interrupt Enable x**

- **OVREx: Overrun Error Interrupt Enable x**

- **DRDY: Data Ready Interrupt Enable**

- **GOVRE: General Overrun Error Interrupt Enable**

- **ENDRX: End of Receive Buffer Interrupt Enable**

- **RXBUFF: Receive Buffer Full Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

48.6.9 ADC Interrupt Disable Register

Name: ADC_IDR

Address: 0xFFFC0028

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion Interrupt Disable x**

- **OVREx: Overrun Error Interrupt Disable x**

- **DRDY: Data Ready Interrupt Disable**

- **GOVRE: General Overrun Error Interrupt Disable**

- **ENDRX: End of Receive Buffer Interrupt Disable**

- **RXBUFF: Receive Buffer Full Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

48.6.10 ADC Interrupt Mask Register

Name: ADC_IMR

Address: 0xFFFC002C

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: End of Conversion Interrupt Mask x**

- **OVREx: Overrun Error Interrupt Mask x**

- **DRDY: Data Ready Interrupt Mask**

- **GOVRE: General Overrun Error Interrupt Mask**

- **ENDRX: End of Receive Buffer Interrupt Mask**

- **RXBUFF: Receive Buffer Full Interrupt Mask**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

48.6.11 ADC Channel Data Register

Name: ADC_CDRx

Address: 0xFFFC0030

Access: Read-only

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

- **DATA: Converted Data**

The analog-to-digital conversion data is placed into this register at the end of a conversion and remains until a new conversion is completed. The Convert Data Register (CDR) is only loaded if the corresponding analog channel is enabled.

49. AT91CAP9 Electrical Characteristics

49.1 Absolute Maximum Ratings

Table 49-1. Absolute Maximum Ratings

Operating Temperature (Industrial)	-40· C to +85°C
Storage Temperature	-60°C to +150°C
Voltage on Input Pins with Respect to Ground	-0.3V to +4.0V
Maximum Operating Voltage (VDDCORE and VDDBU)	1.5V
Maximum Operating Voltage ((VDDIOM, VDDIOP0, VDDIOP1, VDDIOMPA, VDDIOMPB, VDDPLL, VDDUTMII and VDDANA)	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.

49.2 DC Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C, unless otherwise specified.

Table 49-2. DC Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{VDDCORE}$	DC Supply Core		1.08		1.32	V
V_{VDDBU}	DC Supply Backup		1.08		1.32	V
V_{VDDPLL}	DC Supply PLL		3.0		3.6	V
$V_{VDDUPLL}$	DC Supply USB PLL		1.08		1.32	V
V_{VDDIOM}	DC Supply Memory I/Os	Selectable by Software in Bus Matrix	1.65/3.0		1.95/3.6	V
$V_{VDDIOP0}$	DC Supply Peripheral 0 I/Os		3.0		3.6	V
$V_{VDDIOP1}$	DC Supply Peripheral 1 I/Os		1.65		3.6	V
$V_{VDDIOMPA}$	DC Supply MP Block I/O A lines		1.65		3.6	
$V_{VDDIOMPB}$	DC Supply MP Block I/O B lines		1.65		3.6	
$V_{VDDUTMII}$	DC Supply USB UTMI Interface		3.0		3.6	V
$V_{VDDUTMIC}$	DC Supply USB UTMI Core		1.08		1.32	V
V_{VDDANA}	DC Supply ADC		3.0		3.6	V
V_{IL}	Input Low-level Voltage		-0.3		0.8	V
V_{IH}	Input High-level Voltage	V_{VDDIO}= V_{VDDIOM} or V_{VDDIOP}	2		V_{VDDIO}+0.3	V
V_{OL}	Output Low-level Voltage				0.4	V
V_{OH}	Output High-level Voltage	V_{VDDIO}= V_{VDDIOM} or V_{VDDIOP}	V_{VDDIO}-0.4			V
R_{PULLUP}	Pull-up Resistance	PA0-PA31, PB0-PB31, PC0-PC31, PD0-PD31	70	100	175	kOhm
I_O	Output Current	PA0-PA31, PB0-PB31, PC0-PC31, PD0-PD31			8	mA
I_{SC}	Static Current	On $V_{VDDCORE}$ = 1.2V, MCK = 0 Hz, excluding POR; T_A =25°C	TBD		TBD	µA
		All inputs driven TMS, TDI, TCK, NRST = 1; T_A =85°C	TBD		TBD	µA
		On V_{VDDBU} = 1.2V, Logic cells consumption, excluding POR; T_A =25°C	TBD		TBD	nA
		All inputs driven WKUP = 0; T_A =85°C	TBD		TBD	nA

49.3 Power Consumption

This section contains:

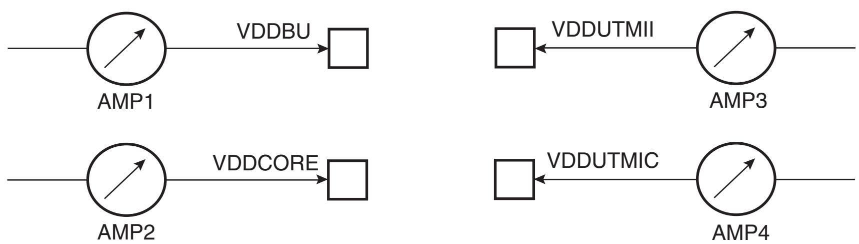
- The typical power consumption of PLLs, Slow Clock and Main Oscillator.
- The power consumption of power supply in three different modes: Active, Ultra Low-power and Backup.
- The power consumption by peripheral: calculated as the difference in current measurement after having enabled then disabled the corresponding clock.

49.3.1 Power Consumption versus Modes

The values in Table 49-3 and Table 49-4 on page 1084 are estimated values of the power consumption with operating conditions as follows:

- $V_{VDDIOM} = V_{DDIOP0} = V_{DDIOP1} = V_{DDIOMPA} = V_{DDIOMPB} = V_{DDANA} = 3.3$ V
- $V_{DDPLL} = V_{DDUPLL} = V_{DDUTMII} = 3.3$V
- $V_{DDCORE} = V_{DDBU} = V_{DDUTMIC} = 1.2$V
- $T_A = 25°$ C
- There is no consumption on the I/Os of the device

Figure 49-1. Measures Schematics



These figures represent the power consumption estimated on the power supplies.

Table 49-3. Power Consumption for Different Modes

Mode	Conditions	Consumption	Unit
Active	ARM Core clock is 200 MHz. MCK is 100 MHz. All peripheral clocks disabled Icache enabled onto AMP2	53	mA
Idle	ARM Core in idle state, waiting an interrupt MCK is 100 MHz All peripheral clocks disabled onto AMP2	23	mA
Ultra low power	ARM Core clock is 500 Hz MCK is 500 Hz All peripheral clocks disabled onto AMP2	190	µA
Backup	Device only V_{DDBU} powered onto AMP1	4	µA

Table 49-4. Power Consumption by Peripheral in Active Mode (onto AMP2)

Peripheral	Consumption	Unit
PIO Controller	19	µA/MHz
USART	12	
UHP	9	
UDPHS	48	
ADC	7	
TWI	6	
SPI	7	
MCI	10	
SSC	12	
Timer Counter Channels	6	
CAN	50	
PWMC	7	
EMAC	68	
LCDC	38	
ISI	8	
AC97	10	
HDMA	60	

49.4 32 kHz Crystal Oscillator Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C and worst case of power supply, unless otherwise specified.

Table 49-5. 32 kHz Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CP32KHz})$	Crystal Oscillator Frequency			32.768		kHz
$C_{CRYSTAL32}$	Crystal Load Capacitance	Crystal @ 32.768 kHz	6		12.5	pF
C_{LEXT32} [(2)]	External Load Capacitance	$C_{CRYSTAL32}$ = 6 pF[(3)]		7		pF
		$C_{CRYSTAL32}$ = 12.5 pF[(3)]		20		pF
$DUTY_{32}$	Duty Cycle		40		60	%
t_{ST}	Startup Time	R_S < 50 kΩ, C_L = 6pF[(1)]			400	ms
		R_S < 50 kΩ, C_L = 12.5 pF[(1)]			900	ms
		R_S < 100 kΩ, C_L = 6pF[(1)]			600	ms
		R_S < 100 kΩ, C_L = 12.5 pF[(1)]			1200	ms

Notes: 1. R_S is the equivalent series resistance, C_L is the equivalent load capacitance.
2. C_{LEXT32} is determined by taking into account internal parasitic and package load capacitance.
3. Additional board load capacitance should be subtracted from C_{LEXT32}.

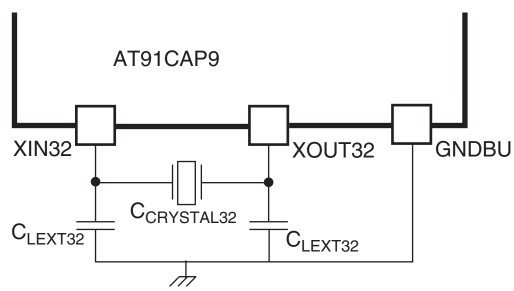


Table 49-6. 32KHz Crystal Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
ESR	Equivalent Series Resistor Rs			50	100	Ω
C_M	Motional Capacitance		0.6		3	fF
C_S	Shunt Capacitance		0.6		2	pF

Table 49-7. RC Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CPRC})$	RC Oscillator Frequency		20	32	44	KHz
$DUTY_{RC}$	Duty Cycle		45	50	55	%

Table 49-7. RC Oscillator Characteristics (Continued)

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{ST}	Startup Time				75	µs
I_{DDON}	Current Consumption	After Startup Time		0.65	1.0	µA
I_{DDSTB}	Standby Current Consumption				0.2	µA

49.5 12 MHz Main Oscillator Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C and worst case of power supply, unless otherwise specified.

Table 49-8. 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 pF(1)		25		pF
		$C_{CRYSTAL}$ = 20 pF(1)		35		
$DUTY_{MAIN}$	Duty Cycle		40	50	60	%
t_{ST}	Startup Time				2	ms
I_{DDST}	Standby Current Consumption	Standby mode			1	µA
P_{ON}	Drive Level				150	µW
$I_{DD\ ON}$	Current Dissipation	@ 16MHz		550	950	µA
I_{BYPASS}	Bypass Current Dissipation			3.3	5.0	µW/MHz

Note: 1. Additional board load capacitance should be subtracted from C_{LEXT}.

Figure 49-2. 12 MHz Crystal Connection

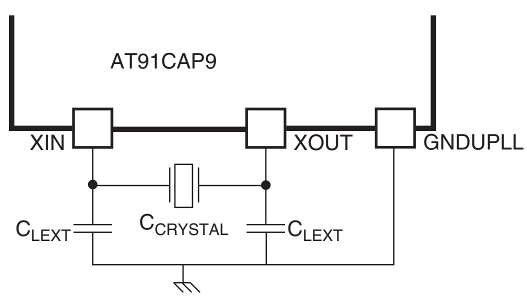


Table 49-9 gives the characteristics that the crystal must satisfy for correct operation with the oscillator.

Table 49-9. Main Crystal Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
ESR	Equivalent Series Resistor Rs				80	Ω
C_M	Motional Capacitance		5		9	fF
C_S	Shunt Capacitance				7	pF

Table 49-10 gives the Electrical Characteristics of the XIN pin when the oscillator is in Bypass Mode.

Table 49-10. XIN Clock Electrical Characteristics in Bypass Mode

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPXIN})$	XIN Clock Frequency			50	MHz
t_{CPXIN}	XIN Clock Period		20		ns
t_{CHXIN}	XIN Clock High Half-period		0.4 x t_{CPXIN}	0.6 x t_{CPXIN}	
t_{CLXIN}	XIN Clock Low Half-period		0.4 x t_{CPXIN}	0.6 x t_{CPXIN}	
C_{IN}	XIN Input Capacitance	(1)		5	pF
R_{IN}	XIN Pulldown Resistor	(1)		500	kΩ

Note: These characteristics apply only when Main Oscillator is in Bypass Mode (i.e., when MOSCEN = 0 and OSCBYPASS = 1) in the CKGR_MOR register. See PMC Clock Generator Main Oscillator Register in the Power Management Section.

49.6 PLLA and PLLB Characteristics

The PLLA and PLLB are identical and their characteristics are described in Table 49-11.

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C and worst case of power supply, unless otherwise specified.

Table 49-11. Phase Lock Loop A and B Characteristics

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	µA

Note: 1. Startup time depends on PLL RC filter. A calculation tool is provided by Atmel.

49.7 UTMI PLL Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C and worst case of power supply, unless otherwise specified.

Table 49-12. 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	µA

49.8 USB HS Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C and worst case of power supply, unless otherwise specified.

49.8.1 Electrical Characteristics

Table 49-13. 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	µs
T_{OSC}	Oscillator settling time	With Crystal 12MHz			2	ms
$T_{SETTLING}$	Settling time	F_{IN} = 12 MHz		0.3	0.5	ms

49.8.2 Static Power Consumption

Table 49-14. Static Power Consumption

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{BIAS}	Bias current consumption on VBG				1	µA
$I_{VDDUTMII}$	HS Transceiver and I/O current consumption				8	µA
	LS / FS Transceiver and I/O current consumption	No connection[(1)]			3	µA
$I_{VDDUTMIC}$	Core current consumption				2	µA

Note: 1. If cable is connected, add 200 µA (typical) due to Pull-up/Pull-down current consumption.

49.8.3 Dynamic Power Consumption

Table 49-15. 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(1)		4	6	mA
	LS / FS Transceiver current consumption	FS transmission 5m cable(1)		26	30	mA
	LS / FS Transceiver current consumption	FS reception(1)		3	4.5	mA
$I_{VDDUTMIC}$	Core current consumption			5.5	9	mA

Note: 1. Including 1mA due to Pull-up/Pull-down current consumption.

49.9 ADC

Table 49-16. Channel Conversion Time and ADC CLock

Parameter	Conditions	Min	Typ	Max	Units
ADC Clock Frequency	10-bit resolution mode			5	MHz
ADC Clock Frequency	8-bit resolution mode			TBD	MHz
Startup Time	Return from Idle Mode			40	µs
Track and Hold Acquisition Time		500			ns
Conversion Time	ADC Clock = 5 MHz			2	µs
Throughput Rate	ADC Clock = 5 MHz			384(1)	kSPS

Notes: 1. Corresponds to 13 clock cycles at 5 MHz: 3 clock cycles for track and hold acquisition time and 10 clock cycles for conversion.

Table 49-17. External Voltage Reference Input

Parameter	Conditions	Min	Typ	Max	Units
ADVREF Input Voltage Range		2.6		VDDANA	V
ADVREF Average Current	On 13 samples with ADC Clock = 5 MHz			250	µA
Current Consumption on VDDANA				300	µA

Table 49-18. Analog Inputs

Parameter	Min	Typ	Max	Units
Input Voltage Range	0		ADVREF	V
Input Leakage Current		1		µA
Input Capacitance			5	pF

The user can drive ADC input with impedance up to:

- $Z_{OUT} \leq$ (SHTIM -500) x 12.5

with SHTIM (Sample and Hold Time register) expressed in ns and Z_{OUT} expressed in ohms.

Table 49-19. Transfer Characteristics

Parameter	Min	Typ	Max	Units
Resolution		10		Bit
Integral Non-linearity			±2	LSB
Differential Non-linearity			±0.9	LSB
Offset Error	-10		10	mV
Gain Error			±2	LSB

49.10 Core Power Supply POR Characteristics

49.10.1 Power Sequence Requirements

The CAP9S board design must comply with the power-up and power-down sequence guidelines below to guarantee reliable operation of the device. Any deviation from these sequences may lead to the following situations:

- Cause excessive current during power-up phase
- Prevent the device from booting

49.10.2 Power-Up Sequence

Table 49-20. 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	µs

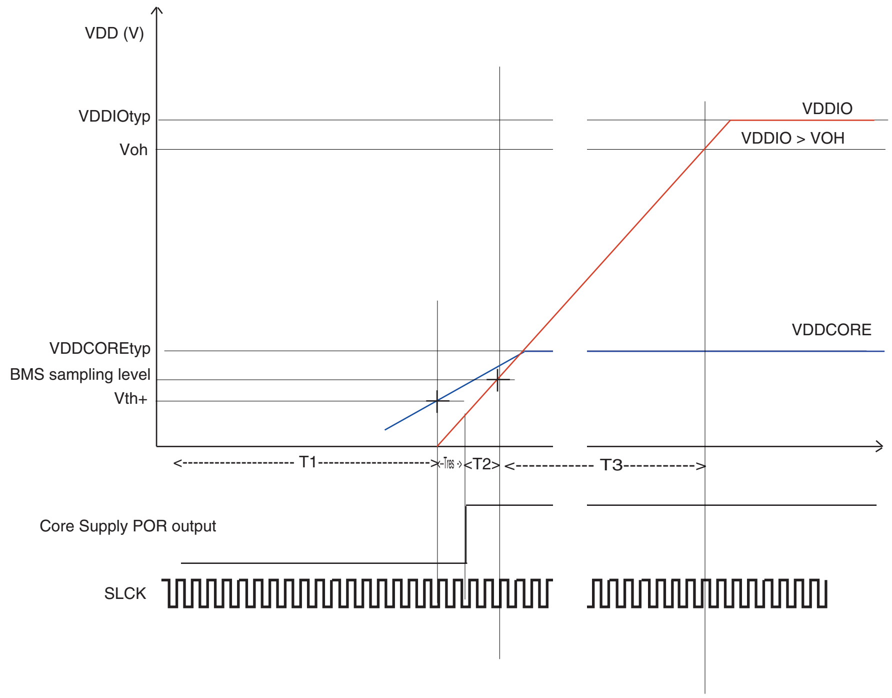
VDDCORE and VDDBU are controlled by internal POR (Power On Reset) to guarantee that these power sources reach their target values prior to the release of POR.

- VDDIOM, VDDIOP0, VDDIOP1, VDDMPIOA and VDDMPIOB must NOT be powered until VDDCORE has reached a level superior to V_{TH+}.
- VDDIOM, VDDIOP0, VDDIOP1, VDDMPIOA and VDDMPIOB must be $\leq$ to V_{IH} (refer to Table 49-2 for more details) within (Tres + T2) after VDDCORE has reached V_{TH+}.
- VDDIOM, VDDIOP0, VDDIOP1, VDDMPIOA and VDDMPIOB must reach V_{OH} (refer to Table 49-2 for more details) within (Tres + T2 + T3) after VDDCORE has reached V_{TH+}.
  – T_{RES} is a POR characteristic
  – T2 = 3 x T_{SLCK}
  – T3 = 16 x T_{SLCK}

The T_{SLCK} min (22 µs) is obtained for the maximum frequency of the internal RC oscillator (44 kHz).

– T_{RES} = 80 µs
– T2 = 66 µs
– T3 = 352 µs

Figure 49-3. $V_{VDDCORE}$ and V_{VDDIO} Constraints at Startup



49.10.3 Power-Down Sequence

Switch-off the VDDIOM, VDDIOP0, VDDIOP1, VDDMPIOA and VDDMPIOB power supply prior to or at the same time as VDDCORE.

No power-up or power-down restrictions apply to VDDBU, VDDPLL, VDDUPLL, VDDUTMII, VDDUTMIC and VDDANA.

49.11 Timings

49.11.1 Corner Definition

Table 49-21. Corner Definition

Corner	Process	Temp (External Junction)	VDDCORE: 1.2V	VDDIO: 1.8V	VDDIO: 3.3V
MAX	Slow	85°C; 100°C	1.10V	1.65V	3.0V
MIN	Fast	-40C; -40C	1.32V	1.95V	3.6V

Timings in MAX corner always result from the extraction and comparison of timings in MAX and MIN corners.

49.11.2 Processor Clock

Table 49-22. Processor Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Units
$1/(t_{CPPCK})$	Processor Clock Frequency	Corner MAX		200	MHz

49.11.3 Maximum Speed of the I/Os

Criteria used to define the maximum frequency of the I/Os:

- output duty cycle (40%-60%)
- minimum output swing: 100mV to VDDIO - 100mV
- Addition of rising and falling time inferior to 75% of the period Addition of rising and falling time inferior to 75% of the period

Table 49-23. I/O Characteristics

Symbol	Parameter	Conditions	Min	Max	Units
FreqMax	VDDIOP0 Pin Group frequency	3.3V domain (1)		50	MHz
FreqMax	VDDIOP1 Pin Group frequency	3.3V domain (1)		50	MHz

Notes: 1. 3.3V domain: V_{VDDIOP} from 3.0V to 3.6V, maximum external capacitor = 30pF

49.11.4 SMC Timings

49.11.4.1 Capacitance

Timings are given assuming a capacitance load on data, control and address pads.

Table 49-24. Capacitance Load on Address Pads

	Corner	
Supply	**MAX**	**MIN**
3.3V	50pF	5 pF
1.8V	30 pF	5 pF

Table 49-25. Capacitance Load on Data Pads

	Corner	
Supply	**MAX**	**MIN**
3.3V	50pF	5 pF
1.8V	20 pF	5 pF

In the following tables, t_{CPMCK} is MCK period.

49.11.4.2 Read Timings

Table 49-26. SMC Read Signals - NRD Controlled (READ_MODE= 1)

Symbol	Parameter	Min		Units
	VDDIOM supply	**1.8V**	**3.3V**	
NO HOLD SETTINGS (nrd hold = 0)				
SMC_1	Data Setup before NRD High	13.8	12.5	ns
SMC_2	Data Hold after NRD High	0	0	ns
HOLD SETTINGS (nrd hold …0)				
SMC_3	Data Setup before NRD High	9.7	8.4	ns
SMC_4	Data Hold after NRD High	0	0	ns
HOLD or NO HOLD SETTINGS (nrd hold …0, nrd hold =0)				
SMC_5	NBS0/A0, NBS1, NBS2/A1, NBS3, A2 - A25 Valid before NRD High	(nrd setup + nrd pulse)* t_{CPMCK} - 0.7	(nrd setup + nrd pulse)* t_{CPMCK} - 0.8	ns
SMC_6	NCS low before NRD High	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} - 0.8	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} - 0.8	ns
SMC_7	NRD Pulse Width	nrd pulse * t_{CPMCK} - 0.5	nrd pulse * t_{CPMCK} - 0.6	ns

Table 49-27. SMC Read Signals - NCS Controlled (READ_MODE= 0)

Symbol	Parameter	Min		Units
	VDDIOM supply	**1.8V**	**3.3V**	
NO HOLD SETTINGS (ncs rd hold = 0)				
SMC_8	Data Setup before NCS High	15.1	13.8	ns
SMC_9	Data Hold after NCS High	0	0	ns
HOLD SETTINGS (ncs rd hold ≠ 0)				
SMC_{10}	Data Setup before NCS High	11.1	9.7	ns
SMC_{11}	Data Hold after NCS High	0	0	ns
HOLD or NO HOLD SETTINGS (ncs rd hold ≠ 0, ncs rd hold = 0)				
Symbol	**Parameter**	**Min**		**Units**
	VDDIOM supply	**1.8V**	**3.3V**	
SMC_{12}	NBS0/A0, NBS1, NBS2/A1, NBS3, A2 - A25 valid before NCS High	(ncs rd setup + ncs rd pulse)* t_{CPMCK} - 0.5	(ncs rd setup + ncs rd pulse)* t_{CPMCK} - 0.5	ns
SMC_{13}	NRD low before NCS High	(ncs rd setup + ncs rd pulse - nrd setup)* t_{CPMCK} - 0.3	(ncs rd setup + ncs rd pulse - nrd setup)* t_{CPMCK} - 0.4	ns
SMC_{14}	NCS Pulse Width	ncs rd pulse length * t_{CPMCK} - 0.5	ncs rd pulse length * t_{CPMCK} - 0.6	ns

49.11.4.3 Write Timings

Table 49-28. SMC Write Signals - NWE controlled (WRITE_MODE = 1)

Symbol	Parameter	Min		Units
		1.8V Supply	**3.3V Supply**	
HOLD or NO HOLD SETTINGS (nwe hold ≠ 0, nwe hold = 0)				
SMC_{15}	Data Out Valid before NWE High	nwe pulse * t_{CPMCK} - 0.8	nwe pulse * t_{CPMCK} - 0.9	ns
SMC_{16}	NWE Pulse Width	nwe pulse * t_{CPMCK} - 0.5	nwe pulse * t_{CPMCK} -0.5	ns
SMC_{17}	NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25 valid before NWE low	nwe setup * t_{CPMCK} - 0.6	nwe setup * t_{CPMCK} - 0.7	ns
SMC_{18}	NCS low before NWE high	(nwe setup - ncs rd setup + nwe pulse) * t_{CPMCK} - 0.7	(nwe setup - ncs rd setup + nwe pulse) * t_{CPMCK} - 0.7	ns
HOLD SETTINGS (nwe hold ≠ 0)				
SMC_{19}	NWE High to Data OUT, NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25 change	nwe hold * t_{CPMCK} - 0.8	nwe hold * t_{CPMCK} - 0.8	ns

Table 49-28. SMC Write Signals - NWE controlled (WRITE_MODE = 1)

Symbol	Parameter	Min		Units
		1.8V Supply	3.3V Supply	
SMC_{20}	NWE High to NCS Inactive [1]	(nwe hold - ncs wr hold)* t_{CPMCK} - 0.3	(nwe hold - ncs wr hold)* t_{CPMCK} - 0.3	ns
NO HOLD SETTINGS (nwe hold = 0)				
SMC_{21}	NWE High to Data OUT, NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25, NCS change[1]	3.0	2.7	ns

Notes: 1. hold length = total cycle duration - setup duration - pulse duration. “hold length” is for “ncs wr hold length” or “NWE hold length”.

Table 49-29. SMC Write NCS Controlled (WRITE_MODE = 0)

Symbol	Parameter	Min		Units
		1.8V Supply	3.3V Supply	
SMC_{22}	Data Out Valid before NCS High	ncs wr pulse * t_{CPMCK} - 0.8	ncs wr pulse * t_{CPMCK} - 0.7	ns
SMC_{23}	NCS Pulse Width	ncs wr pulse * t_{CPMCK} - 0.5	ncs wr pulse * t_{CPMCK} - 0.6	ns
SMC_{24}	NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25 valid before NCS low	ncs wr setup * t_{CPMCK} - 0.5	ncs wr setup * t_{CPMCK} - 0.6	ns
SMC_{25}	NWE low before NCS high	(ncs wr setup - nwe setup + ncs pulse)* t_{CPMCK} - 0.4	(ncs wr setup - nwe setup + ncs pulse)* t_{CPMCK} - 0.5	ns
SMC_{26}	NCS High to Data Out, NBS0/A0, NBS1, NBS2/A1, NBS3, A2 - A25, change	ncs wr hold * t_{CPMCK} - 0.8	ncs wr hold * t_{CPMCK} - 0.8	ns
SMC_{27}	NCS High to NWE Inactive	(ncs wr hold - nwe hold)* t_{CPMCK} - 0.5	(ncs wr hold - nwe hold)* t_{CPMCK} - 0.5	ns

Figure 49-4. SMC Timings - NCS Controlled Read and Write

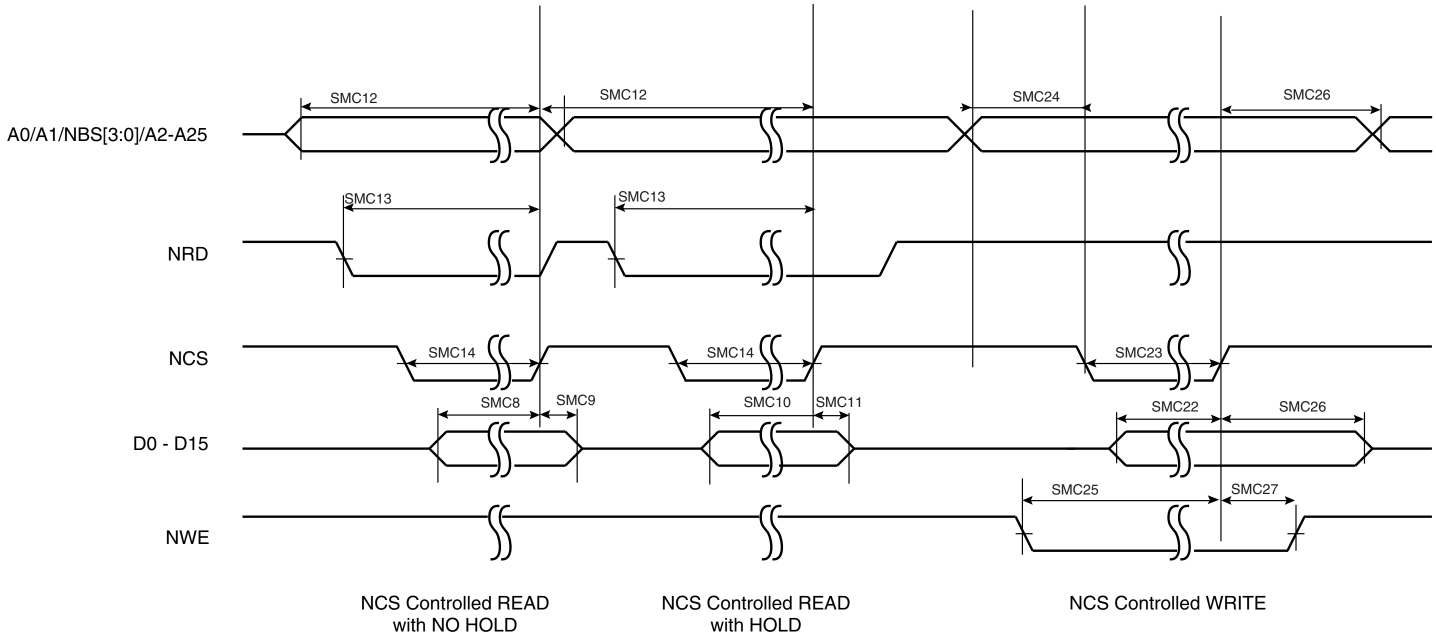
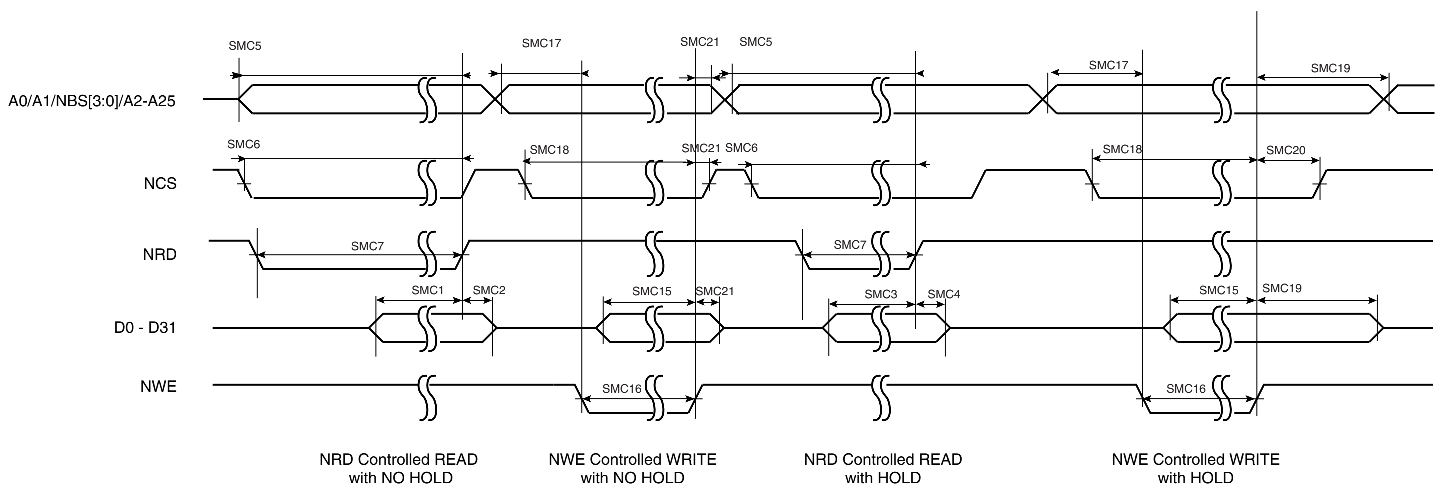


Figure 49-5. SMC Timings - NRD Controlled Read and NWE Controlled Write



49.11.5 DDR SDRAMC Timings

The DDR SDRAM controller satisfies the timings of standard Mobile SDRAM, timings for which are given in Table 49-35, in **MAX and STH corners.**

Timings are given assuming a capacitance load on data, control and address pads:

Table 49-30. Capacitance Load on Control and Address Pads

	Corner	
Supply	**MAX**	**MIN**
3.3V	30 pF	5 pF
1.8V	30 pF	5 pF

Table 49-31. Capacitance Load on Data Pads

	Corner	
Supply	**MAX**	**MIN**
3.3V	50 pF	5 pF
1.8V	20 pF	5 pF

Table 49-32. Capacitance Load on SDCK/SDCKN Pads

Corner	
MAX	**MIN**
10pF	10 pF

The timings of the DDR SDRAM controller support the use of PC100 (3.3V supply) and of Mobile SDRAM (1.8 supply) in MAX corner.

Table 49-33. SDRAM PC100 Characteristics

Parameter	Min 3.3V Supply	Max 3.3V Supply	Unit
SDRAM Controller Clock Frequency		100	MHz
Control/Address/Data In Setup(1)	2		ns
Control/Address/Data In Hold(1)	1		ns
Data Out Access time after SDCK rising		6	ns
Data Out Change time after SDCK rising	3		ns

Table 49-34. Mobile SDRAM Characteristics

Parameter	Min 1.8V Supply	Max 1.8V Supply	Unit
SDRAM Controller Clock Frequency		100 (2)	MHz
Control/Address/Data In Setup(1)	2.5		ns

Table 49-34. Mobile SDRAM Characteristics (Continued)

Parameter	Min 1.8V Supply	Max 1.8V Supply	Unit
Control/Address/Data In Hold[(1)]	1		ns
Data Out Access time after SDCK rising		8	ns
Data Out change time after SDCK rising	1.8		ns

Notes: 1. Control is the set of following timings: A0-A9, A11-A13, SDCKE, SDCS, RAS, CAS, SDA10, BAx, DQMx, and SDWE

2. 100 MHz with CL= 3, 100 MHz with CL= 2

The timings of the DDR SDRAM controller support the use of LPDDR200 Double Data Rate Mobile SDRAM in MAX corner.

Table 49-35. LPDDR200 Characteristics

Parameter	Symbol	Min	Max	Units
SDCK/SDCKN Clock Period	t_{CK}	10		ns
SDCK/SDCKN Clock low or high level	t_{CL},t_{CH}	0.45	0.55	t_{CK}
Control/Address In Setup[(1)]	t_{IS}	1.5		ns
Control/Address In Hold[(1)]CH	t_{IH}	1.5		ns
DQS output access time from SDCK/SDCKN	t_{DQSCK}	2	7.0	ns
DQS[(2)] - Data Out Skew	t_{DQSQ}		0.7	ns
Data out output hold time from DQS[(2)]	t_{QH}	min(t_{CL},t_{CH})-1.0		ns
Data in and DQM[(3)] Setup before DQS edge	t_{DS}	1.1		ns
Data in and DQM[(3)] hold after DQS edge	t_{DH}	1.1		ns

Notes: 1. Control/Address is the set of following signals: A0-A9, A11-A13, SDCKE, SDCS, RAS, CAS, SDA10, BAx and SDWE.

2. Data out, DQS refer to D0-D7, DQS0 or to D8-D15, DQS1.

3. DQM refers to DQM0 or DQM1.

49.11.6 SPI

Timings are given assuming a capacitance load on MISO, SPCK and MOSI:

Table 49-36. Capacitance Load for MISO, SPCK and MOSI

	Corner	
Supply	**MAX**	**MIN**
3.3V	30 pF	5 pF

Figure 49-6. SPI Master Mode with (CPOL = NCPHA = 0) or (CPOL= NCPHA= 1)

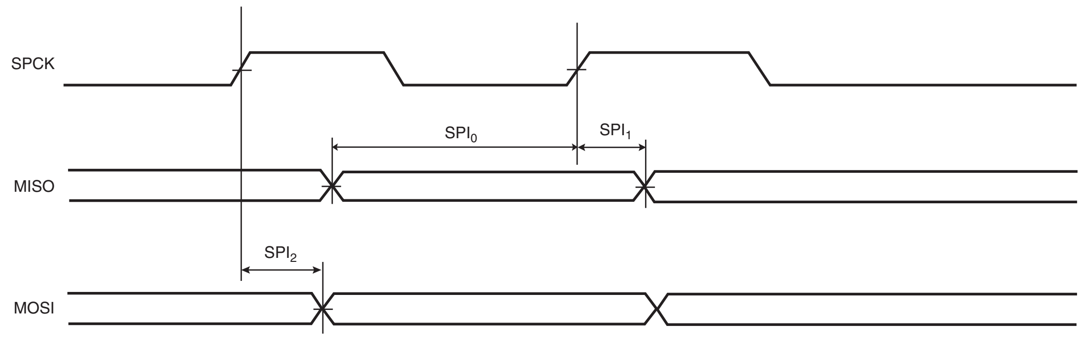


Figure 49-7. SPI Master Mode with (CPOL=0 and NCPHA=1) or (CPOL=1 and NCPHA=0)

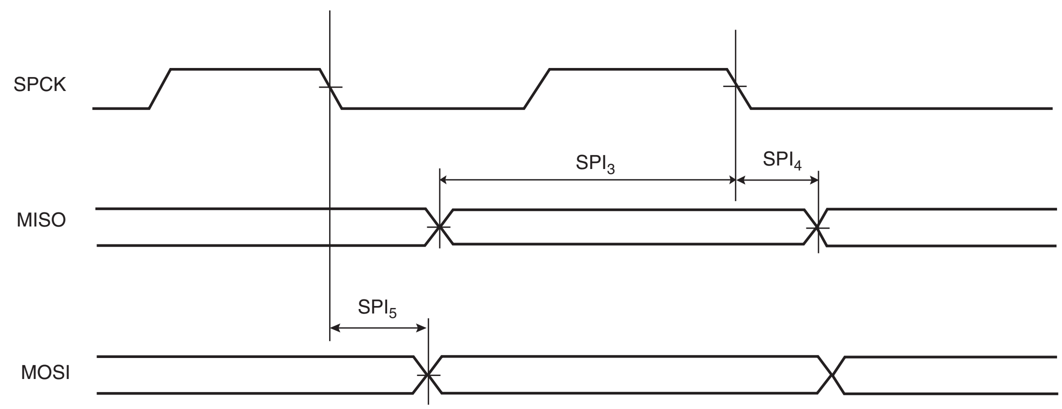


Figure 49-8. SPI Slave Mode with (CPOL=0 and NCPHA=1) or (CPOL=1 and NCPHA=0)

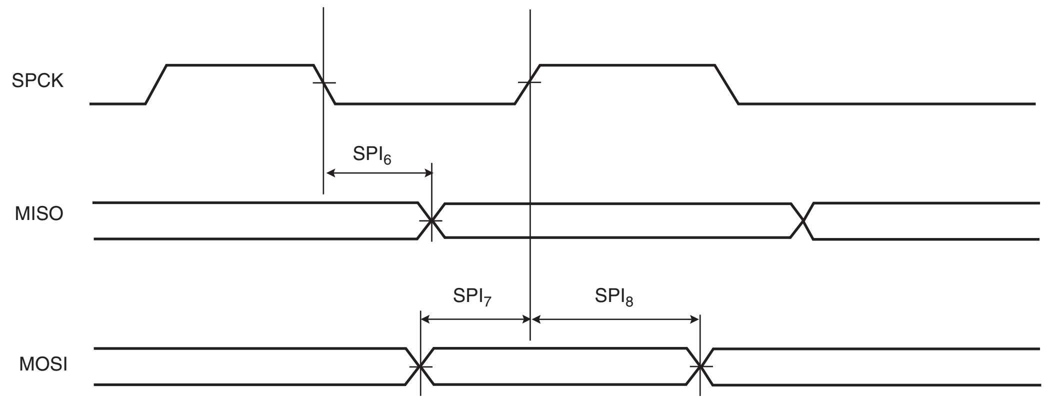


Figure 49-9. SPI Slave Mode with (CPOL = NCPHA = 0) or (CPOL= NCPHA= 1)

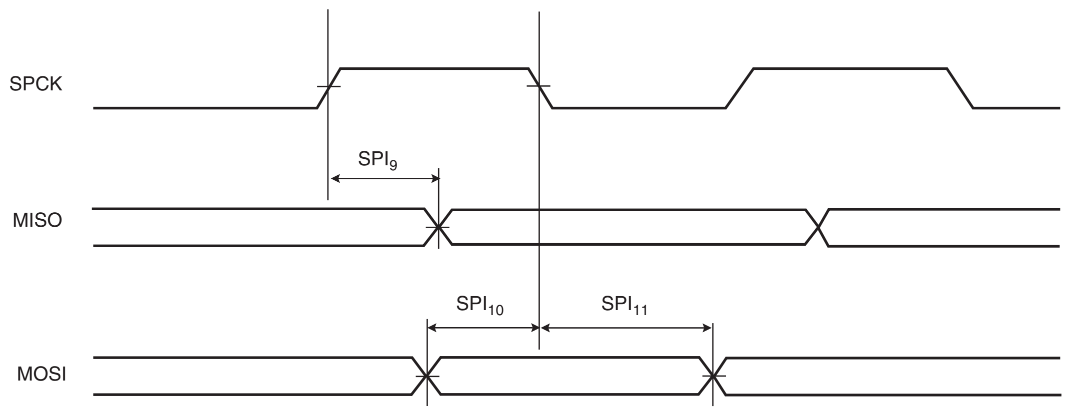


Figure 49-10. SPI Slave Mode - NPCS Timings

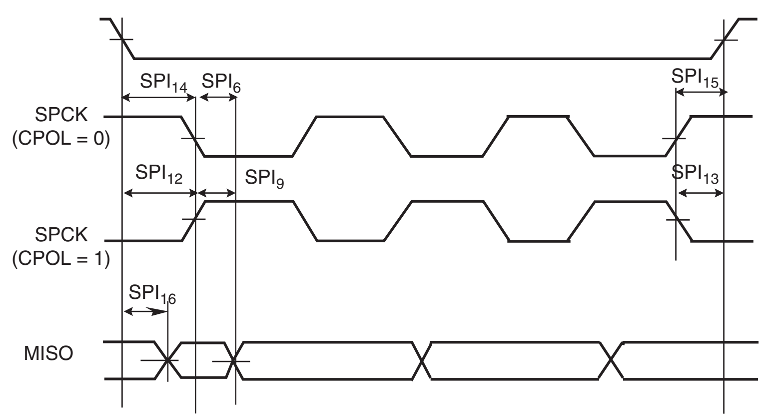


Table 49-37. SPI Timings

Symbol	Parameter	Cond	Min	Max	Units
Master Mode					
SPI_0	MISO Setup time before SPCK rises	MAX corner	18.5 + 0.5*t_{CPMCK}		ns
SPI_1	MISO Hold time after SPCK rises	MAX corner	-17.3 - 0.5* t_{CPMCK}		ns
SPI_2	SPCK rising to MOSI	MAX corner	-2.3		ns
SPI_3	MISO Setup time before SPCK falls	MAX corner	18.8 + 0.5*t_{CPMCK}		ns
SPI_4	MISO Hold time after SPCK falls	MAX corner	-17.6 - 0.5* t_{CPMCK}		ns
SPI_5	SPCK falling to MOSI	MAX corner	-2.3	0.4	ns
Slave Mode					
SPI_6	SPCK falling to MISO	MAX corner	4.0	16.2	ns
SPI_7	MOSI Setup time before SPCK rises	MAX corner	8.9		ns
SPI_8	MOSI Hold time after SPCK rises	MAX corner	-8.4		ns
SPI_9	SPCK rising to MISO	MAX corner	5.2	16.0	ns
SPI_{10}	MOSI Setup time before SPCK falls	MAX corner	9.1		ns
SPI_{11}	MOSI Hold time after SPCK falls	MAX corner	-8.6		ns
SPI_{12}	NPCS0 Setup to SPCK rising	MAX corner	-5.5		ns
SPI_{13}	NPCS0 Hold after SPCK falling	MAX corner	5.5		ns
SPI_{14}	NPCS0 Setup to SPCK falling	MAX corner	-5.3		ns
SPI_{15}	NPCS0 Hold after SPCK rising	MAX corner	5.5		ns
SPI_{16}	NPCS0 falling to MISO valid	MAX corner		12.4	ns

49.11.7 SSC

Timings are given assuming a capacitance load on TK0, TK1, RK0, RK1, RF0, RF1, TD0, TD1:

Table 49-38. Capacitance Load

	Corner	
Supply	**MAX**	**MIN**
3.3V	30pF	5 pF

Figure 49-11. SSC Transmitter, TK and TF in Output

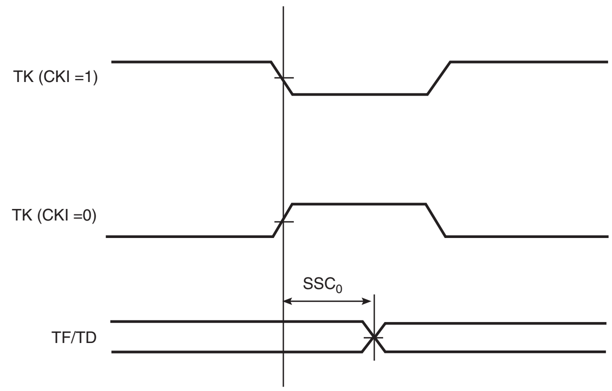


Figure 49-12. SSC Transmitter, TK in Input and TF in Output

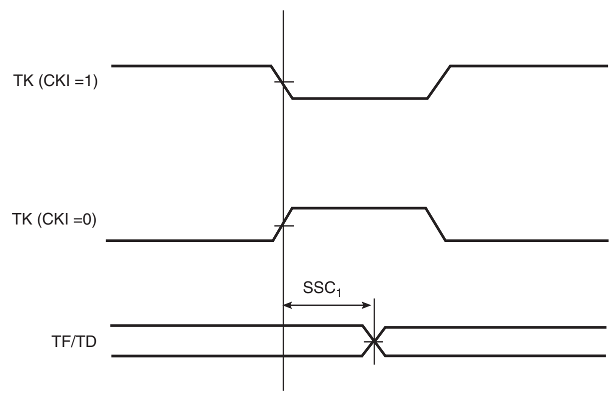


Figure 49-13. SSC Transmitter, TK in Output and TF in Input

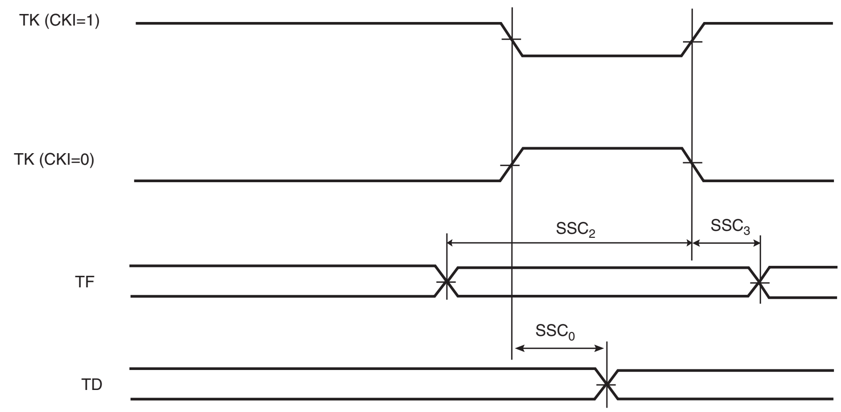


Figure 49-14. SSC Transmitter, TK and TF in Input

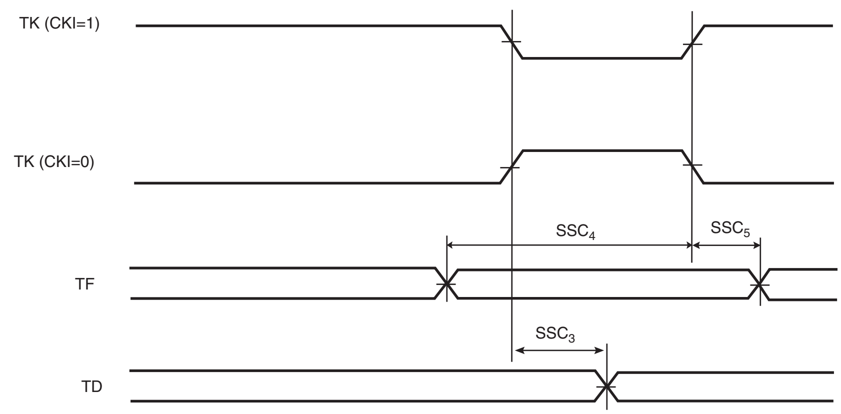


Figure 49-15. SSC Receiver RK and RF in Input

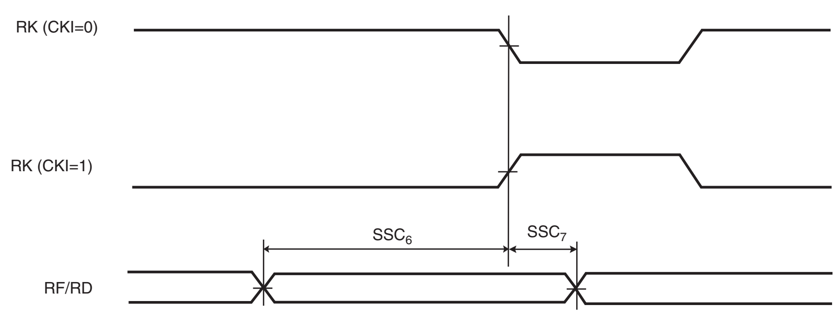


Figure 49-16. SSC Receiver, RK in input and RF in Output

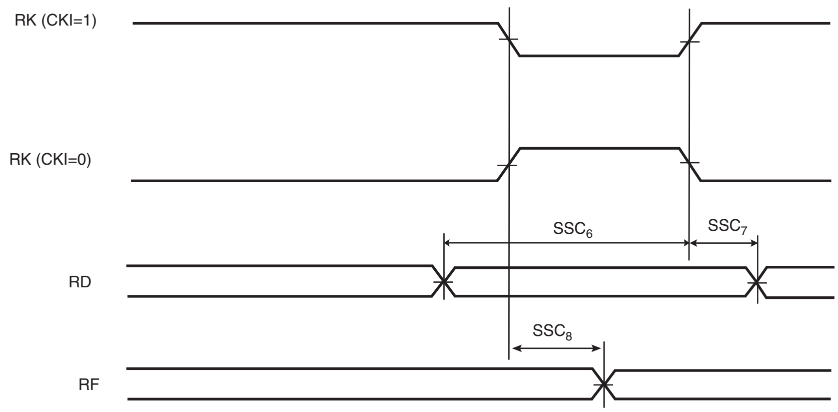


Figure 49-17. SSC Receiver, RK and RF in Output

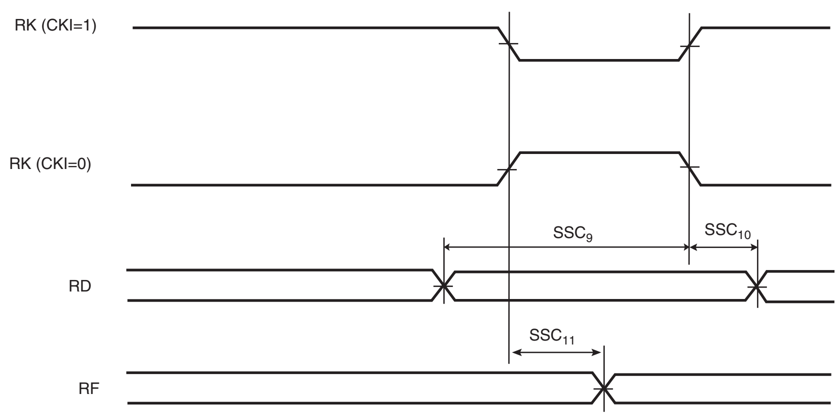


Figure 49-18. SSC Receiver, RK in Output and RF in Input

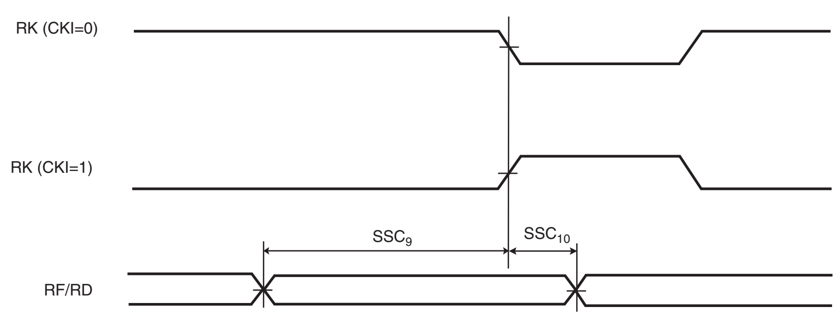


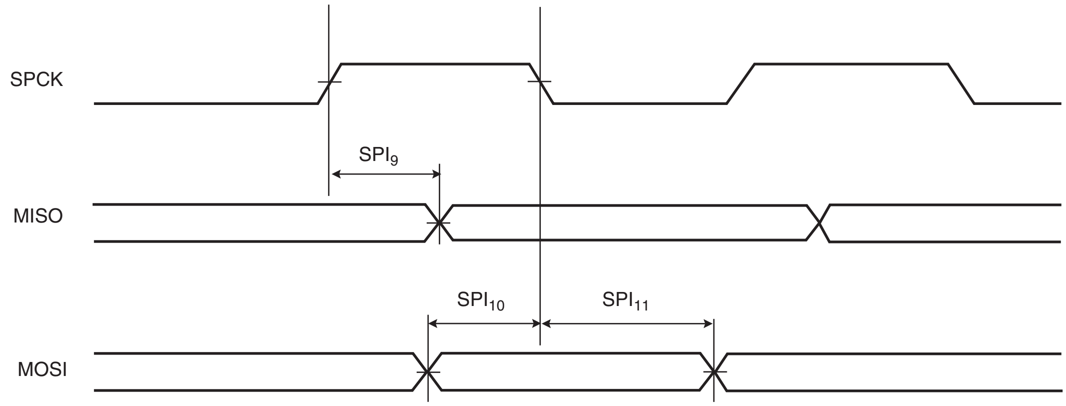
Table 49-39. SSC Timings

Symbol	Parameter	Conditions	Min	Max	Units
		Transmitter			
SSC_0	TK edge to TF/TD (TK output, TF output)	MAX corner, VDDIO = 1.8V	-0.8 ((2)	3.7 (2)	ns
		MAX corner, VDDIO = 3.3V	-1.4 (2)	4.3 (2)	ns
SSC_1	TK edge to TF/TD (TK input, TF output)	MAX corner, VDDIO = 1.8V	4.8 (2)	15.8 (2)	ns
		MAX corner, VDDIO = 3.3V	4.8 (2)	16.4 (2)	ns
SSC_2	TF setup time before TK edge (TK output)	MAX corner, VDDIO = 1.8V	18.4 - t_{CPMCK}		ns
		MAX corner, VDDIO = 3.3V	19.0 - t_{CPMCK}		ns
SSC_3	TF hold time after TK edge (TK output)	MAX corner, VDDIO = 1.8V	t_{CPMCK} - 6.1		ns
		MAX corner, VDDIO = 3.3V	t_{CPMCK} - 6.1		ns
SSC_4 (1)	TK edge to TF/TD (TK output, TF input)	MAX corner, VDDIO = 1.8V	-0.8(1)(2)	2 * t_{CPMCK} + 3.5(1)(2)	ns
		MAX corner, VDDIO = 3.3V	-1.4(1)(2)	2 * t_{CPMCK} + 4.1(1)(2)	ns
SSC_5	TF setup time before TK edge (TK input)	MAX corner, VDDIO = 1.8V	0		ns
		MAX corner, VDDIO = 3.3V	0		ns
SSC_6	TF hold time after TK edge (TK input)	MAX corner, VDDIO = 1.8V	t_{CPMCK}		ns
		MAX corner, VDDIO = 3.3V	t_{CPMCK}		ns
SSC_7(1)	TK edge to TF/TD (TK input, TF input)	MAX corner, VDDIO = 1.8V	4.8(1)	3 * t_{CPMCK} + 14.7(1)	ns
		MAX corner, VDDIO = 3.3V	4.8(1)	3 * t_{CPMCK} + 15.3(1)	ns
		Receiver			
SSC_8	RF/RD setup time before RK edge (RK input)	MAX corner, VDDIO = 1.8V	0		ns
		MAX corner, VDDIO = 3.3V	0		ns
SSC_9	RF/RD hold time after RK edge (RK input)	MAX corner, VDDIO = 1.8V	t_{CPMCK}		ns
		MAX corner, VDDIO = 3.3V	t_{CPMCK}		ns
SSC_{10}	RK edge to RF (RK input)	MAX corner, VDDIO = 1.8V	4.7(2)	16.9(2)	ns
		MAX corner, VDDIO = 3.3V	4.7(2)	17.5(2)	ns
SSC_{11}	RF/RD setup time before RK edge (RK output)	MAX corner, VDDIO = 1.8V	19.8 - t_{CPMCK}		ns
		MAX corner, VDDIO = 3.3V	20.4 - t_{CPMCK}		ns
SSC_{12}	RF/RD hold time after RK edge (RK output)	MAX corner, VDDIO = 1.8V	t_{CPMCK} - 6.1		ns
		MAX corner, VDDIO = 3.3V	t_{CPMCK} - 6.1		ns
SSC_{13}	RK edge to RF (RK output)	MAX corner, VDDIO = 1.8V	-1.3	1.4	ns
		MAX corner, VDDIO = 3.3V	-2.0	2.0	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 49-19 illustrates Min and Max accesses for SSC0. The same applies for SSC1, SSC4, and SSC7, SSC10 and SSC13.

Figure 49-19. Min and Max Access for SSC0



49.11.8 ISI Timings

Timings are given assuming the following capacitance load on ISI_D0..ISI_D11, ISI_PCK, ISI_HSYNC, ISI_VSYNC, ISI_MCK:

Table 49-40. Capacitance Load

	Corner	
Supply	**MAX**	**MIN**
3.3V	30 pF	5 pF

Figure 49-20. ISI Timing Diagram

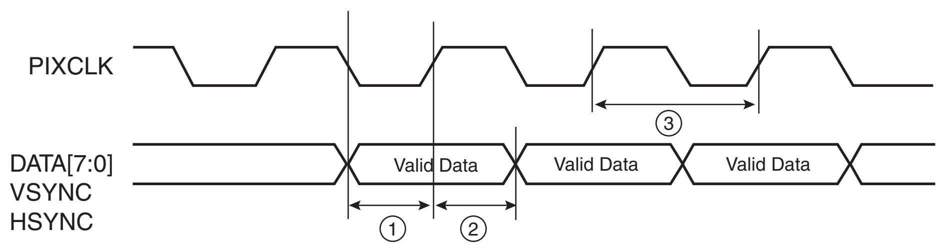


Table 49-41. ISI Timings with Peripheral Supply 3.3V

Symbol	Parameter	Min	Max	Units
ISI_1	DATA/VSYNC/HSYNC setup time	8.4		ns
ISI_2	DATA/VSYNC/HSYNC hold time	1.7		ns
ISI_3	PIXCLK frequency		54MHz	MHz

Table 49-42. ISI Timings with Peripheral Supply 1.8V

Symbol	Parameter	Min	Max	Units
ISI_1	DATA/VSYNC/HSYNC setup time	2.2		ns
ISI_2	DATA/VSYNC/HSYNC hold time	3.0		ns
ISI_3	PIXCLK frequency		54MHz	MHz

49.11.9 MCI Timings

Capacitance loads on data and clock are given in Table 49-43.

Figure 49-21. MCI Timings

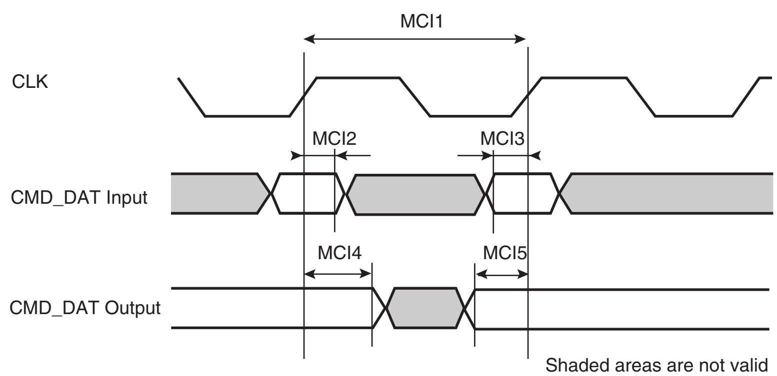


Table 49-43. MCI Timings

Symbol	Parameter	CLoad	Min	Max	Units
MCI_1	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_2	Input hold time		3		ns

Table 49-43. MCI Timings

Symbol	Parameter	CLoad	Min	Max	Units
MCI_3	Input setup time		3		ns
MCI_4	Output change after CLK rising		5		ns
MCI_5	Output valid before CLK rising		5		ns

49.11.10 EMAC Timings

The Ethernet controller satisfies the timings of standard given in Table 49-45 and Table 49-46, in **MAX corner.**

Timings are given assuming the following capacitance load on data and clock:

Table 49-44. Capacitance Load on Data, Clock Pads

	Corner	
Supply	**Max**	**Min**
3.3V	30 pF	5 pF

Table 49-45. EMAC Signals Relative to EMDC

Symbol	Parameter	Min (ns)	Max (ns)
$EMAC_1$	Setup for EMDIO from EMDC rising	10 ns	
$EMAC_2$	Hold for EMDIO from EMDC rising	10 ns	
$EMAC_3$	EMDIO toggling from EMDC rising	0 ns	300 ns

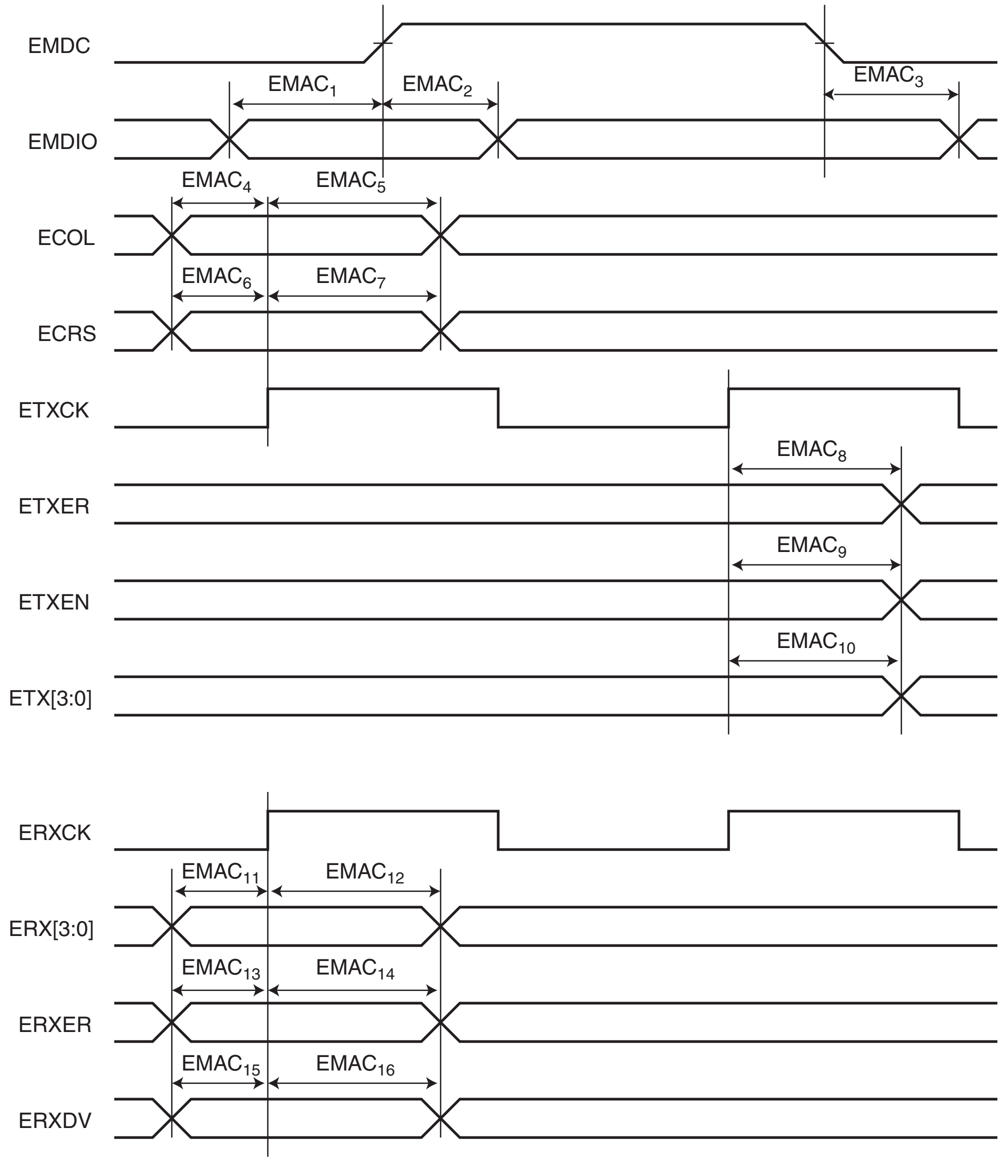
49.11.10.1 MII Mode

Table 49-46. EMAC MII Specific Signals

Symbol	Parameter	Min (ns)	Max (ns)
$EMAC_4$	Setup for ECOL from ETXCK rising	10	
$EMAC_5$	Hold for ECOL from ETXCK rising	10	
$EMAC_6$	Setup for ECRS from ETXCK rising	10	
$EMAC_7$	Hold for ECRS from ETXCK rising	10	
$EMAC_8$	ETXER toggling from ETXCK rising	10	25
$EMAC_9$	ETXEN toggling from ETXCK rising	10	25
$EMAC_{10}$	ETX toggling from ETXCK rising	10	25
$EMAC_{11}$	Setup for ERX from ERXCK	10	
$EMAC_{12}$	Hold for ERX from ERXCK	10	
$EMAC_{13}$	Setup for ERXER from ERXCK	10	
$EMAC_{14}$	Hold for ERXER from ERXCK	10	
$EMAC_{15}$	Setup for ERXDV from ERXCK	10	
$EMAC_{16}$	Hold for ERXDV from ERXCK	10	

Note: 1. VDDIO from 3.0V to 3.6V, maximum external capacitor = 20 pF

Figure 49-22. EMAC MII Mode

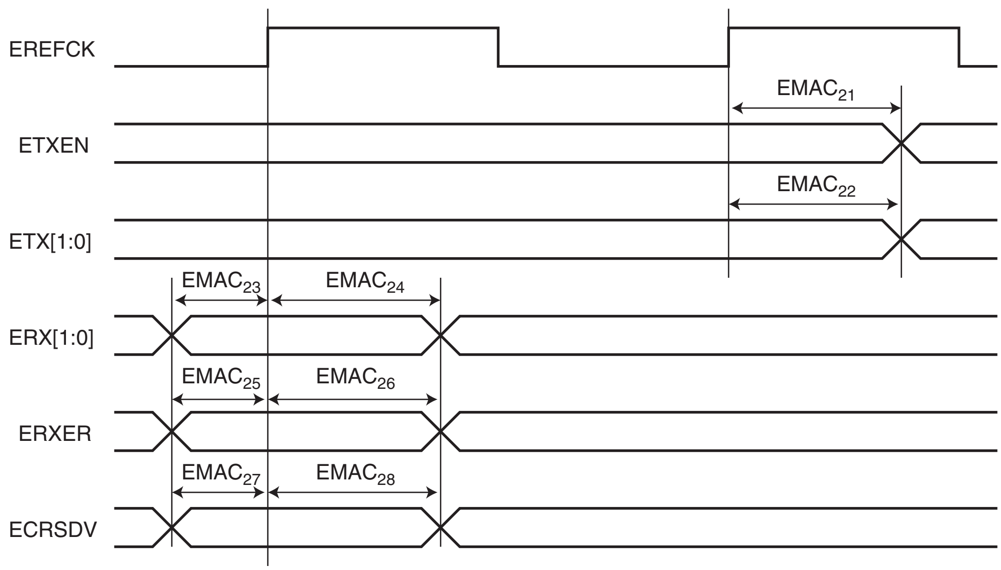


49.11.10.2 RMII Mode

Table 49-47. RMII Mode

Symbol	Parameter	Min (ns)	Max (ns)
$EMAC_{21}$	ETXEN toggling from EREFCK rising	2	16
$EMAC_{22}$	ETX toggling from EREFCK rising	2	16
$EMAC_{23}$	Setup for ERX from EREFCK rising	4	
$EMAC_{24}$	Hold for ERX from EREFCK rising	2	
$EMAC_{25}$	Setup for ERXER from EREFCK rising	4	
$EMAC_{26}$	Hold for ERXER from EREFCK rising	2	
$EMAC_{27}$	Setup for ECRSDV from EREFCK rising	4	
$EMAC_{28}$	Hold for ECRSDV from EREFCK rising	2	

Figure 49-23. EMAC RMII Timings



50. AT91CAP9S Mechanical Characteristics

50.1 Package Drawing

Figure 50-1. 400-ball LFBGA Package Drawing

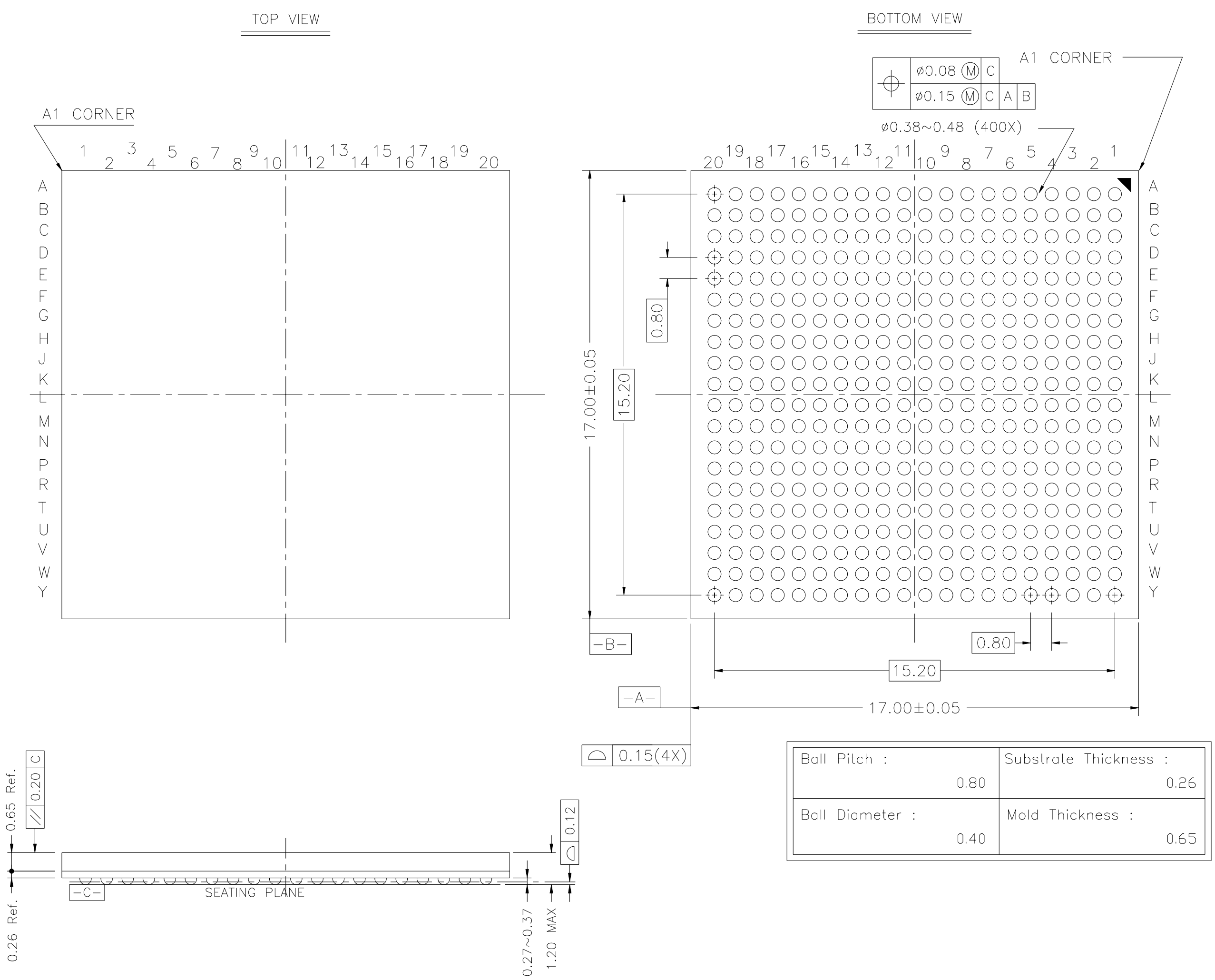


Table 50-1. Soldering Information

Ball Land	0.4 mm +/- 0.05
Soldering Mask Opening	0.275 mm +/- 0.03

Table 50-2. Device and 400-ball LFBGA Package Maximum Weight

	mg

Table 50-3. 400-ball LFBGA Package Characteristics

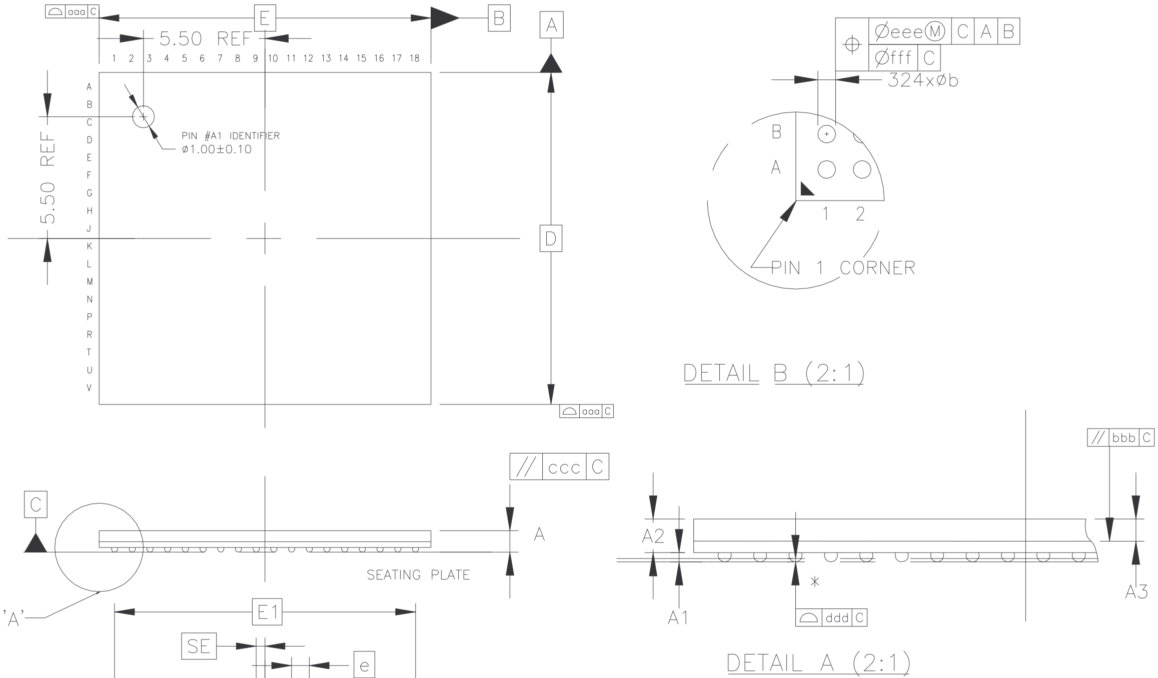
Moisture Sensitivity Level	3

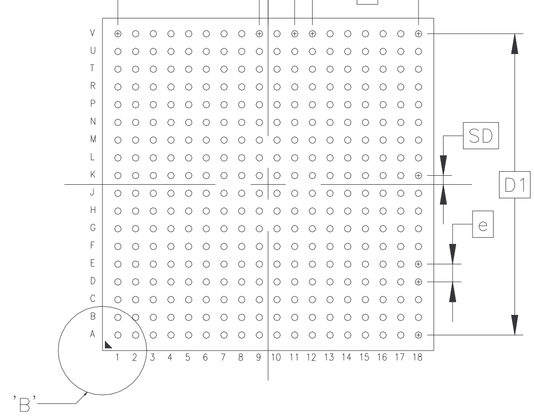
Table 50-4. Package Reference

JEDEC Drawing Reference	MO-210
JESD97 Classification	e1

This package respects the recommendations of the NEMI User Group.

Figure 50-2. 324-ball TFBGA Package





ALL DIMENSIONS ARE IN MILLIMETERS.						
SYMBOL	MILLIMETER			INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	---	---	1.20	---	----	0.0472
A1	0.16	0.21	0.26	0.0063	0.0083	0.0102
A2	0.72	0.76	0.80	0.0283	0.0299	0.0315
A3	0.50 BASIC			0.0197 BASIC		
D	14.95	15.00	15.05	0.5886	0.5906	0.5926
D1	13.60 BASIC			0.5354 BASIC		
E	14.95	15.00	15.05	0.5866	0.5906	0.5926
E1	13.60 BASIC			0.5354 BASIC		
SD	0.40 BASIC			0.0157 BASIC		
SE	0.40 BASIC			0.0157 BASIC		
e	0.80 BASIC			0.0315		
b	0.25	0.30	0.35	0.0098	0.0118	0.0138
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		

50.2 Soldering Profile

Table 50-5 gives the recommended soldering profile from J-STD-020C.

Table 50-5. 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.

51. AT91CAP9S Ordering Information

Table 51-1. CAP9S Ordering Information

Ordering Code	Package	Package Type	Temperature Operating Range
AT91CAP9S250A-CJ	BGA400	RoHS Compliant	Industrial -40°C to 85°C
AT91CAP9S500A-CJ			
AT91CAP9S250A-CJ	BGA324	RoHS Compliant	Industrial -40°C to 85°C
AT91CAP9S500A-CJ			

52. AT91CAP9S Errata

52.1 Marking

All devices are marked with the Atmel logo and the ordering code.
Additional marking has the following format:

```
YYWW      V□
XXXXXXXXX        ARM
```

where

- “YY”: manufactory year
- “WW”: manufactory week
- “V”: revision
- “XXXXXXXXX”: lot number

AT91CAP9S errata is available on Atmel’s web site: http://www.atmel.com/products/AT91CAP/ (lit° 6459)

53. Revision History

In the following pages, the most recent revision history table appears first.

Document Ref. 6246C	Comments	Change Request Ref.
	Datasheet update corresponds to the most recent version of the CAP9S and includes the following:. Product **Overview** updated (pages 1 to 54) includes new sections: Section 8.1.1.2"Internal ROM" Section 8.1.2"Boot Strategies" Section 8.2.5"NAND Flash Error Corrected Code Controller" Section 9.5"Slow Clock Selection" includes Updates to: "Features", – 32 Kbyte Internal ROM – 128-byte FIFOs – Selectable 32768 Hz Low-power Oscillator or Internal Low-power RC Oscillator •Two MultiMedia Card Interfaces Section 4.4"324-ball TFBGA Package Pinout", text updated with PD0-PD10 Table 3-1, Signal List, "DDR/SDRAM Controller", updated. Section 6.3"Reset Pins", updated. Section 8."Memories", 2nd paragraph updated EBI/external chip selects. Section 8.1"Embedded Memories", 32 Kbyte ROM – double cycle access..... Section 8.2.1"External Bus Interface", updated. Section 9.4"Clock Generator", updated. Section 9.13"Chip Identification", updated. Figure 2-1, Block Diagram updated with SDCS signal	5553/5242
	Section 14."Boot Program", updated	5553
	Section 19."General Purpose Backup Registers (GPBR)", added to datasheet.	rfo
	Bus MATRIX: Section 21.5"EBI Pad Output Strength", added to datasheet Section 21.7.3"EBI Chip Select Assignment Register", some bitfieds updated with output strength configuration.	rfo
	Clock Generator: Section 29.4.4"Slow Clock Configuration Register" added to datasheet. Section 30.9.11"PMC Clock Generator PLL B Register", updated USBDIV field (28-31) division factor of USB clocks is USBDIV+1	rfo
	PMC: Section 30.9.1"PMC System Clock Enable Register", updated with DDR management Section 30.9.2"PMC System Clock Disable Register", updated with DDR management Section 30.9.12"PMC Master Clock Register", PDIV field added, manages processor clock divider. Section 30.9.18"Write Protect Mode Register", added to datasheet Section 30.9.19"Write Protect Status Register", added to datasheet	rfo
	Section 49."AT91CAP9 Electrical Characteristics", update corresponds to current version of the CAP9S.	5553
	Section 52."AT91CAP9S Errata", errata sheet referenced with link.	rfo

Document Ref. 6246B	Comments	Change Request Ref.
	"Features" PIOD typo fixed. "Required Power Supplies:" on page 3, important update and new supplies added. "One 8-channel, 10-bit Analog-to-Digital Converter (ADC)", added to features Section 10.4.15"Analog-to-digital Converter", added.	4490
	Section 10.2.2"DMA Controller Request Signals", section added. Section 10.4.11"USB High Speed Device Port", endpoints 3 and 4 are "HS isochronous capable". Figure 4-1, "400-ball LFBGA Package Outline and Marking (Top View)," on page 11, updated with package marking.	rfo
	"Features" 32-ball BGA Package added. Section 4."Package and Pinout", 324-ball TFBGA package added. Figure 50-2, "324-ball TFBGA Package," on page 1115, added. Figure 2-1 on page 4, Block Diagram updated Table 3-1, "Signal Description List," 324-ball TFBGA package options added in note to MPBLOCK parameters. Table 4-1 on page 12, pin W2, "OWAIT" replaced by "BCOWAIT". Section 10.3.1 on page 42, Section 10.3.2 on page 43, Section 10.3.3 on page 44, Section 10.3.4 on page 45, Multiplexing on PIO I/Os updated with 324-ball TFBGA options. Section 10.4.15"Analog-to-digital Converter", Endpoint information corrected.	4916
	Figure 8-1, "AT91CAP9S500A/AT91CAP9S250A Memory Mapping," on page 26, note associated with "boot memory" updated. Table 8-1, "Internal Memory Mapping," on page 28, updated.	4263
	ADC: Section 47.6.2 "ADC Mode Register", PRESCAL and STARTUP bit fields expanded "TRGSEL: Trigger Selection" bit description, Selected TRGSEL column updated.	4430 spec
	CAN: Section 40.6.4.6 "Error Interrupt Handler" on page 743, added to datasheet. Section 40.8.5"CAN Status Register", cross reference to new chapter added to the bit descriptions: WARN, BOFF, ERRA, ERRP.	4736
	DMAC: Updated IEN bit description in Section 27.5.18 "DMAC Channel x [x = 0..3] Control B Register" on page 369 and FIFOCFG bit description in Section 27.5.19 "DMAC Channel x [x = 0..3] Configuration Register" on page 371. Section 26.5 "DMA Controller (DMAC) User Interface", the register mapping table updated with indexed channels.	RTt XCn
	DDRSDRC: All instances of mobile SDR-SDRAM and mobile DDR-SDRAM changed to low-power SDR-SDRAM and low-power DDR-SDRAM. Section 24.6.1 "DDRSDRC Mode Register" on page 245, changed MODE bit description Section 24.3.1 on page 225, Section 24.3.3 "Low-power DDR-SDRAM Initialization" on page 227, Step (1), corrected reference to Memory Device Register.	4500 4594 rfo
	LCDC: Section 45. "LCD Controller (LCDC)" added to datasheet	4808

Document Ref. 6246B	Comments (Continued)	Change Request Ref.
	PWM: Section 41.6“Pulse Width Modulation Controller (PWM) User Interface” Channel-dependent registers have been indexed in the Register Mapping table. See Section 41.6.9“PWM Channel Mode Register”, Section 41.6.10“PWM Channel Duty Cycle Register”, Section 41.6.11“PWM Channel Period Register”, Section 41.6.12“PWM Channel Counter Register” and Section 41.6.13“PWM Channel Update Register”	4486
	SHDWC: Figure 20-1, “Shutdown Controller Block Diagram,” on page 131, corrected register names	4734
	SSC: Section 36.8.3 “SSC Receive Clock Mode Register”, corrected bit name to STTDLY	4778
	TC: Section 38.6 “Timer Counter (TC) User Interface” previous two Register Mapping tables consolidated Table 38-4 on page 711, From Section 38.6.3 on page 714 to Section 38.6.13 on page 728, Register Names updated and functional value of WAVE is given, when relevant. Section 38.6.2 “TC Block Mode Register” typo corrected in bit fields 2 and 3. Section 38.6.4 “TC Channel Mode Register: Capture Mode”, bit field 15 updated.	4583
	TWI: fixed typo in ARBLAST bit fields, Section 34.10.7 “TWI Interrupt Enable Register”, Section 34.10.8 “TWI Interrupt Disable Register”, Section 34.10.9 “TWI Interrupt Mask Register” inserted EOSACC bit field description, Section 34.10.7 “TWI Interrupt Enable Register”	4582 4586
	UDPHS: Section 44.2 “Block Diagram” updated block diagram Section 44.5 “USB High Speed Device Port (UDPHS) User Interface”, endpoint and DMA channel registers indexed. Figure 44-17, “UDPHS Interrupt Control Interface,” on page 922, EPT_INT_x changed to EPT_x Section 44.4.12.4 “From Powered State to Default State (Reset)”, EPT_INT_x changed to EPT_x Like wise in bit field descriptions: “EPT_x: Endpoint x Interrupt Enable” on page 936 and “INTDIS_DMA: Interrupt Disables DMA” on page 949	4511
	UHP: Section 43.1 “Overview”**,** added a hyperlink.	4364
	USART: Section 36.6.2“Receiver and Transmitter Control”, t he 4th paragraph, updated by replacing 2nd sentence. Section 36.6.3.1“Transmitter Operations” last paragraph updated Section 36.6.5“IrDA Mode”, updated with instruction to receive IrDA signals Section 36.2“Block Diagram”, signal directions from pads to PIO updated.	4367 rfo 4912 4905
	Mechanical Characteristics: Removed Thermal Considerations and references to junction temperature.	4659

Document Ref 6246A	Comments	Change Request Ref.
	First issue.	

Table of Contents

Features ***1***

1 Description ***3***

2 AT91CAP9S500A/AT91CAP9S250A Block Diagram ***4***

3 Signal Description ***5***

4 Package and Pinout ***11***

4.1 400-ball LFBGA Package Outline 11
4.2 400-ball LFBGA Package Pinout 12
4.3 324-ball TFBGA Package Outline 15
4.4 324-ball TFBGA Package Pinout 15

5 Power Considerations ***16***

5.1 Power Supplies 16
5.2 Power Consumption 16
5.3 Programmable I/O Lines Power Supplies 17

6 I/O Line Considerations ***18***

6.1 JTAG Port Pins 18
6.2 Test Pin 18
6.3 Reset Pins 18
6.4 PIO Controllers 18
6.5 Shutdown Logic Pins 18

7 Processor and Architecture ***19***

7.1 ARM926EJ-S Processor 19
7.2 Bus Matrix 19
7.3 Matrix Masters 20
7.4 Matrix Slaves 20
7.5 Master-to-Slave Access 21
7.6 Peripheral DMA Controller 23
7.7 DMA Controller 24
7.8 Debug and Test Features 24

8 Memories ***26***

8.1 Embedded Memories 27
8.2 External Memories 29

9 System Controller 32
9.1 System Controller Block Diagram 33
9.2 Reset Controller 34
9.3 Shutdown Controller 34
9.4 Clock Generator 34
9.5 Slow Clock Selection 35
9.6 Power Management Controller 35
9.7 Periodic Interval Timer 36
9.8 Watchdog Timer 36
9.9 Real-time Timer 36
9.10 General-Purpose Backup Registers 36
9.11 Advanced Interrupt Controller 37
9.12 Debug Unit 37
9.13 Chip Identification 38
9.14 PIO Controllers 38

10 Peripherals 39
10.1 User Interface 39
10.2 Identifiers 39
10.3 Peripheral Signal Multiplexing on I/O Lines 41
10.4 Embedded Peripherals 46

11 Metal Programmable Block 51
11.1 Internal Connectivity 51
11.2 External Connectivity 53
11.3 Prototyping Solution 53

12 ARM926EJ-S Processor Overview 55
12.1 Overview 55
12.2 Block Diagram 56
12.3 ARM9EJ-S Processor 57
12.4 CP15 Coprocessor 65
12.5 Memory Management Unit (MMU) 67
12.6 Caches and Write Buffer 69
12.7 Bus Interface Unit 71

13 Debug and Test 73
13.1 Description 73
13.2 Block Diagram 74

13.3 Application Examples 75
13.4 Debug and Test Pin Description 76
13.5 Functional Description 76

14 Boot Program 79
14.1 Description 79
14.2 Flow Diagram 79
14.3 Device Initialization 80
14.4 NVM Boot 81
14.5 SAM-BA Monitor 88

15 Reset Controller (RSTC) 93
15.1 Description 93
15.2 Block Diagram 93
15.3 Functional Description 93
15.4 Reset Controller (RSTC) User Interface 103

16 Real-time Timer (RTT) 107
16.1 Description 107
16.2 Block Diagram 107
16.3 Functional Description 107
16.4 Real-time Timer (RTT) User Interface 109

17 Periodic Interval Timer (PIT) 113
17.1 Description 113
17.2 Block Diagram 113
17.3 Functional Description 114
17.4 Periodic Interval Timer (PIT) User Interface 116

18 Watchdog Timer (WDT) 121
18.1 Description 121
18.2 Block Diagram 121
18.3 Functional Description 122
18.4 Watchdog Timer (WDT) User Interface 124

19 General Purpose Backup Registers (GPBR) 129
19.1 Description 129
19.2 General Purpose Backup Registers (GPBR) User Interface 129

20 Shutdown Controller (SHDWC) 131
20.1 Description 131

20.2 Block Diagram 131
20.3 I/O Lines Description 131
20.4 Product Dependencies 131
20.5 Functional Description 131
20.6 Shutdown Controller (SHDWC) User Interface 133

21 Bus Matrix 137

21.1 Description 137
21.2 Memory Mapping 137
21.3 Special Bus Granting Techniques 137
21.4 Arbitration 138
21.5 EBI Pad Output Strength 140
21.6 Bus Matrix (MATRIX) User Interface 141
21.7 Chip Configuration User Interface 147

22 External Bus Interface (EBI) 155

22.1 Description 155
22.2 I/O Lines Description 157
22.3 Application Example 159
22.4 Product Dependencies 162
22.5 Functional Description 162
22.6 Implementation Examples 170

23 Static Memory Controller (SMC) 181

23.1 Description 181
23.2 I/O Lines Description 181
23.3 Multiplexed Signals 181
23.4 Application Example 182
23.5 Product Dependencies 182
23.6 External Memory Mapping 183
23.7 Connection to External Devices 183
23.8 Standard Read and Write Protocols 188
23.9 Automatic Wait States 196
23.10 Data Float Wait States 201
23.11 External Wait 205
23.12 Slow Clock Mode 211
23.13 Asynchronous Page Mode 214
23.14 Static Memory Controller (SMC) User Interface 217

24 DDR/SDR SDRAM Controller (DDRSDRC) 223

24.1 Description 223
24.2 DDRSDRC Module Diagram 224
24.3 Product Dependencies 225
24.4 Functional Description 228
24.5 Software Interface/SDRAM Organization, Address Mapping 242
24.6 DDR/SDR SDRAM Controller (DDRSDRC) User Interface 244

25 Burst Cellular RAM Controller (BCRAMC) 257

25.1 Description 257
25.2 BCRAMC Block Diagram 258
25.3 Product Dependencies 259
25.4 Functional Description 260
25.5 Burst Cellular RAM Controller (BCRAMC) User Interface 269

26 Error Corrected Code (ECC) Controller 279

26.1 Description 279
26.2 Block Diagram 279
26.3 Functional Description 279
26.4 Error Corrected Code Controller (ECC) User Interface 284
26.5 Registers for 1 ECC for a page of 512/1024/2048/4096 bytes 295
26.6 Registers for 1 ECC per 512 bytes for a page of 512/2048/4096 bytes, 8-bit word 297
26.7 Registers for 1 ECC per 256 bytes for a page of 512/2048/4096 bytes, 8-bit word 305

27 DMA Controller (DMAC) 321

27.1 Description 321
27.2 Block Diagram 322
27.3 Functional Description 323
27.4 DMAC Software Requirements 349
27.5 DMA Controller (DMAC) User Interface 351

28 Peripheral DMA Controller (PDC) 375

28.1 Description 375
28.2 Block Diagram 376
28.3 Functional Description 376
28.4 Peripheral DMA Controller (PDC) User Interface 379

29 Clock Generator 387

29.1 Overview 387

29.2 Slow Clock Crystal Oscillator 387

29.3 Slow Clock RC Oscillator 387

29.4 Slow Clock Selection 387

29.5 Main Oscillator 390

29.6 Divider and PLL Block 392

30 Power Management Controller (PMC) 394

30.1 Overview 394

30.2 Master Clock Controller 394

30.3 Processor Clock Controller 395

30.4 USB Clock Controller 395

30.5 Peripheral Clock Controller 396

30.6 Programmable Clock Output Controller 396

30.7 Programming Sequence 396

30.8 Clock Switching Details 401

30.9 Power Management Controller (PMC) User Interface 405

31 Advanced Interrupt Controller (AIC) 427

31.1 Description 427

31.2 Block Diagram 428

31.3 Application Block Diagram 428

31.4 AIC Detailed Block Diagram 428

31.5 I/O Line Description 429

31.6 Product Dependencies 429

31.7 Functional Description 430

31.8 Advanced Interrupt Controller (AIC) User Interface 440

32 Debug Unit (DBGU) 453

32.1 Description 453

32.2 Block Diagram 454

32.3 Product Dependencies 455

32.4 UART Operations 455

32.5 Debug Unit (DBGU) User Interface 463

33 Parallel Input/Output Controller (PIO) 477

33.1 Description 477

33.2 Block Diagram 478

33.3 Product Dependencies 479

33.4 Functional Description 480
33.5 I/O Lines Programming Example 485
33.6 Parallel Input/Output Controller (PIO) User Interface 486

34 Serial Peripheral Interface (SPI) 503

34.1 Description 503
34.2 Block Diagram 504
34.3 Application Block Diagram 504
34.4 Signal Description 505
34.5 Product Dependencies 505
34.6 Functional Description 507
34.7 Serial Peripheral Interface (SPI) User Interface 516

35 Two-wire Interface (TWI) 529

35.1 Description 529
35.2 List of Abbreviations 529
35.3 Block Diagram 530
35.4 Application Block Diagram 530
35.5 I/O Lines Description 531
35.6 Product Dependencies 531
35.7 Functional Description 532
35.8 Master Mode 533
35.9 Multi-master Mode 545
35.10 Slave Mode 548
35.11 Two-wire Interface (TWI) User Interface 556

36 Universal Synchronous Asynchronous Receiver Transceiver (USART) 571

36.1 Description 571
36.2 Block Diagram 572
36.3 Application Block Diagram 573
36.4 I/O Lines Description 573
36.5 Product Dependencies 574
36.6 Functional Description 575
36.7 Universal Synchronous Asynchronous Receiver Transmitter (USART) User Interface 605

37 Synchronous Serial Controller (SSC) 625

37.1 Description 625

37.2 Block Diagram 626
37.3 Application Block Diagram 626
37.4 Pin Name List 627
37.5 Product Dependencies 627
37.6 Functional Description 628
37.7 SSC Application Examples 640
37.8 Synchronous Serial Controller (SSC) User Interface 642

38 AC97 Controller (AC97C) 665

38.1 Overview 665
38.2 Block Diagram 665
38.3 Pin Name List 666
38.4 Application Block Diagram 666
38.5 Product Dependencies 667
38.6 Functional Description 668
38.7 AC97 Controller (AC97C) User Interface 679

39 Timer Counter (TC) 697

39.1 Description 697
39.2 Block Diagram 698
39.3 Pin Name List 699
39.4 Product Dependencies 699
39.5 Functional Description 700
39.6 Timer Counter (TC) User Interface 713

40 Controller Area Network (CAN) 731

40.1 Overview 731
40.2 Block Diagram 732
40.3 Application Block Diagram 732
40.4 I/O Lines Description 733
40.5 Product Dependencies 733
40.6 CAN Controller Features 734
40.7 Functional Description 747
40.8 Controller Area Network (CAN) User Interface 759

41 Pulse Width Modulation Controller (PWM) 791

41.1 Description 791
41.2 Block Diagram 791
41.3 I/O Lines Description 792

41.4 Product Dependencies 792
41.5 Functional Description 793
41.6 Pulse Width Modulation Controller (PWM) User Interface 801

42 MultiMedia Card Interface (MCI) 815

42.1 Description 815
42.2 Block Diagram 816
42.3 Application Block Diagram 816
42.4 Pin Name List 817
42.5 Product Dependencies 817
42.6 Bus Topology 818
42.7 MultiMedia Card Operations 819
42.8 SD/SDIO Card Operations 828
42.9 MultiMedia Card Interface (MCI) User Interface 829

43 10/100 Ethernet MAC (EMAC) 847

43.1 Description 847
43.2 Block Diagram 847
43.3 Functional Description 848
43.4 Programming Interface 860
43.5 Ethernet MAC 10/100 (EMAC) User Interface 863

44 USB Host Port (UHP) 901

44.1 Description 901
44.2 Block Diagram 901
44.3 Product Dependencies 902
44.4 Functional Description 902
44.5 Typical Connection 904

45 USB High Speed Device Port (UDPHS) 905

45.1 Description 905
45.2 Block Diagram 906
45.3 Typical Connection 907
45.4 Functional Description 908
45.5 USB High Speed Device Port (UDPHS) User Interface 932

46 LCD Controller (LCDC) 973

46.1 Description 973
46.2 Block Diagram 974

46.3 I/O Lines Description 975
46.4 Product Dependencies 975
46.5 Functional Description 977
46.6 Interrupts 997
46.7 Configuration Sequence 997
46.8 Double-buffer Technique 998
46.9 2D Memory Addressing 999
46.10 Register Configuration Guide 1001
46.11 LCD Controller (LCDC) User Interface 1002

47 Image Sensor Interface (ISI) 1031

47.1 Overview 1031
47.2 Block Diagram 1032
47.3 Functional Description 1032
47.4 Image Sensor Interface (ISI) User Interface 1041

48 Analog-to-Digital Converter (ADC) 1061

48.1 Description 1061
48.2 Block Diagram 1061
48.3 Signal Description 1062
48.4 Product Dependencies 1062
48.5 Functional Description 1063
48.6 Analog-to-Digital Converter (ADC) User Interface 1068

49 AT91CAP9 Electrical Characteristics 1081

49.1 Absolute Maximum Ratings 1081
49.2 DC Characteristics 1082
49.3 Power Consumption 1083
49.4 32 kHz Crystal Oscillator Characteristics 1085
49.5 12 MHz Main Oscillator Characteristics 1086
49.6 PLLA and PLLB Characteristics 1087
49.7 UTMI PLL Characteristics 1087
49.8 USB HS Characteristics 1088
49.9 ADC 1090
49.10 Core Power Supply POR Characteristics 1091
49.11 Timings 1093

50 AT91CAP9S Mechanical Characteristics 1113

50.1 Package Drawing 1113

50.2 Soldering Profile 1116

51 AT91CAP9S Ordering Information 1117

52 AT91CAP9S Errata 1119

52.1 Marking 1119

53 Revision History 1121

Table of Contents 1

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
Tel: (852) 2245-6100
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
www.atmel.com/AT91CAP

Technical Support
Atmel techincal support

Sales Contacts
www.atmel.com/contacts/

Literature Requests
www.atmel.com/literature