Features

- Incorporates the ARM926EJ-S™ ARM® Thumb® Processor
  - DSP Instruction Extensions, Jazelle® Technology for Java® Acceleration
  - 16 Kbyte Data Cache, 16 Kbyte Instruction Cache, Write Buffer
  - 220 MIPS at 200 MHz
  - Memory Management Unit
  - EmbeddedICE™, Debug Communication Channel Support
  - Mid-level Implementation Embedded Trace Macrocell™
- Bus Matrix
  - Nine 32-bit-layer Matrix, Allowing a Total of 28.8 Gbps of On-chip Bus Bandwidth
  - Boot Mode Select Option, Remap Command
- Embedded Memories
  - One 128 Kbyte Internal ROM, Single-cycle Access at Maximum Bus Matrix Speed
  - One 80 Kbyte Internal SRAM, Single-cycle Access at Maximum Processor or Bus Matrix Speed
  - One 16 Kbyte Internal SRAM, Single-cycle Access at Maximum Bus Matrix Speed
- Dual External Bus Interface (EBI0 and EBI1)
  - EBI0 Supports SDRAM, Static Memory, ECC-enabled NAND Flash and CompactFlash®
  - EBI1 Supports SDRAM, Static Memory and ECC-enabled NAND Flash
- DMA Controller (DMAC)
  - Acts as one Bus Matrix Master
  - Embeds 2 Unidirectional Channels with Programmable Priority, Address Generation, Channel Buffering and Control
- Twenty Peripheral DMA Controller Channels (PDC)
- 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 Virtual Screen Buffers
- Two D Graphics Accelerator
  - Line Draw, Block Transfer, Clipping, Commands Queuing
- 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) Host Double Port
  - Dual On-chip Transceivers
  - Integrated FIFOs and Dedicated DMA Channels
- USB 2.0 Full Speed (12 Mbits per second) Device Port
  - On-chip Transceiver, 2,432-byte Configurable Integrated DPRAM
- Ethernet MAC 10/100 Base-T
  - Media Independent Interface or Reduced Media Independent Interface
  - 28-byte FIFOs and Dedicated DMA Channels for Receive and Transmit
- Fully-featured System Controller, including
  - Reset Controller, Shutdown Controller
  - Twenty 32-bit Battery Backup Registers for a Total of 80 Bytes
  - Clock Generator and Power Management Controller
  - Advanced Interrupt Controller and Debug Unit

AT91 ARM Thumb Microcontrollers

AT91SAM9263

Preliminary

6249H–ATARM–27-Jul-09

  - Periodic Interval Timer, Watchdog Timer and Double Real-time Timer
- Reset Controller (RSTC)
  - Based on Two Power-on Reset Cells, Reset Source Identification and Reset Output Control
- Shutdown Controller (SHDWC)
  - Programmable Shutdown Pin Control and Wake-up Circuitry
- Clock Generator (CKGR)
  - 32768Hz Low-power Oscillator on Battery Backup Power Supply, Providing a Permanent Slow Clock
  - 3 to 20 MHz On-chip Oscillator and Two Up to 240 MHz PLLs
- 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
  - Mode for General Purpose Two-wire UART Serial Communication
- 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
- Two Real-time Timers (RTT)
  - 32-bit Free-running Backup Counter Running at Slow Clock with 16-bit Prescaler
- Five 32-bit Parallel Input/Output Controllers (PIOA, PIOB, PIOC, PIOD and PIOE)
  - 160 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
- One Part 2.0A and Part 2.0B-compliant CAN Controller
  - 16 Fully-programmable Message Object Mailboxes, 16-bit Time Stamp Counter
- Two Multimedia Card Interface (MCI)
  - SDCard/SDIO and MultiMediaCard™ Compliant
  - Automatic Protocol Control and Fast Automatic Data Transfers with PDC
  - Two SDCard Slots Support on eAch Controller
- Two Synchronous Serial Controllers (SSC)
  - Independent Clock and Frame Sync Signals for Each Receiver and Transmitter
  - I²S 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
- 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 16-bit PWM Controller (PWMC)
- One Two-wire Interface (TWI)
  - Master Mode Support, All Two-wire Atmel® EEPROMs Supported

- **IEEE® 1149.1 JTAG Boundary Scan on All Digital Pins**
- **Required Power Supplies**
  - **1.08V to 1.32V for VDDCORE and VDDBU**
  - **3.0V to 3.6V for VDDOSC and VDDPLL**
  - **2.7V to 3.6V for VDDIOP0 (Peripheral I/Os)**
  - **1.65V to 3.6V for VDDIOP1 (Peripheral I/Os)**
  - **Programmable 1.65V to 1.95V or 3.0V to 3.6V for VDDIOM0/VDDIOM1 (Memory I/Os)**
- **Available in a 324-ball TFBGA Green Package**

1. Description

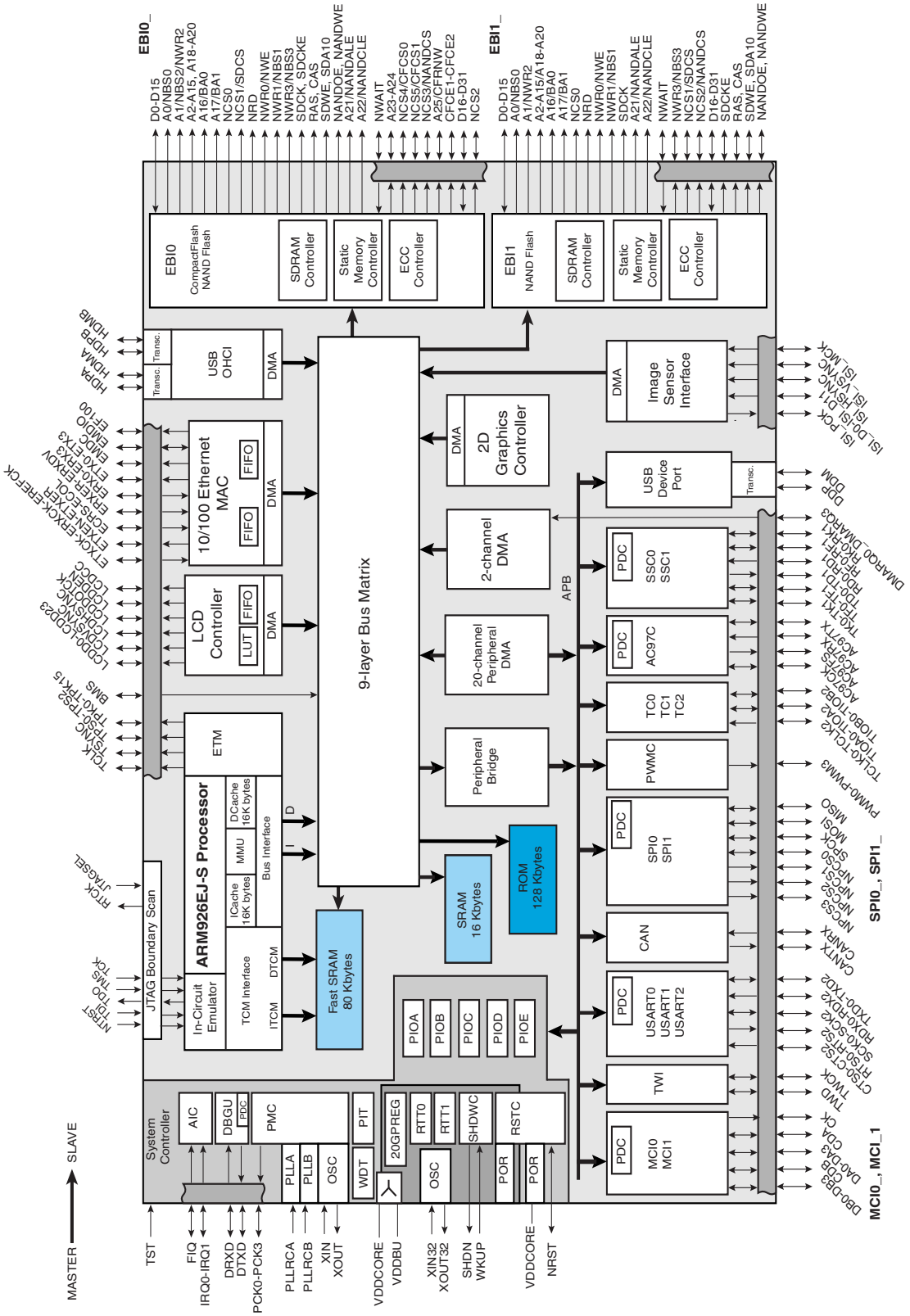
The AT91SAM9263 32-bit microcontroller, based on the ARM926EJ-S processor, is architectured on a 9-layer matrix, allowing a maximum internal bandwidth of nine 32-bit buses. It also features two independent external memory buses, EBI0 and EBI1, capable of interfacing with a wide range of memory devices and an IDE hard disk. Two external buses prevent bottlenecks, thus guaranteeing maximum performance.

The AT91SAM9263 embeds an LCD Controller supported by a Two D Graphics Controller and a 2-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.

When coupled with an external GPS engine, the AT91SAM9263 provides the ideal solution for navigation systems.

2. AT91SAM9263 Block Diagram

Figure 2-1. AT91SAM9263 Block Diagram



3. Signal Description

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

Table 3-1. Signal Description List

Signal Name	Function	Type	Active Level	Comments
Power Supplies				
VDDIOM0	EBI0 I/O Lines Power Supply	Power		1.65V to 3.6V
VDDIOM1	EBI1 I/O Lines Power Supply	Power		1.65V to 3.6V
VDDIOP0	Peripherals I/O Lines Power Supply	Power		2.7V to 3.6V
VDDIOP1	Peripherals I/O 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
VDDOSC	Oscillator 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		
GNDBU	Backup Ground	Ground		
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		Driven at 0V only. Do not tie over VDDBU.
WKUP	Wake-up Input	Input		Accepts between 0V and VDDBU.
ICE and JTAG				
NTRST	Test Reset Signal	Input	Low	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. Accepts between 0V and VDDBU.
RTCK	Return Test Clock	Output		

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
Embedded Trace Module - ETM				
TSYNC	Trace Synchronization Signal	Output		
TCLK	Trace Clock	Output		
TPS0 - TPS2	Trace ARM Pipeline Status	Output		
TPK0 - TPK15	Trace Packet Port	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		
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 - PIOE				
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
PE0 - PE31	Parallel IO Controller E	I/O		Pulled-up input at reset
Direct Memory Access Controller - DMA				
DMARQ0-DMARQ3	DMA Requests	Input		
External Bus Interface - EBI0 - EBI1				
EBIx_D0 - EBIx_D31	Data Bus	I/O		Pulled-up input at reset
EBIx_A0 - EBIx_A25	Address Bus	Output		0 at reset
EBIx_NWAIT	External Wait Signal	Input	Low	
Static Memory Controller - SMC				
EBI0_NCS0 - EBI0_NCS5, EBI1_NCS0 - EBI1_NCS2	Chip Select Lines	Output	Low	
EBIx_NWR0 -EBIx_NWR3	Write Signal	Output	Low	
EBIx_NRD	Read Signal	Output	Low	
EBIx_NWE	Write Enable	Output	Low	
EBIx_NBS0 - EBIx_NBS3	Byte Mask Signal	Output	Low	

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
CompactFlash Support				
EBI0_CFCE1 - EBI0_CFCE2	CompactFlash Chip Enable	Output	Low	
EBI0_CFOE	CompactFlash Output Enable	Output	Low	
EBI0_CFWE	CompactFlash Write Enable	Output	Low	
EBI0_CFIOR	CompactFlash IO Read	Output	Low	
EBI0_CFIOW	CompactFlash IO Write	Output	Low	
EBI0_CFRNW	CompactFlash Read Not Write	Output		
EBI0_CFCS0 - EBI0_CFCS1	CompactFlash Chip Select Lines	Output	Low	
NAND Flash Support				
EBIx_NANDCS	NAND Flash Chip Select	Output	Low	
EBIx_NANDOE	NAND Flash Output Enable	Output	Low	
EBIx_NANDWE	NAND Flash Write Enable	Output	Low	
SDRAM Controller				
EBIx_SDCK	SDRAM Clock	Output		
EBIx_SDCKE	SDRAM Clock Enable	Output	High	
EBIx_SDCS	SDRAM Controller Chip Select	Output	Low	
EBIx_BA0 - EBIx_BA1	Bank Select	Output		
EBIx_SDWE	SDRAM Write Enable	Output	Low	
EBIx_RAS - EBIx_CAS	Row and Column Signal	Output	Low	
EBIx_SDA10	SDRAM Address 10 Line	Output		
Multimedia Card Interface				
MCIx_CK	Multimedia Card Clock	Output		
MCIx_CDA	Multimedia Card Slot A Command	I/O		
MCIx_CDB	Multimedia Card Slot B Command	I/O		
MCIx_DA0 - MCIx_DA3	Multimedia Card Slot A Data	I/O		
MCIx_DB0 - MCIx_DB3	Multimedia Card Slot B Data	I/O		
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		

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
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				
PWMx	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				
TWD	Two-wire Serial Data	I/O		
TWCK	Two-wire Serial Clock	I/O		
CAN Controllers				
CANRX	CAN Input	Input		
CANTX	CAN Output	Output		
LCD Controller - LCDC				
LCDD0 - LCDD23	LCD Data Bus	Output		
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		

Table 3-1. Signal Description List (Continued)

Signal Name	Function	Type	Active Level	Comments
Ethernet 10/100				
ETXCK	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 Device Port				
DDM	USB Device Port Data -	Analog		
DDP	USB Device Port Data +	Analog		
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		
Image Sensor Interface - ISI				
ISI_D0-ISI_D11	Image Sensor Data	Input		
ISI_MCK	Image Sensor Reference Clock	Output		Provided by PCK3
ISI_HSYNC	Image Sensor Horizontal Synchro	Input		
ISI_VSYNC	Image Sensor Vertical Synchro	Input		
ISI_PCK	Image Sensor Data Clock	Input		

4. Package and Pinout

The AT91SAM9263 is available in a 324-ball TFBGA Green package, 15 x 15 mm, 0.8mm ball pitch.

4.1 324-ball TFBGA Package Outline

Figure 4-1 shows the orientation of the 324-ball TFBGA package.

A detailed mechanical description is given in the section “AT91SAM9263 Mechanical Characteristics” in the product datasheet.

Figure 4-1. 324-ball TFBGA Pinout (Top View)

4.2 324-ball TFBGA Package Pinout

Table 4-1. AT91SAM9263 Pinout for 324-ball TFBGA Package

Pin	Signal Name
A1	EBI0_D2
A2	EBI0_SDCKE
A3	EBI0_NWE_NWR0
A4	EBI0_NCS1_SDCS
A5	EBI0_A19
A6	EBI0_A11
A7	EBI0_A10
A8	EBI0_A5
A9	EBI0_A1_NBS2_NWR2
A10	PD4
A11	PC30
A12	PC26
A13	PC24
A14	PC19
A15	PC12
A16	VDDCORE
A17	VDDIOP0
A18	DDP
B1	EBI0_D4
B2	EBI0_NANDOE
B3	EBI0_CAS
B4	EBI0_RAS
B5	EBI0_NBS3_NWR3
B6	EBI0_A22
B7	EBI0_A15
B8	EBI0_A7
B9	EBI0_A4
B10	PD0
B11	PC28
B12	PC21
B13	PC17
B14	PC9
B15	PC7
B16	PC5
B17	PB16
B18	DDM
C1	EBI0_D6
C2	EBI0_D0
C3	EBI0_NANDWE
C4	EBI0_SDWE
C5	EBI0_SDCK
C6	EBI0_A21
C7	EBI0_A13
C8	EBI0_A8
C9	EBI0_A3
C10	PD2
C11	PC29
C12	PC23
C13	PC14
C14	PC8
E10	PC31
E11	PC22
E12	PC15
E13	PC11
E14	PC4
E15	PB30
E16	PC0
E17	PB31
E18	HDPA
F1	PD7
F2	EBI0_D13
F3	EBI0_D9
F4	EBI0_D11
F5	EBI0_D12
F6	EBI0_NCS0
F7	EBI0_A16_BA0
F8	EBI0_A12
F9	EBI0_A6
F10	PD3
F11	PC27
F12	PC18
F13	PC13
F14	PB26
F15	PB25
F16	PB29
F17	PB27
F18	HDMA
G1	PD17
G2	PD12
G3	PD6
G4	EBI0_D14
G5	PD5
G6	PD8
G7	PD10
G8	GND
G9	NC[1]
G10	GND
G11	GND
G12	GND
G13	PB21
G14	PB20
G15	PB23
G16	PB28
G17	PB22
G18	PB18
H1	PD24
H2	PD13
H3	PD15
H4	PD9
H5	PD11
K1	PE6
K2	PD28
K3	PE0
K4	PE1
K5	PD27
K6	PD31
K7	PD29
K8	PD25
K9	GND
K10	VDDIOM0
K11	GND
K12	VDDIOM0
K13	PB3/BMS
K14	PA14
K15	PA15
K16	PB1
K17	PB0
K18	PB2
L1	PE10
L2	PE4
L3	PE9
L4	PE7
L5	PE5
L6	PE2
L7	PE3
L8	VDDIOP1
L9	VDDIOM1
L10	VDDIOM0
L11	VDDIOP0
L12	GNDBU
L13	PA13
L14	PB4
L15	PA9
L16	PA12
L17	PA10
L18	PA11
M1	PE18
M2	PE14
M3	PE15
M4	PE11
M5	PE13
M6	PE12
M7	PE8
M8	VDDBU
M9	EBI1_A21
M10	VDDIOM1
M11	GND
M12	GND
M13	VDDIOM1
M14	PA6
P10	EBI1_NCS0
P11	EBI1_NWE_NWR0
P12	EBI1_D4
P13	EBI1_D10
P14	PA3
P15	PA2
P16	PE28
P17	TDI
P18	PLLRCB
R1	XOUT32
R2	TST
R3	PA18
R4	PA25
R5	PA30
R6	EBI1_A2
R7	EBI1_A14
R8	EBI1_A13
R9	EBI1_A17_BA1
R10	EBI1_D1
R11	EBI1_D8
R12	EBI1_D12
R13	EBI1_D15
R14	PE26
R15	EBI1_SDCK
R16	PE30
R17	TCK
R18	XOUT
T1	VDDOSC
T2	VDDIOM1
T3	PA19
T4	PA21
T5	PA26
T6	PA31
T7	EBI1_A7
T8	EBI1_A12
T9	EBI1_A18
T10	EBI1_D0
T11	EBI1_D7
T12	EBI1_D14
T13	PE23
T14	PE25
T15	PE29
T16	PE31
T17	GNDPLL
T18	XIN
U1	PA17
U2	PA20
U3	PA23
U4	PA24
U5	PA28

Table 4-1. AT91SAM9263 Pinout for 324-ball TFBGA Package (Continued)

Pin	Signal Name
C15	PC3
C16	GND
C17	VDDIOP0
C18	HDPB
D1	EBI0_D10
D2	EBI0_D3
D3	NC(1)
D4	EBI0_D1
D5	EBI0_A20
D6	EBI0_A17_BA1
D7	EBI0_A18
D8	EBI0_A9
D9	EBI0_A2
D10	PD1
D11	PC25
D12	PC20
D13	PC6
D14	PC16
D15	PC10
D16	PC2
D17	PC1
D18	HDMB
E1	EBI0_D15
E2	EBI0_D7
E3	EBI0_D5
E4	EBI0_D8
E5	EBI0_NBS1_NWR1
E6	EBI0_NRD
E7	EBI0_A14
E8	EBI0_SDA10
E9	EBI0_A0_NBS0

Pin	Signal Name
H6	PD14
H7	PD16
H8	VDDIOM0
H9	GND
H10	VDDCORE
H11	GND
H12	PB19
H13	PB17
H14	PB15
H15	PB13
H16	PB24
H17	PB14
H18	PB12
J1	PD30
J2	PD26
J3	PD22
J4	PD19
J5	PD18
J6	PD23
J7	PD21
J8	PD20
J9	GND
J10	GND
J11	GND
J12	PB11
J13	PB9
J14	PB10
J15	PB5
J16	PB6
J17	PB7
J18	PB8

Pin	Signal Name
M15	PA4
M16	PA7
M17	PA5
M18	PA8
N1	NC
N2	NC
N3	PE19
N4	NC(1)
N5	PE17
N6	PE16
N7	EBI1_A6
N8	EBI1_A11
N9	EBI1_A22
N10	EBI1_D2
N11	EBI1_D6
N12	EBI1_D9
N13	GND
N14	GNDPLL
N15	PA1
N16	PA0
N17	TMS
N18	TDO
P1	XIN32
P2	SHDN
P3	PA16
P4	WKUP
P5	JTAGSEL
P6	PE20
P7	EBI1_A8
P8	EBI1_A4
P9	EBI1_A19

Pin	Signal Name
U6	EBI1_A0_NBS0
U7	EBI1_A5
U8	EBI1_A10
U9	EBI1_A16_BA0
U10	EBI1_NRD
U11	EBI1_D3
U12	EBI1_D13
U13	PE22
U14	PE27
U15	RTCK
U16	NTRST
U17	VDDPLLA
U18	PLLRCA
V1	VDDCORE
V2	PA22
V3	PA27
V4	PA29
V5	EBI1_A1_NWR2
V6	EBI1_A3
V7	EBI1_A9
V8	EBI1_A15
V9	EBI1_A20
V10	EBI1_NBS1_NWR1
V11	EBI1_D5
V12	EBI1_D11
V13	PE21
V14	PE24
V15	NRST
V16	GND
V17	GND
V18	VDDPLLB

Note: 1. NC pins must be left unconnected.

5. Power Considerations

5.1 Power Supplies

AT91SAM9263 has several types of power supply pins:

- VDDCORE pins: Power the core, including the processor, the embedded memories and the peripherals; voltage ranges from 1.08V to 1.32V, 1.2V nominal.
- VDDIOM0 and VDDIOM1 pins: Power the External Bus Interface 0 I/O lines and the External Bus Interface 1 I/O lines, respectively; 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 Peripheral I/O lines and the USB transceivers; voltage ranges from 2.7V to 3.6V, 3.3V nominal.
- VDDIOP1 pins: Power the Peripheral 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.
- VDDBU pin: Powers the Slow Clock oscillator and a part of the System Controller; voltage ranges from 1.08V to 1.32V, 1.2V nominal.
- VDDPLL pin: Powers the PLL cells; voltage ranges from 3.0V to 3.6V, 3.3V nominal.
- VDDOSC pin: Powers the Main Oscillator cells; voltage ranges from 3.0V to 3.6V, L3.3V nominal.

The power supplies VDDIOM0, VDDIOM1 and VDDIOP0, 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 GND are common to VDDOSC, VDDCORE, VDDIOM0, VDDIOM1, VDDIOP0 and VDDIOP1 pins power supplies. Separated ground pins are provided for VDDBU and VDDPLL. These ground pins are respectively GNDBU and GNDPLL.

5.2 Power Consumption

The AT91SAM9263 consumes about 700 µA (worst case) of static current on VDDCORE at 25°C. This static current rises at up to 7 mA if the temperature increases to 85°C.

On VDDBU, the current does not exceed 3 µA @25°C, but can rise at up to 20 µA @85°C. An automatic switch to VDDCORE guarantees low power consumption on the battery when the system is on.

For dynamic power consumption, the AT91SAM9263 consumes a maximum of 70 mA on VDDCORE at maximum conditions (1.2V, 25°C, processor running full-performance algorithm).

5.3 Programmable I/O Lines Power Supplies

The power supply pins VDDIOM0 and VDDIOM1 accept two voltage ranges. This allows the device to reach its maximum speed, either out of 1.8V or 3.0V external memories.

The maximum speed is 100 MHz on the pin SDCK (SDRAM Clock) loaded with 10 pF. The other signals (control, address and data signals) do not go over 50 MHz, loaded with 30 pF for power supply at 1.8V and 50 pF for power supply at 3.3V.

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. However, the device cannot reach its maximum speed if the voltage supplied to

the pins is only 1.8V without reprogramming the EBI0 voltage range. The user must be sure to program the EBI0 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 (VDDBU). 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.

All JTAG signals except JTAGSEL (VDDBU) 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, which manage the processor and the JTAG reset, the NRST and NTRST pins can be left unconnected.

The NRST and NTRST pins both integrate a permanent pull-up resistor of 100 kΩ minimum to VDDIOP0.

The NRST signal is inserted in the Boundary Scan.

6.4 PIO Controllers

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

After reset, all the I/O lines default as inputs with pull-up resistors enabled, except those which are multiplexed with the External Bus Interface signals that require to be enabled as Peripheral at reset. This is explicitly indicated in the column “Reset State” of the PIO Controller multiplexing tables on page 36 and following.

6.5 Shutdown Logic Pins

The SHDN pin is a tri-state output only pin, which is driven by the Shutdown Controller. There is no internal pull-up. An external pull-up to VDDBU is needed and its value must be higher than 1

MΩ. The resistor value is calculated according to the regulator enable implementation and the SHDN level.

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 Harvard 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

- 9-layer Matrix, handling requests from 9 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 AT91SAM9263 manages nine masters, thus each master can perform an access concurrently with others to an available slave peripheral or memory.

Each master has its own decoder, which is defined specifically for each master.

Table 7-1. List of Bus Matrix Masters

Master 0	OHCI USB Host Controller
Master 1	Image Sensor Interface
Master 2	Two D Graphic Controller
Master 3	DMA Controller
Master 4	Ethernet MAC
Master 5	LCD Controller
Master 6	Peripheral DMA Controller
Master 7	ARM926 Data
Master 8	ARM926™ Instruction

7.4 Matrix Slaves

The Bus Matrix of the AT91SAM9263 manages eight slaves. Each slave has its own arbiter, thus allowing to program a different arbitration per slave.

The LCD Controller, the DMA Controller, the USB OTG and the USB Host have a user interface mapped as a slave on 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 ROM
Slave 1	Internal 80 Kbyte SRAM
Slave 2	Internal 16 Kbyte SRAM
Slave 3	LCD Controller User Interface
	DMA Controller User Interface
	USB Host User Interface
Slave 4	External Bus Interface 0
Slave 5	External Bus Interface 1
Slave 6	Peripheral Bridge

7.5 Master to Slave Access

In most cases, all the masters can access all the slaves. However, some paths do not make sense, for example, allowing access from the Ethernet MAC to the Internal Peripherals. Thus, these paths are forbidden or simply not wired, and are shown as "-" in Table 7-3.

Table 7-3. Masters to Slaves Access

Master		0	1	2	3	4	5	6	7&8
Slave		OHCI USB Host Controller	Image Sensor Interface	Two D Graphics Controller	DMA Controller	Ethernet MAC	LCD Controller	Peripheral DMA Controller	ARM926 Data & Instruction
0	Internal ROM	X	X	X	X	X	X	X	X
1	Internal 80 Kbyte SRAM	X	X	X	X	X	X	X	X
2	Internal 16 Kbyte SRAM Bank	X	X	X	X	X	X	X	X
3	LCD Controller User Interface	-	-	-	-	-	-	-	X
	DMA Controller User Interface	-	-	-	-	-	-	-	X
	USB Host User Interface	-	-	-	-	-	-	-	X
4	External Bus Interface 0	X	X	X	X	X	X	X	X
5	External Bus Interface 1	X	X	X	X	X	X	X	X
6	Peripheral Bridge	-	-	-	X	-	-	X	X

7.6 Peripheral DMA Controller

- Acts as one Matrix Master
- Allows data transfers between a peripheral and memory without any intervention of the processor
- Next Pointer support, removes heavy real-time constraints on buffer management.
- Twenty channels
  - Two for each USART
  - Two for the Debug Unit
  - Two for each Serial Synchronous Controller
  - Two for each Serial Peripheral Interface
  - Two for the AC97 Controller
  - One for 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
- 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

- Acts as one Matrix Master
- Embeds 2 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
  - Two 8-word FIFOs
  - 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

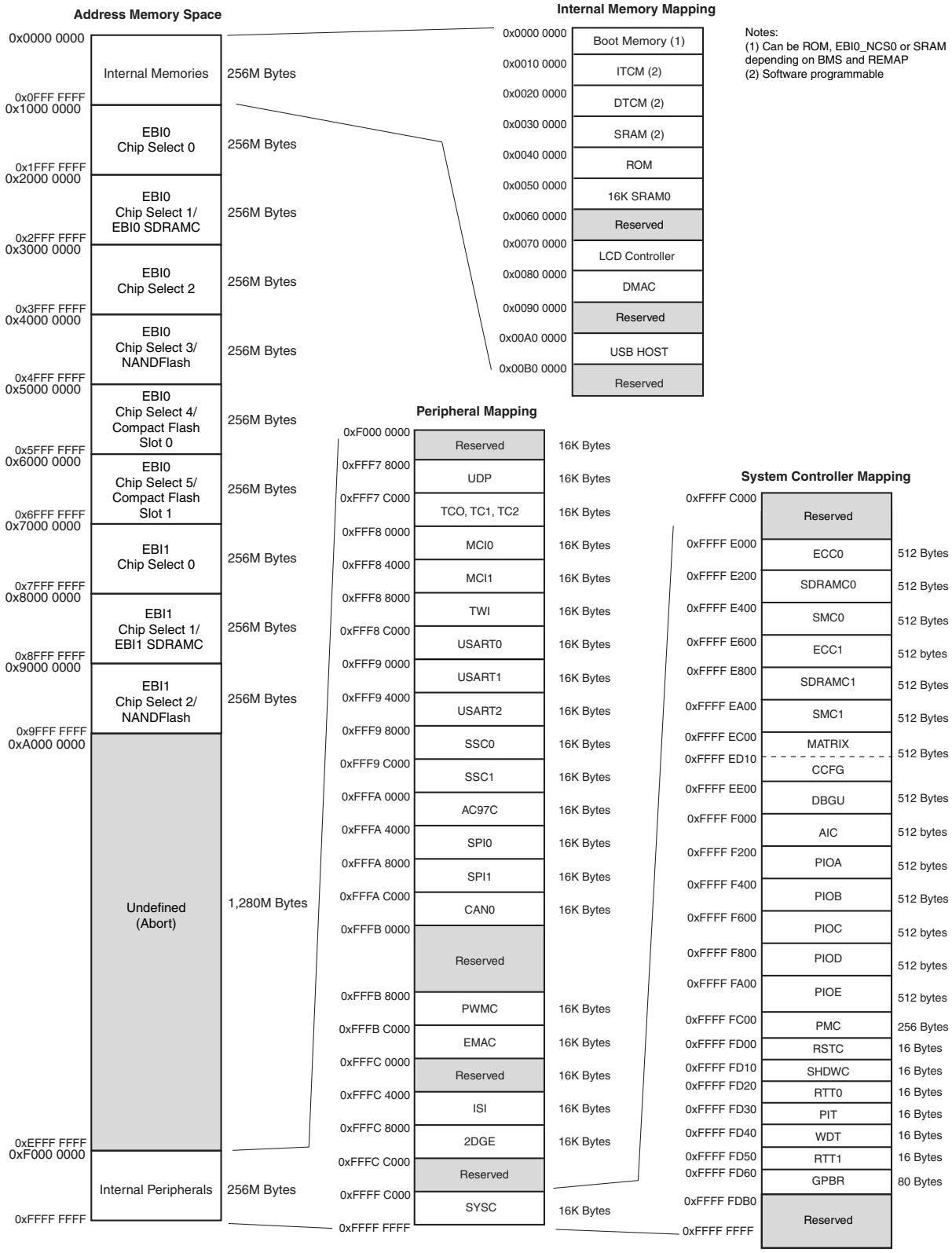
  - Suspend DMA operation
  - Programmable DMA lock transfer support.
- Transfer Initiation
  - Supports four external DMA Requests
  - Support for software handshaking interface. Memory mapped registers can be used to control the flow of a DMA transfer in place of a hardware handshaking interface
- Interrupt
  - 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
- Embedded Trace Macrocell: ETM9™
  - Medium+ Level Implementation
  - Half-rate Clock Mode
  - Four Pairs of Address Comparators
  - Two Data Comparators
  - Eight Memory Map Decoder Inputs
  - Two 16-bit Counters
  - One 3-stage Sequencer
  - One 45-byte FIFO
- IEEE1149.1 JTAG Boundary-scan on All Digital Pins

8. Memories

Figure 8-1. AT91SAM9263 Memory Mapping



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

Decoding breaks up the 4G bytes of address space into 16 banks of 256M bytes. The banks 1 to 9 are directed to the EBI0 that associates these banks to the external chip selects EBI0_NCS0 to EBI0_NCS5 and EBI1_NCS0 to EBI1_NCS2. The bank 0 is reserved for the addressing of the internal memories, and a second level of decoding provides 1M bytes of internal memory area. Bank 15 is reserved for the peripherals and provides access to the Advanced Peripheral Bus (APB).

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

Each master has its own bus and its own decoder, thus allowing a different memory mapping for each 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 22 for details.

A complete memory map is presented in Figure 8-1 on page 21.

8.1 Embedded Memories

- 128 Kbyte ROM
  - Single Cycle Access at full matrix speed
- One 80 Kbyte Fast SRAM
  - Single Cycle Access at full matrix speed
  - Supports ARM926EJ-S TCM interface at full processor speed
  - Allows internal Frame Buffer for up to 1/4 VGA 8 bpp screen
- 16 Kbyte Fast SRAM
  - Single Cycle Access at full matrix speed

8.1.1 Internal Memory Mapping

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

Table 8-1. Internal Memory Mapping

Address	REMAP = 0		REMAP = 1
	BMS = 1	BMS = 0	
0x0000 0000	ROM	EBI0_NCS0	SRAM C

8.1.1.1 Internal 80 Kbyte Fast SRAM

The AT91SAM9263 device embeds a high-speed 80 Kbyte SRAM. This internal SRAM is split into three areas. Its memory mapping is presented in Figure 8-1 on page 21.

- Internal SRAM A is the ARM926EJ-S Instruction TCM. The user can map this SRAM block anywhere in the ARM926 instruction memory space using CP15 instructions and the TCR

configuration register located in the Chip Configuration User Interface. This SRAM block is also accessible by the ARM926 Data Master and by the AHB Masters through the AHB bus at address 0x0010 0000.

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

Within the 80 Kbytes of SRAM available, the amount of memory assigned to each block is software programmable as a multiple of 16 Kbytes as shown in Table 8-2. This table provides the size of the Internal SRAM C according to the size of the internal SRAM A and the internal SRAM B.

Table 8-2. Internal SRAM Block Size

Internal SRAM C		Internal SRAM A (ITCM) Size		
		0	16 Kbytes	32 Kbytes
Internal SRAM B (DTCM) size	0	80 Kbytes	64 Kbytes	48 Kbytes
	16 Kbytes	64 Kbytes	48 Kbytes	32 Kbytes
	32 Kbytes	48 Kbytes	32 Kbytes	16 Kbytes

Note that among the five 16 Kbyte blocks making up the Internal SRAM, one is permanently assigned to Internal SRAM C.

At reset, the whole memory (80 Kbytes) is assigned to Internal SRAM C.

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

Table 8-3 illustrates different configurations and the related 16 Kbyte blocks assignments (RB0 to RB4).

Table 8-3. 16 Kbyte Block Allocation

Decoded Area	Address	Configuration examples and related 16 Kbyte block assignments				
		ITCM = 0 Kbyte DTCM = 0 Kbyte AHB = 80 Kbytes (1)	ITCM = 32 Kbytes DTCM = 32 Kbytes AHB = 16 Kbytes	ITCM = 16 Kbytes DTCM = 32 Kbytes AHB = 32 Kbytes	ITCM = 32 Kbytes DTCM = 16 Kbytes AHB = 32 Kbytes	ITCM = 16 Kbytes DTCM = 16 Kbytes AHB = 48 Kbytes
Internal SRAM A (ITCM)	0x0010 0000		RB1	RB1	RB1	RB1
	0x0010 4000		RB0		RB0	
Internal SRAM B (DTCM)	0x0020 0000		RB3	RB3	RB3	RB3
	0x0020 4000		RB2	RB2		

Table 8-3. 16 Kbyte Block Allocation (Continued)

Decoded Area	Address	Configuration examples and related 16 Kbyte block assignments				
		ITCM = 0 Kbyte DTCM = 0 Kbyte AHB = 80 Kbytes (1)	**ITCM = 32 Kbytes DTCM = 32 Kbytes AHB = 16 Kbytes**	**ITCM = 16 Kbytes DTCM = 32 Kbytes AHB = 32 Kbytes**	**ITCM = 32 Kbytes DTCM = 16 Kbytes AHB = 32 Kbytes**	**ITCM = 16 Kbytes DTCM = 16 Kbytes AHB = 48 Kbytes**
Internal SRAM C (AHB)	0x0030 0000	RB4	RB4	RB4	RB4	RB4
	0x0030 4000	RB3		RB0	RB2	RB2
	0x0030 8000	RB2				RB0
	0x0030 C000	RB1				
	0x0031 0000	RB0				

Note: 1. Configuration after reset.

When accessed from the Bus Matrix, the internal 80 Kbytes of Fast SRAM is single cycle accessible at full matrix speed (MCK). When accessed from the processor's TCM Interface, they are also single cycle accessible at full processor speed.

8.1.1.2 Internal 16 Kbyte Fast SRAM

The AT91SAM9263 integrates a 16 Kbyte SRAM, mapped at address 0x0050 0000. This SRAM is single cycle accessible at full Bus Matrix speed.

8.1.2 Boot Strategies

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

REMAP allows the user to layout the internal SRAM bank to 0x0. This is done by software once the system has booted. Refer to the section "AT91SAM9263 Bus Matrix" in the product datasheet for more details.

When REMAP = 0, BMS allows the user to layout at address 0x0 either the ROM or an external memory. This is done via hardware at reset.

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 21.

The AT91SAM9263 Bus Matrix manages a boot memory that depends on the level on the pin BMS at reset. The internal memory area mapped between address 0x0 and 0x000F FFFF is reserved to this effect.

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

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

8.1.2.1 BMS = 1, Boot on Embedded ROM

The system boots on Boot Program.

- Boot at slow clock
- 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

- SD Card
- NAND Flash
- SPI DataFlash® and Serial Flash connected on NPCS0 of the SPI0

- Interface with SAM-BA® Graphic User Interface to enable code loading via:
  - Serial communication on a DBGU
  - USB Bulk Device Port

8.1.2.2 BMS = 0, Boot on External Memory

- Boot at slow clock
- 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.

The customer-programmed software must perform a complete configuration.

To speed up the boot sequence when booting at 32 kHz EBI0 CS0 (BMS=0) the user must:

1. Program the PMC (main oscillator enable or bypass mode).
2. Program and Start the PLL.
3. Reprogram the SMC setup, cycle, hold, mode timings registers for CS0 to adapt them to the new clock.
4. Switch the main clock to the new value.

8.2 External Memories

The external memories are accessed through the External Bus Interfaces 0 and 1. Each Chip Select line has a 256 Mbyte memory area assigned.

Refer to Figure 8-1 on page 21.

8.2.1 External Bus Interfaces

The AT91SAM9263 features two External Bus Interfaces to offer more bandwidth to the system and to prevent bottlenecks while accessing external memories.

8.2.1.1 External Bus Interface 0

- Integrates three External Memory Controllers:
  - Static Memory Controller
  - SDRAM Controller
  - 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
  - SDRAM Controller or Static Memory Controller on NCS1
  - Static Memory Controller on NCS2
  - Static Memory Controller on NCS3, Optional NAND Flash support
  - Static Memory Controller on NCS4 - NCS5, Optional CompactFlash support
- Optimized for Application Memory Space

8.2.1.2 *External Bus Interface 1*

- Integrates three External Memory Controllers:
  - Static Memory Controller
  - SDRAM Controller
  - ECC Controller
- Additional logic for NAND Flash
- Optional Full 32-bit External Data Bus
- Up to 23-bit Address Bus (up to 8 Mbytes linear)
- Up to 3 Chip Selects, Configurable Assignment:
  - Static Memory Controller on NCS0
  - SDRAM Controller or Static Memory Controller on NCS1
  - Static Memory Controller on NCS2, Optional NAND Flash support
- Allows supporting an external Frame Buffer for the embedded LCD Controller without impacting processor performance.

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 SDRAM Controller

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

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

8.2.4 Error Corrected Code Controller

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

9. System Controller

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

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

- EBI0 and EBI1 chip select assignment and voltage range for external memories
- ARM Processor Tightly Coupled Memories

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

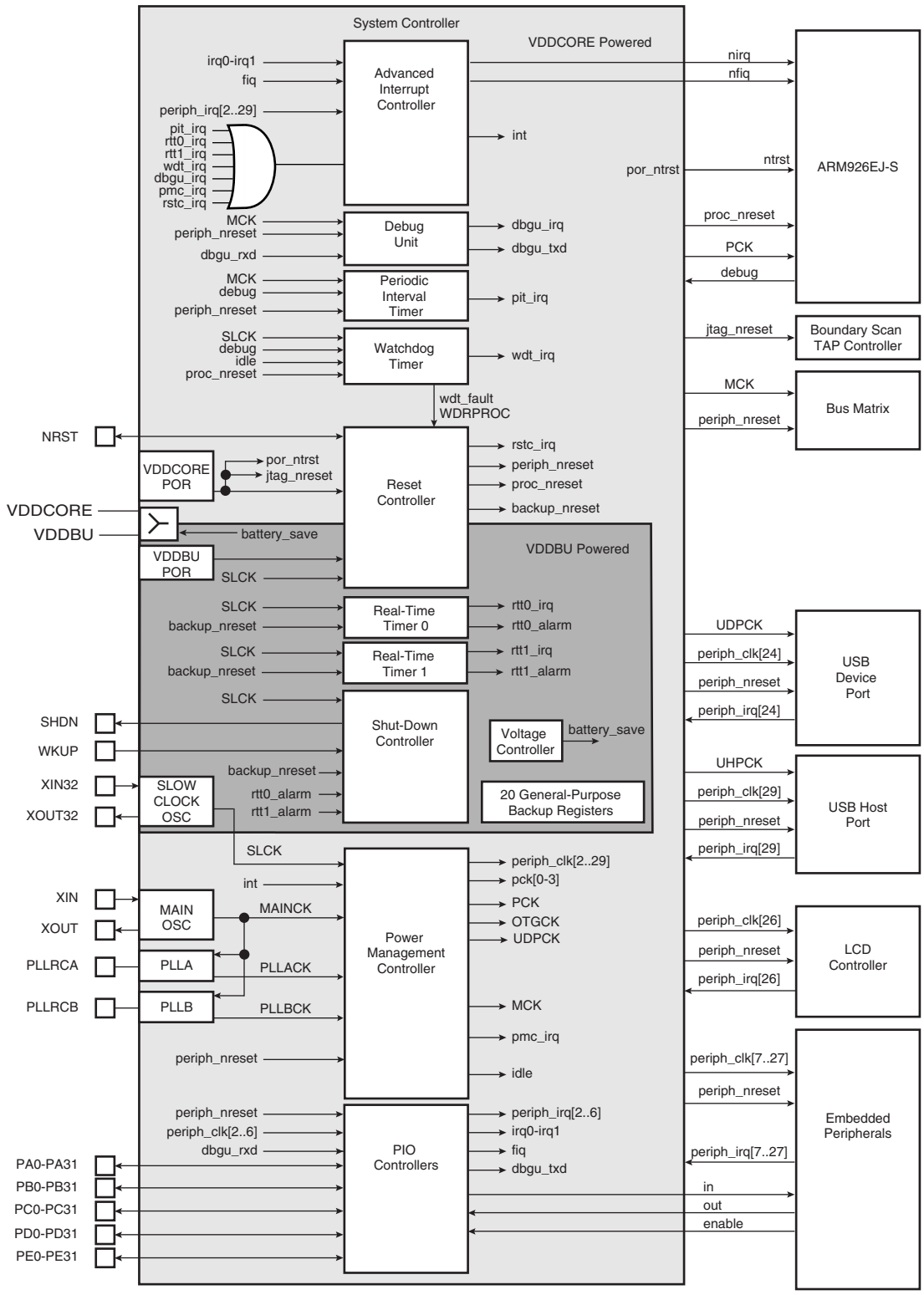
However, all the registers of the System Controller are mapped on the top of the address space. This allows all the registers of the System Controller to be addressed from a single pointer by using the standard ARM instruction set, as the Load/Store instructions have an indexing mode of ± 4 Kbytes.

Figure 9-1 on page 28 shows the System Controller block diagram.

Figure 8-1 on page 21 shows the mapping of the User Interfaces of the System Controller peripherals.

9.1 System Controller Block Diagram

Figure 9-1. AT91SAM9263 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

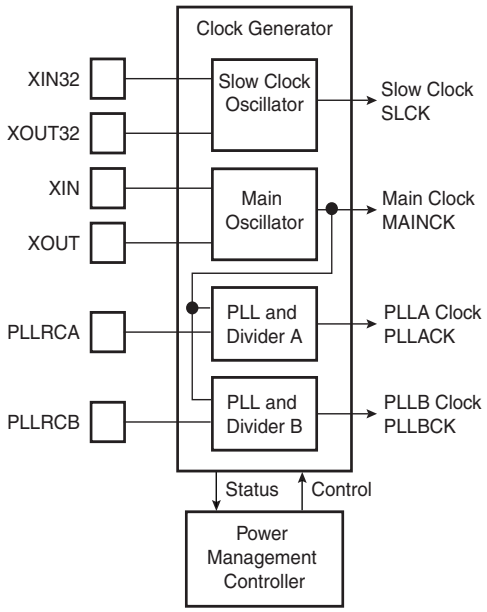
9.3 Shutdown Controller

- Shutdown and Wake-up logic
  - Software programmable assertion of the SHDN pin (SHDN is push-pull)
  - Deassertion programmable on a WKUP pin level change or on alarm

9.4 Clock Generator

- Embeds the low-power 32768 Hz Slow Clock Oscillator
  - Provides the permanent Slow Clock SLCK to the system
- Embeds the Main Oscillator
  - Oscillator bypass feature
  - Supports 3 to 20 MHz crystals
- Embeds 2 PLLs
  - Output 80 to 240 MHz clocks
  - Integrates an input divider to increase output accuracy
  - 1 MHz Minimum input frequency

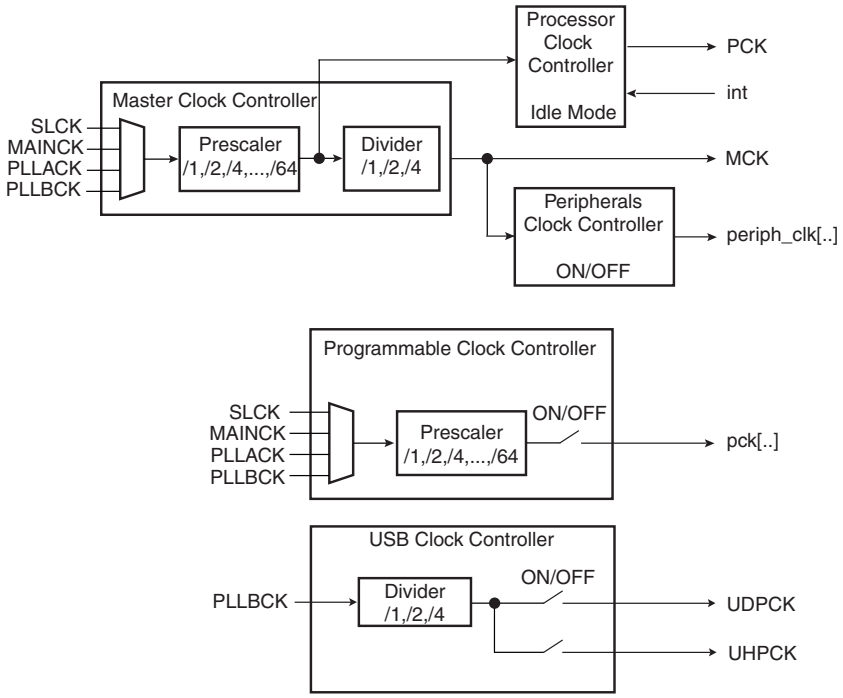
Figure 9-2. Clock Generator Block Diagram



9.5 Power Management Controller

- Provides:
  - the Processor Clock PCK
  - the Master Clock MCK, in particular to the Matrix and the memory interfaces
  - the USB Device Clock UDPCK
  - 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 with processor and peripherals running at a programmable frequency
  - Idle Mode with processor stopped while waiting for an interrupt
  - Slow Clock Mode with processor and peripherals running at low frequency
  - Standby Mode, mix of Idle and Backup Mode, with peripherals running at low frequency, processor stopped waiting for an interrupt
  - Backup Mode with Main Power Supplies off, VDDBU powered by a battery

Figure 9-3. AT91SAM9263 Power Management Controller Block Diagram



9.6 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®/WindowsCE® compliant tick generator

9.7 Watchdog Timer

- 16-bit key-protected Counter, programmable only once

- Windowed, prevents the processor deadlocking on the watchdog access

9.8 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 32.768Hz oscillator
  - Alarm Register capable of generating a wake-up of the system through the Shutdown Controller

9.9 General-purpose Backup Registers

- Twenty 32-bit general-purpose backup registers

9.10 Backup Power Switch

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

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

  - 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
  - Mode for general purpose Two-wire UART serial communication
- 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: 0x019607A0
- JTAG ID: 0x05B0C03F
- ARM926 TAP ID: 0x0792603F

9.14 PIO Controllers

- Five PIO Controllers, PIOA to PIOE, controlling a total of 160 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
  - PIOE 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 21.

10.2 Identifiers

Table 10-1 defines the Peripheral Identifiers. 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. AT91SAM9263 Peripheral Identifiers

Peripheral ID	Peripheral Mnemonic	Peripheral Name	External Interrupt
0	AIC	Advanced Interrupt Controller	FIQ
1	SYSC	System Controller Interrupt	
2	PIOA	Parallel I/O Controller A	
3	PIOB	Parallel I/O Controller B	
4	PIOC to PIOE	Parallel I/O Controller C, D and E	
5	reserved		
6	reserved		
7	US0	USART 0	
8	US1	USART 1	
9	US2	USART 2	
10	MCI0	Multimedia Card Interface 0	
11	MCI1	Multimedia Card Interface 1	
12	CAN	CAN Controller	
13	TWI	Two-Wire Interface	
14	SPI0	Serial Peripheral Interface 0	
15	SPI1	Serial Peripheral Interface 1	
16	SSC0	Synchronous Serial Controller 0	
17	SSC1	Synchronous Serial Controller 1	
18	AC97C	AC97 Controller	
19	TC0, TC1, TC2	Timer/Counter 0, 1 and 2	
20	PWMC	Pulse Width Modulation Controller	
21	EMAC	Ethernet MAC	
22	reserved		
23	2DGE	2D Graphic Engine	
24	UDP	USB Device Port	
25	ISI	Image Sensor Interface	
26	LCDC	LCD Controller	
27	DMA	DMA Controller	
28	reserved		

Table 10-1. AT91SAM9263 Peripheral Identifiers (Continued)

Peripheral ID	Peripheral Mnemonic	Peripheral Name	External Interrupt
29	UHP	USB Host Port	
30	AIC	Advanced Interrupt Controller	IRQ0
31	AIC	Advanced Interrupt Controller	IRQ1

Note: Setting AIC, SYSC, UHP and IRQ0 - 1 bits in the clock set/clear registers of the PMC has no effect.

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

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

10.2.1.2 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.

10.3 Peripherals Signals Multiplexing on I/O Lines

The AT91SAM9263 device features 5 PIO controllers, PIOA, PIOB, PIOC, PIOD and PIOE, which 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 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 specified, 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 specified 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 10-2. Multiplexing on PIO Controller A

PIO Controller A					Application Usage	
I/O Line	Peripheral A	Peripheral B	Reset State	Power Supply	Function	Comments
PA0	MCI0_DA0	SPI0_MISO	I/O	VDDIOP0		
PA1	MCI0_CDA	SPI0_MOSI	I/O	VDDIOP0		
PA2		SPI0_SPCK	I/O	VDDIOP0		
PA3	MCI0_DA1	SPI0_NPCS1	I/O	VDDIOP0		
PA4	MCI0_DA2	SPI0_NPCS2	I/O	VDDIOP0		
PA5	MCI0_DA3	SPI0_NPCS0	I/O	VDDIOP0		
PA6	MCI1_CK	PCK2	I/O	VDDIOP0		
PA7	MCI1_CDA		I/O	VDDIOP0		
PA8	MCI1_DA0		I/O	VDDIOP0		
PA9	MCI1_DA1		I/O	VDDIOP0		
PA10	MCI1_DA2		I/O	VDDIOP0		
PA11	MCI1_DA3		I/O	VDDIOP0		
PA12	MCI0_CK		I/O	VDDIOP0		
PA13	CANTX	PCK0	I/O	VDDIOP0		
PA14	CANRX	IRQ0	I/O	VDDIOP0		
PA15	TCLK2	IRQ1	I/O	VDDIOP0		
PA16	MCI0_CDB	EBI1_D16	I/O	VDDIOM1		
PA17	MCI0_DB0	EBI1_D17	I/O	VDDIOM1		
PA18	MCI0_DB1	EBI1_D18	I/O	VDDIOM1		
PA19	MCI0_DB2	EBI1_D19	I/O	VDDIOM1		
PA20	MCI0_DB3	EBI1_D20	I/O	VDDIOM1		
PA21	MCI1_CDB	EBI1_D21	I/O	VDDIOM1		
PA22	MCI1_DB0	EBI1_D22	I/O	VDDIOM1		
PA23	MCI1_DB1	EBI1_D23	I/O	VDDIOM1		
PA24	MCI1_DB2	EBI1_D24	I/O	VDDIOM1		
PA25	MCI1_DB3	EBI1_D25	I/O	VDDIOM1		
PA26	TXD0	EBI1_D26	I/O	VDDIOM1		
PA27	RXD0	EBI1_D27	I/O	VDDIOM1		
PA28	RTS0	EBI1_D28	I/O	VDDIOM1		
PA29	CTS0	EBI1_D29	I/O	VDDIOM1		
PA30	SCK0	EBI1_D30	I/O	VDDIOM1		
PA31	DMARQ0	EBI1_D31	I/O	VDDIOM1		

10.3.2 PIO Controller B Multiplexing

Table 10-3. Multiplexing on PIO Controller B

PIO Controller B					Application Usage	
I/O Line	Peripheral A	Peripheral B	Reset State	Power Supply	Function	Comments
PB0	AC97FS	TF0	I/O	VDDIOP0		
PB1	AC97CK	TK0	I/O	VDDIOP0		
PB2	AC97TX	TD0	I/O	VDDIOP0		
PB3	AC97RX	RD0	I/O	VDDIOP0		
PB4	TWD	RK0	I/O	VDDIOP0		
PB5	TWCK	RF0	I/O	VDDIOP0		
PB6	TF1	DMARQ1	I/O	VDDIOP0		
PB7	TK1	PWM0	I/O	VDDIOP0		
PB8	TD1	PWM1	I/O	VDDIOP0		
PB9	RD1	LCDCC	I/O	VDDIOP0		
PB10	RK1	PCK1	I/O	VDDIOP0		
PB11	RF1	SPI0_NPCS3	I/O	VDDIOP0		
PB12	SPI1_MISO		I/O	VDDIOP0		
PB13	SPI1_MOSI		I/O	VDDIOP0		
PB14	SPI1_SPCK		I/O	VDDIOP0		
PB15	SPI1_NPCS0		I/O	VDDIOP0		
PB16	SPI1_NPCS1	PCK1	I/O	VDDIOP0		
PB17	SPI1_NPCS2	TIOA2	I/O	VDDIOP0		
PB18	SPI1_NPCS3	TIOB2	I/O	VDDIOP0		
PB19			I/O	VDDIOP0		
PB20			I/O	VDDIOP0		
PB21			I/O	VDDIOP0		
PB22			I/O	VDDIOP0		
PB23			I/O	VDDIOP0		
PB24		DMARQ3	I/O	VDDIOP0		
PB25			I/O	VDDIOP0		
PB26			I/O	VDDIOP0		
PB27		PWM2	I/O	VDDIOP0		
PB28		TCLK0	I/O	VDDIOP0		
PB29		PWM3	I/O	VDDIOP0		
PB30			I/O	VDDIOP0		
PB31			I/O	VDDIOP0		

10.3.3 PIO Controller C Multiplexing

Table 10-4. Multiplexing on PIO Controller C

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

10.3.4 PIO Controller D Multiplexing

Table 10-5. Multiplexing on PIO Controller D

PIO Controller D					Application Usage	
I/O Line	Peripheral A	Peripheral B	Reset State	Power Supply	Function	Comments
PD0	TXD1	SPI0_NPCS2	I/O	VDDIOP0		
PD1	RXD1	SPI0_NPCS3	I/O	VDDIOP0		
PD2	TXD2	SPI1_NPCS2	I/O	VDDIOP0		
PD3	RXD2	SPI1_NPCS3	I/O	VDDIOP0		
PD4	FIQ	DMARQ2	I/O	VDDIOP0		
PD5	EBI0_NWAIT	RTS2	I/O	VDDIOM0		
PD6	EBI0_NCS4/CFCS0	CTS2	I/O	VDDIOM0		
PD7	EBI0_NCS5/CFCS1	RTS1	I/O	VDDIOM0		
PD8	EBI0_CFCE1	CTS1	I/O	VDDIOM0		
PD9	EBI0_CFCE2	SCK2	I/O	VDDIOM0		
PD10		SCK1	I/O	VDDIOM0		
PD11	EBI0_NCS2	TSYNC	I/O	VDDIOM0		
PD12	EBI0_A23	TCLK	A23	VDDIOM0		
PD13	EBI0_A24	TPS0	A24	VDDIOM0		
PD14	EBI0_A25_CFRNW	TPS1	A25	VDDIOM0		
PD15	EBI0_NCS3/NANDCS	TPS2	I/O	VDDIOM0		
PD16	EBI0_D16	TPK0	I/O	VDDIOM0		
PD17	EBI0_D17	TPK1	I/O	VDDIOM0		
PD18	EBI0_D18	TPK2	I/O	VDDIOM0		
PD19	EBI0_D19	TPK3	I/O	VDDIOM0		
PD20	EBI0_D20	TPK4	I/O	VDDIOM0		
PD21	EBI0_D21	TPK5	I/O	VDDIOM0		
PD22	EBI0_D22	TPK6	I/O	VDDIOM0		
PD23	EBI0_D23	TPK7	I/O	VDDIOM0		
PD24	EBI0_D24	TPK8	I/O	VDDIOM0		
PD25	EBI0_D25	TPK9	I/O	VDDIOM0		
PD26	EBI0_D26	TPK10	I/O	VDDIOM0		
PD27	EBI0_D27	TPK11	I/O	VDDIOM0		
PD28	EBI0_D28	TPK12	I/O	VDDIOM0		
PD29	EBI0_D29	TPK13	I/O	VDDIOM0		
PD30	EBI0_D30	TPK14	I/O	VDDIOM0		
PD31	EBI0_D31	TPK15	I/O	VDDIOM0		

10.3.5 PIO Controller E Multiplexing

Table 10-6. Multiplexing on PIO Controller E

PIO Controller E					Application Usage	
I/O Line	Peripheral A	Peripheral B	Reset State	Power Supply	Function	Comments
PE0	ISI_D0		I/O	VDDIOP1		
PE1	ISI_D1		I/O	VDDIOP1		
PE2	ISI_D2		I/O	VDDIOP1		
PE3	ISI_D3		I/O	VDDIOP1		
PE4	ISI_D4		I/O	VDDIOP1		
PE5	ISI_D5		I/O	VDDIOP1		
PE6	ISI_D6		I/O	VDDIOP1		
PE7	ISI_D7		I/O	VDDIOP1		
PE8	ISI_PCK	TIOA1	I/O	VDDIOP1		
PE9	ISI_HSYNC	TIOB1	I/O	VDDIOP1		
PE10	ISI_VSYNC	PWM3	I/O	VDDIOP1		
PE11		PCK3	I/O	VDDIOP1		
PE12		ISI_D8	I/O	VDDIOP1		
PE13		ISI_D9	I/O	VDDIOP1		
PE14		ISI_D10	I/O	VDDIOP1		
PE15		ISI_D11	I/O	VDDIOP1		
PE16			I/O	VDDIOP1		
PE17			I/O	VDDIOP1		
PE18		TIOA0	I/O	VDDIOP1		
PE19		TIOB0	I/O	VDDIOP1		
PE20		EBI1_NWAIT	I/O	VDDIOM1		
PE21	ETXCK	EBI1_NANDWE	I/O	VDDIOM1		
PE22	ECRS	EBI1_NCS2/NANDCS	I/O	VDDIOM1		
PE23	ETX0	EB1_NANDOE	I/O	VDDIOM1		
PE24	ETX1	EBI1_NWR3/NBS3	I/O	VDDIOM1		
PE25	ERX0	EBI1_NCS1/SDCS	I/O	VDDIOM1		
PE26	ERX1		I/O	VDDIOM1		
PE27	ERXER	EBI1_SDCKE	I/O	VDDIOM1		
PE28	ETXEN	EBI1_RAS	I/O	VDDIOM1		
PE29	EMDC	EBI1_CAS	I/O	VDDIOM1		
PE30	EMDIO	EBI1_SDWE	I/O	VDDIOM1		
PE31	EF100	EBI1_SDA10	I/O	VDDIOM1		

10.4 System Resource Multiplexing

10.4.1 LCD Controller

The LCD Controller can interface with several LCD panels. It supports 4 bits per pixel (bpp), 8 bpp or 16 bpp without limitation. Interfacing 24 bpp TFT panels prevents using the Ethernet MAC. 16 bpp TFT panels are interfaced through peripheral B functions, as color data is output on LCDD3 to LCDD7, LCDD11 to LCDD15 and LCDD19 to LCDD23. Intensity bit is output on LCDD10. Using the peripheral B does not prevent using MAC lines. 16 bpp STN panels are interfaced through peripheral A and color data is output on LCDD0 to LCDD15, thus MAC lines can be used on peripheral B.

Mapping the LCD signals on peripheral A and peripheral B makes is possible to use 24 bpp TFT panels in 24 bits (peripheral A) or 16 bits (peripheral B) by reprogramming the PIO controller and thus without hardware modification.

10.4.2 ETM™

Using the ETM prevents the use of the EBI0 in 32-bit mode. Only 16-bit mode (EBI0_D0 to EBI0_D15) is available, makes EBI0 unable to interface CompactFlash and NAND Flash cards, reduces EBI0's address bus width which makes it unable to address memory ranges bigger than 0x7FFFFF and finally it makes impossible to use EBI0_NCS2 and EBI0_NCS3.

10.4.3 EBI1

Using the following features prevents using EBI1 in 32-bit mode:

- the second slots of MCI0 and/or MCI1
- USART0
- DMA request 0 (DMARQ0)

10.4.4 Ethernet 10/100MAC

Using the following features of EBI1 prevent using Ethernet 10/100MAC:

- SDRAM
- NAND (unless NANDCS, NANDOE and NANDWE are managed by PIO)
- SMC 32 bits (SMC 16 bits is still available)
- NCS1, NCS2 are not available in SMC mode

10.4.5 SSC

Using SSC0 prevents using the AC97 Controller and Two-wire Interface.

Using SSC1 prevents using DMA Request 1, PWM0, PWM1, LCDCC and PCK1.

10.4.6 USART

Using USART2 prevents using EBI0's NWAIT signal, Chip Select 4 and CompactFlash Chip Enable 2.

Using USART1 prevents using EBI0's Chip Select 5 and CompactFlash Chip Enable1.

10.4.7 NAND Flash

Using the NAND Flash interface on EBI1 prevents using Ethernet MAC.

10.4.8 CompactFlash

Using the CompactFlash interface prevents using NCS4 and/or NCS5 to access other parallel devices.

10.4.9 SPI0 and MCI Interface

SPI0 signals and MCI0 signals are multiplexed, as the DataFlash Card is hardware-compatible with the SDCard. Only one can be used at a time.

10.4.10 Interrupts

Using IRQ0 prevents using the CAN controller.

Using FIQ prevents using DMA Request 2.

10.4.11 Image Sensor Interface

Using ISI in 8-bit data mode prevents using timers TIOA1, TIOB1.

10.4.12 Timers

Using TIOA2 and TIOB2, in this order, prevents using SPI1's Chip Selects [2-3].

10.5 Embedded Peripherals Overview

10.5.1 Serial Peripheral Interface

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

10.5.2 Two-wire Interface

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

10.5.3 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

10.5.4 Serial Synchronous 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

10.5.5 AC97 Controller

- Compatible with AC97 Component Specification V2.2
- Can 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 that can 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 (48 kHz and below)

10.5.6 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

10.5.7 Pulse Width Modulation Controller

- 4 channels, one 16-bit counter per channel
- Common clock generator, providing thirteen different clocks
  - Modulo n counter providing eleven clocks
  - Two independent Linear Dividers working on modulo n counter outputs
- Independent channel programming
  - Independent Enable Disable commands
  - Independent clock selection
  - Independent period and duty cycle, with double bufferization
  - Programmable selection of the output waveform polarity
  - Programmable center or left aligned output waveform

10.5.8 Multimedia Card Interface

- Two double-channel Multimedia Card Interfaces, allowing concurrent transfers with 2 cards
- Compatibility with MultiMediaCard Specification Version 3.31
- Compatibility with SD Memory Card Specification Version 1.0
- Compatibility with SDIO Specification Version V1.1
- Cards clock rate up to Master Clock divided by 2
- Embedded power management to slow down clock rate when not used
- Each MCI has two slots, each supporting
  - One slot for one MultiMediaCard bus (up to 30 cards) or
  - One SD Memory Card
- Support for stream, block and multi-block data read and write

10.5.9 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

10.5.10 USB Host Port

- Compliant with Open HCI Rev 1.0 Specification
- Compliant 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

10.5.11 USB Device Port

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

10.5.12 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

10.5.13 Two D Graphics Controller

- Acts as one Matrix Master
- Commands are passed through the APB User Interface
- Operates directly in the frame buffer of the LCD Controller
  - Line draw
  - Block transfer
  - Clipping
- Commands queuing through a FIFO

10.5.14 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
- DMA Interface, operating as a master on the Memory Controller
- 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

10.5.15 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

11. ARM926EJ-S Processor Overview

11.1 Overview

The ARM926EJ-S processor is a member of the ARM9s 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.

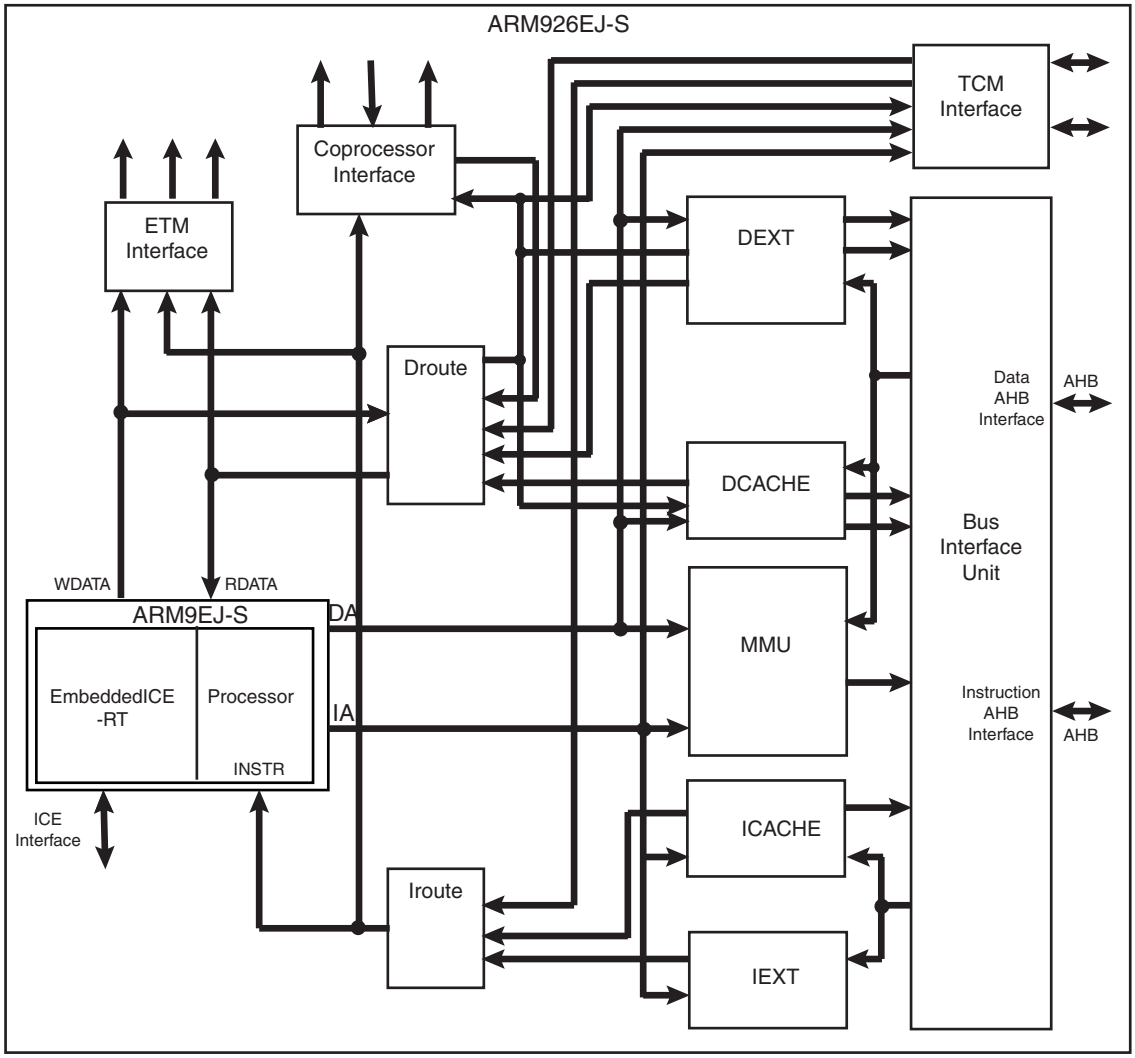
The ARM926EJ-S processor supports the ARM debug architecture and includes logic to assist in both hardware and software debug.

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

11.2 Block Diagram

Figure 11-1. ARM926EJ-S Internal Functional Block Diagram



11.3 ARM9EJ-S Processor

11.3.1 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.

11.3.2 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.

11.3.3 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.

11.3.4 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.

11.3.5 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.

11.3.6 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.

11.3.7 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 11-1 shows all the registers in all modes.

Table 11-1. ARM9TDMI® Modes and Registers Layout

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

Mode-specific banked registers

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

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

In privileged modes (FIQ, Supervisor, Abort, IRQ, Undefined), mode-specific banked registers (r8 to r14 in FIQ mode or r13 to r14 in the other modes) become available. The corresponding banked registers r14_fiq, r14_svc, r14_abt, r14_irq, r14_und are similarly used to hold the values (return address for each mode) of r15 (PC) when interrupts and exceptions arise, or when BL or BLX instructions are executed within interrupt or exception routines. There is another register called Saved Program Status Register (SPSR) that becomes available in privileged modes instead of CPSR. This register contains condition code flags and the current mode bits saved as a result of the exception that caused entry to the current (privileged) mode.

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

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

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

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

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

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

11.3.7.1 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 11-2. Status Register Format

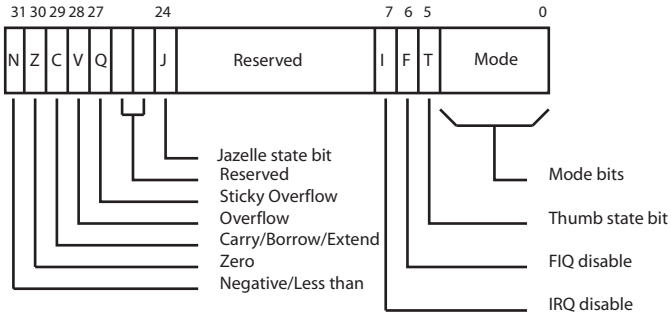


Figure 11-2 shows the status register format, where:

- 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

11.3.7.2 Exceptions

Exception Types and Priorities

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

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

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

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

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

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

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

Exception Modes and Handling

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

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

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

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

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

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

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

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

The breakpoint (BKPT) instruction is a new feature of ARM9EJ-S that is destined to solve the problem of the Prefetch Abort. A breakpoint instruction operates as though the instruction caused a Prefetch Abort.
A breakpoint instruction does not cause the ARM9EJ-S to take the Prefetch Abort exception until the instruction reaches the Execute stage of the pipeline. If the instruction is not executed, for example because a branch occurs while it is in the pipeline, the breakpoint does not take place.

11.3.8 ARM Instruction Set Overview

The ARM instruction set is divided into:

- Branch instructions

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

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

Table 11-2 gives the ARM instruction mnemonic list.

Table 11-2. 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		
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		

11.3.9 New ARM Instruction Set

.

Table 11-3. New ARM Instruction Mnemonic List

Mnemonic	Operation
BXJ	Branch and exchange to Java
BLX [1]	Branch, Link and exchange
SMLAxy	Signed Multiply Accumulate 16 * 16 bit
SMLAL	Signed Multiply Accumulate Long
SMLAWy	Signed Multiply Accumulate 32 * 16 bit
SMULxy	Signed Multiply 16 * 16 bit
SMULWy	Signed Multiply 32 * 16 bit
QADD	Saturated Add
QDADD	Saturated Add with Double
QSUB	Saturated subtract
QDSUB	Saturated Subtract with double

Mnemonic	Operation
MRRC	Move double from coprocessor
MCR2	Alternative move of ARM reg to coprocessor
MCRR	Move double to coprocessor
CDP2	Alternative Coprocessor Data Processing
BKPT	Breakpoint
PLD	Soft Preload, Memory prepare to load from address
STRD	Store Double
STC2	Alternative Store from Coprocessor
LDRD	Load Double
LDC2	Alternative Load to Coprocessor
CLZ	Count Leading Zeroes

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

11.3.10 Thumb Instruction Set Overview

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

The Thumb instruction set is divided into:

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

Table 5 shows the Thumb instruction set. Table 11-4 gives the Thumb instruction mnemonic list.

Table 11-4. Thumb Instruction Mnemonic List

Mnemonic	Operation
MOV	Move
ADD	Add
SUB	Subtract
CMP	Compare
TST	Test
AND	Logical AND

Mnemonic	Operation
MVN	Move Not
ADC	Add with Carry
SBC	Subtract with Carry
CMN	Compare Negated
NEG	Negate
BIC	Bit Clear

Table 11-4. Thumb Instruction Mnemonic List (Continued)

Mnemonic	Operation
EOR	Logical Exclusive OR
LSL	Logical Shift Left
ASR	Arithmetic Shift Right
MUL	Multiply
B	Branch
BX	Branch and Exchange
LDR	Load Word
LDRH	Load Half Word
LDRB	Load Byte
LDRSH	Load Signed Halfword
LDMIA	Load Multiple
PUSH	Push Register to stack
BCC	Conditional Branch

Mnemonic	Operation
ORR	Logical (inclusive) OR
LSR	Logical Shift Right
ROR	Rotate Right
BLX	Branch, Link, and Exchange
BL	Branch and Link
SWI	Software Interrupt
STR	Store Word
STRH	Store Half Word
STRB	Store Byte
LDRSB	Load Signed Byte
STMIA	Store Multiple
POP	Pop Register from stack
BKPT	Breakpoint

11.4 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 11-5.

Table 11-5. CP15 Registers

Register	Name	Read/Write
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

Table 11-5. CP15 Registers

Register	Name	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.

11.4.1 CP15 Registers Access

CP15 registers can only be accessed in privileged mode by:

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

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

The assembler code for these instructions is:

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

The MCR, MRC instructions bit pattern is shown below:

31	30	29	28	27	26	25	24
cond				1	1	1	0

23	22	21	20	19	18	17	16
opcode_1			L	CRn			

15	14	13	12	11	10	9	8
Rd				1	1	1	1

7	6	5	4	3	2	1	0
opcode_2			1	CRm			

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

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

- **opcode_2[7:5]**

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

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

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

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

Determines the destination coprocessor register.

- **L: Instruction Bit**

0 = MCR instruction

1 = MRC instruction

- **opcode_1[23:20]: Coprocessor Code**

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

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

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

11.5 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 11-6. 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

11.5.1 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).

11.5.2 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.

11.5.3 Translation Table Walk Hardware

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

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

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

11.5.4 MMU Faults

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

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

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

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

11.6 Caches and Write Buffer

The ARM926EJ-S contains a 16 KB Instruction Cache (ICache), a 16 KB Data Cache (DCache), and a write buffer. Although the ICache and DCache share common features, each still has some specific mechanisms.

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

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

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

11.6.1 Instruction Cache (ICache)

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

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

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

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

11.6.2 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.

11.6.2.1 DCache

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

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

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

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

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

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

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

11.6.2.2 Write Buffer

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

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

Write-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.

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.

11.7 Tightly-Coupled Memory Interface

11.7.1 TCM Description

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

The user has the possibility to independently configure each TCM size with values within the following ranges, [0KB, 64 KB] for ITCM size and [0KB, 64 KB] for DTCM size.

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

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

11.7.2 Enabling and Disabling TCMs

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

11.7.3 TCM Mapping

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

11.8 Bus Interface Unit

The ARM926EJ-S features a Bus Interface Unit (BIU) that arbitrates and schedules AHB requests. The BIU implements a multi-layer AHB, based on the AHB-Lite protocol, that enables parallel access paths between multiple AHB masters and slaves in a system. This is achieved by using a more complex interconnection matrix and gives the benefit of increased overall bus bandwidth, and a more flexible system architecture.

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

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

11.8.1 Supported Transfers

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

Table 11-7 gives an overview of the supported transfers and different kinds of transactions they are used for.

Table 11-7. 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

11.8.2 Thumb Instruction Fetches

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

11.8.3 Address Alignment

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

12. AT91SAM9263 Debug and Test

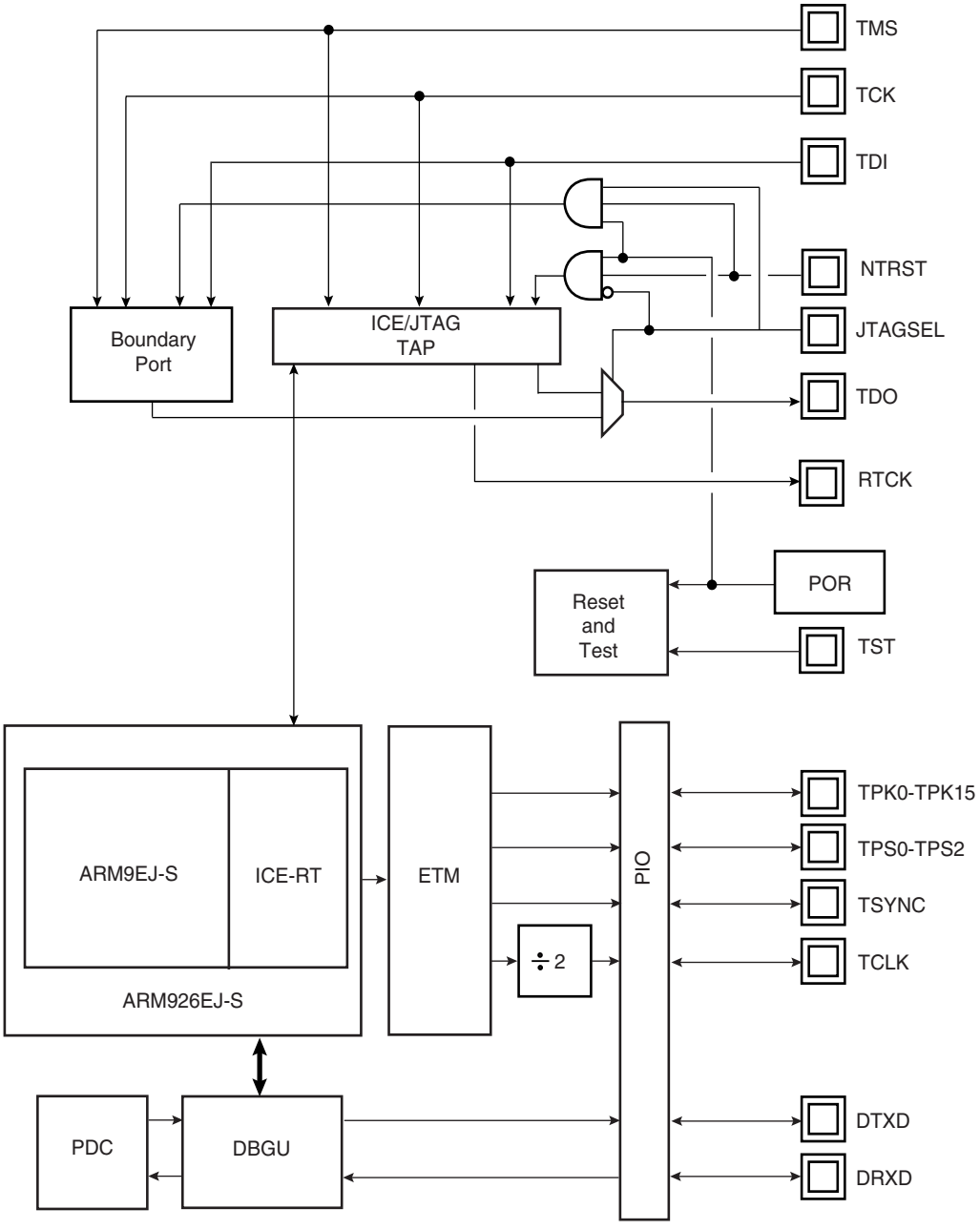
12.1 Overview

The AT91SAM9263 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. An ETM (Embedded Trace Macrocell) provides more sophisticated debug features such as address and data comparators, half-rate clock mode, counters, sequencer and FIFO. 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.

12.2 Block Diagram

Figure 12-1. Debug and Test Block Diagram



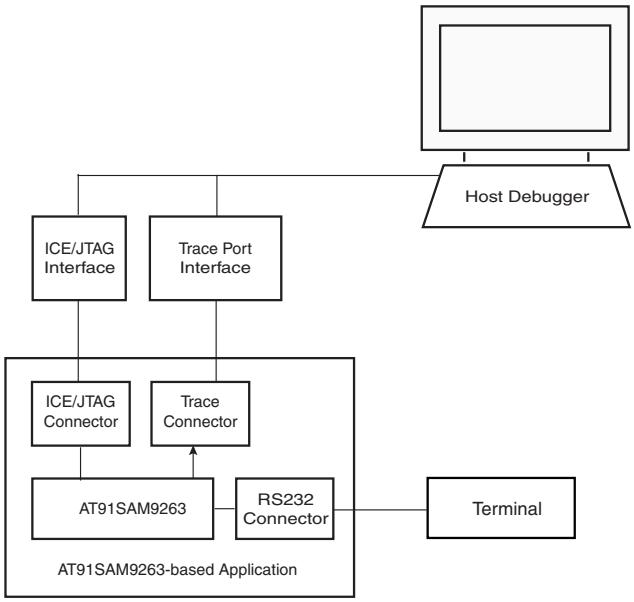
TAP: Test Access Port

12.3 Application Examples

12.3.1 Debug Environment

Figure 12-2 on page 67 shows a complete debug environment example. The ICE/JTAG interface is used for standard debugging functions, such as downloading code and single-stepping through the program. The Trace Port interface is used for tracing information. A software debugger running on a personal computer provides the user interface for configuring a Trace Port interface utilizing the ICE/JTAG interface.

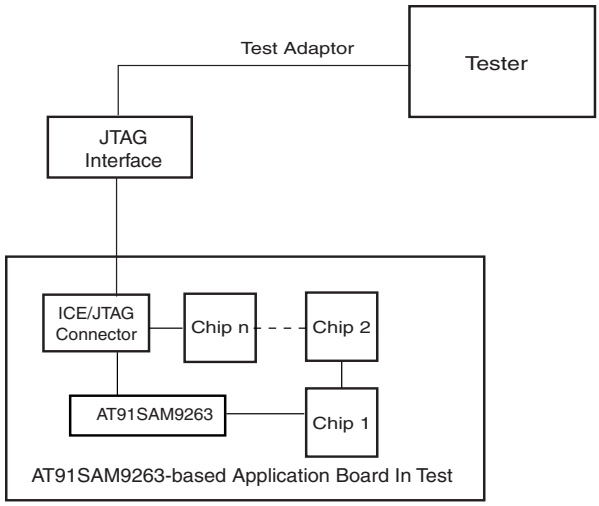
Figure 12-2. Application Debug and Trace Environment Example



12.3.2 Test Environment

Figure 12-3 on page 67 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 12-3. Application Test Environment Example



12.4 Debug and Test Pin Description

Table 12-1. Debug and Test Pin List

Pin Name	Function	Type	Active Level
Reset/Test			
NTRST	Test Reset Signal	Input	Low
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	
JTAGSEL	JTAG Selection	Input	
ETM			
TSYNC	Trace Synchronization Signal	Output	
TCLK	Trace Clock	Output	
TPS0 - TPS2	Trace ARM Pipeline Status	Output	
TPK0 - TPK15	Trace Packet Port	Output	
Debug Unit			
DRXD	Debug Receive Data	Input	
DTXD	Debug Transmit Data	Output	

12.5 Functional Description

12.5.1 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.

12.5.2 Embedded In-circuit Emulator

The ARM9EJ-S Embedded In-Circuit Emulator-RT is supported via the ICE/JTAG port. It is connected to a host computer via an ICE interface. Debug support is implemented using an ARM9EJ-S core embedded within the ARM926EJ-S. The internal state of the ARM926EJ-S is examined through an ICE/JTAG port which allows instructions to be serially inserted into the pipeline of the core without using the external data bus. Therefore, when in debug state, a store-multiple (STM) can be inserted into the instruction pipeline. This exports the contents of the ARM9EJ-S registers. This data can be serially shifted out without affecting the rest of the system.

There are two scan chains inside the ARM9EJ-S processor which support testing, debugging, and programming of the Embedded ICE-RT. The scan chains are controlled by the ICE/JTAG port.

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

For further details on the Embedded In-Circuit-Emulator-RT, see the ARM document:

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

12.5.3 JTAG Signal Description

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

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

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

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

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

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

12.5.4 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 AT91SAM9263 Debug Unit Chip ID value is 0x0196 07A0 on 32-bit width.

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

12.5.5 Embedded Trace Macrocell

The AT91SAM9263 features an Embedded Trace Macrocell (ETM), which is closely connected to the ARM926EJ-S Processor. The Embedded Trace is a standard Medium+ level implementation and contains the following resources:

- Four pairs of address comparators
- Two data comparators
- Eight memory map decoder inputs
- Two 16-bits counters
- One 3-stage sequencer
- Four external inputs
- One external output
- One 45-byte FIFO

The Embedded Trace Macrocell of the AT91SAM9263 works in half-rate clock mode and thus integrates a clock divider. This allows the maximum frequency of all the trace port signals not to exceed one half of the ARM926EJ-S clock speed.

The Embedded Trace Macrocell input and output resources are not used in the AT91SAM9263.

The Embedded Trace is a real-time trace module with the capability of tracing the ARM9EJ-S instruction and data.

For further details on Embedded Trace Macrocell, see the ARM documents:

- ETM9 (Rev2p2) Technical Reference Manual (DDI 0157F)
- Embedded Trace Macrocell Specification (IHI 0014J)

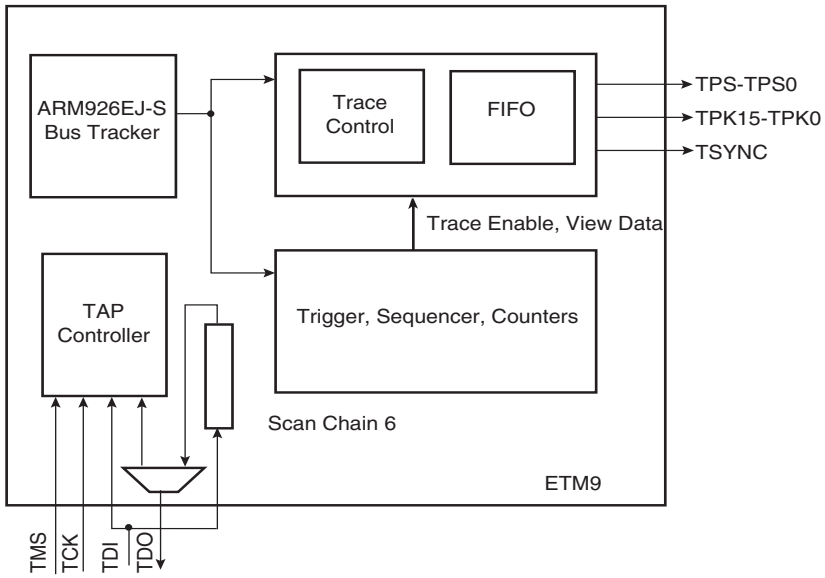
12.5.5.1 Trace Port

The Trace Port is made up of the following pins:

- TSYNC - the synchronization signal (Indicates the start of a branch sequence on the trace packet port.)
- TCLK - the Trace Port clock, half-rate of the ARM926EJ-S processor clock.
- TPS0 to TPS2 - indicate the processor state at each trace clock edge.
- TPK0 to TPK15 - the Trace Packet data value.

The trace packet information (address, data) is associated with the processor state indicated by TPS. Some processor states have no additional data associated with the Trace Packet Port (i.e. failed condition code of an instruction). The packet is 8-bits wide, and up to two packets can be output per cycle.

Figure 12-4. ETM9 Block



12.5.5.2 Implementation Details

This section gives an overview of the Embedded Trace resources.

Three-state Sequencer

The sequencer has three possible next states (one dedicated to itself and two others) and can change on every clock cycle. The sate transition is controlled with internal events. If the user needs multiple-stage trigger schemes, the trigger event is based on a sequencer state.

Address Comparator

In single mode, address comparators compare either the instruction address or the data address against the user-programmed address.

In range mode, the address comparators are arranged in pairs to form a virtual address range resource.

Details of the address comparator programming are:

- The first comparator is programmed with the range start address.
- The second comparator is programmed with the range end address.
- The resource matches if the address is within the following range:
  - (address > = range start address) AND (address < range end address)
- Unpredictable behavior occurs if the two address comparators are not configured in the same way.

Data Comparator

Each full address comparator is associated with a specific data comparator. A data comparator is used to observe the data bus only when load and store operations occur.

A data comparator has both a value register and a mask register, therefore it is possible to compare only certain bits of a preprogrammed value against the data bus.

Memory Decoder Inputs

The eight memory map decoder inputs are connected to custom address decoders. The address decoders divide the memory into regions of on-chip SRAM, on-chip ROM, and peripherals. The address decoders also optimize the ETM9 trace trigger.

Table 12-2. ETM Memory Map Inputs Layout

Product Resource	Area	Access Type	Start Address	End Address
SRAM	Internal	Data	0x0000 0000	0x002F FFFF
SRAM	Internal	Fetch	0x0000 0000	0x002F FFFF
ROM	Internal	Data	0x0040 0000	0x004F FFFF
ROM	Internal	Fetch	0x0040 0000	0x004F FFFF
External Bus Interface	External	Data	0x1000 0000	0x9FFF FFFF
External Bus Interface	External	Fetch	0x1000 0000	0x9FFF FFFF
User Peripherals	Internal	Data	0xF000 0000	0xFFFF BFFF
System Peripherals	Internal	Data	0xFFFF C000	0xFFFF FFFF

FIFO

A 45-byte FIFO is used to store data tracing. The FIFO is used to separate the pipeline status from the trace packet. So, the FIFO can be used to buffer trace packets.

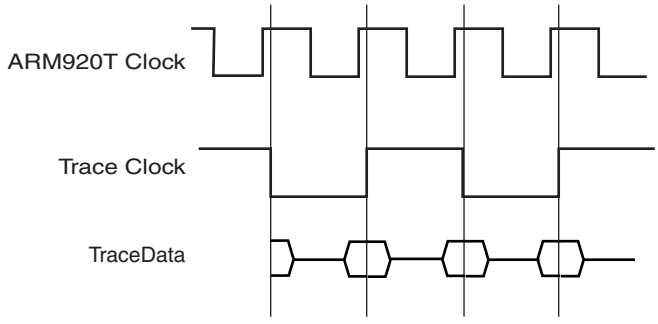
A FIFO overflow is detected by the embedded trace macrocell when the FIFO is full or when the FIFO has less bytes than the user-programmed number.

Half-rate Clocking Mode

The ETM9 is implemented in half-rate mode that allows both rising and falling edge data tracing of the trace clock.

The half-rate mode is implemented to maintain the signal clock integrity of high speed systems (up to 100 MHz).

Figure 12-5. Half-rate Clocking Mode



Half-rate Clocking Mode

Care must be taken on the choice of the trace capture system as it needs to support half-rate clock functionality.

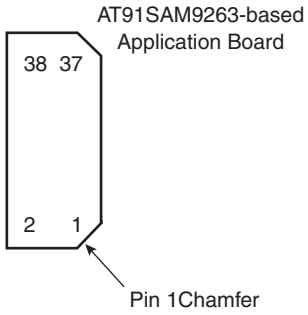
12.5.5.3 *Application Board Restriction*

The TCLK signal needs to be set with care, some timing parameters are required. See “ETM Timings” for more details.

The specified target system connector is the AMP Mictor connector.

The connector must be oriented on the application board as described below in Figure 12-6. The view of the PCB is shown from above with the trace connector mounted near the edge of the board. This allows the Trace Port Analyzer to minimize the physical intrusiveness of the interconnected target.

Figure 12-6. AMP Mictor Connector Orientation



12.5.6 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.

12.5.6.1 *JTAG Boundary Scan Register*

The Boundary Scan Register (BSR) contains 664 bits that correspond to active pins and associated control signals.

Each AT91SAM9263 input/output pin corresponds to a 3-bit register in the BSR. The OUTPUT bit contains data that can be forced on the pad. The INPUT bit facilitates the observability of data applied to the pad. The CONTROL bit selects the direction of the pad.

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register

Bit Number	Pin Name	Pin Type	Associated BSR Cells
663	PA19	IN/OUT	INPUT
662			OUTPUT
661			CONTROL
660	PA20	IN/OUT	INPUT
659			OUTPUT
658			CONTROL
657	PA21	IN/OUT	INPUT
656			OUTPUT
655			CONTROL
654	PA22	IN/OUT	INPUT
653			OUTPUT
652			CONTROL
651	PA23	IN/OUT	INPUT
650			OUTPUT
649			CONTROL
648	PA24	IN/OUT	INPUT
647			OUTPUT
646			CONTROL
645	PA25	IN/OUT	INPUT
644			OUTPUT
643			CONTROL
642	PA26	IN/OUT	INPUT
641			OUTPUT
640			CONTROL
639	PA27	IN/OUT	INPUT
638			OUTPUT
637			CONTROL
636	PA28	IN/OUT	INPUT
635			OUTPUT
634			CONTROL

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
633	PA29	IN/OUT	INPUT
632			OUTPUT
631			CONTROL
630	PA30	IN/OUT	INPUT
629			OUTPUT
628			CONTROL
627	PA31	IN/OUT	INPUT
626			OUTPUT
625			CONTROL
624	EBI1_A0_NBS0	OUT	OUTPUT
623	EBI1_A[7:0]		CONTROL
622	EBI1_A1_NWR2	IN/OUT	INPUT
621			OUTPUT
620	EBI1_A2	OUT	OUTPUT
619	EBI1_A3	OUT	OUTPUT
618	EBI1_A4	OUT	OUTPUT
617	EBI1_A5	OUT	OUTPUT
616	EBI1_A6	OUT	OUTPUT
615	EBI1_A7	OUT	OUTPUT
614	EBI1_A8	OUT	OUTPUT
613	EBI1_A[15:8]		CONTROL
612	EBI1_A9	OUT	OUTPUT
611	EBI1_A10	OUT	OUTPUT
610	EBI1_A11	OUT	OUTPUT
609	EBI1_A12	OUT	OUTPUT
608	EBI1_A13	OUT	OUTPUT
607	EBI1_A14	OUT	OUTPUT
606	EBI1_A15	OUT	OUTPUT
605	EBI1_A16_BA0	OUT	OUTPUT
604	EBI1_A[22:16]		CONTROL
603	EBI1_A17	OUT	OUTPUT
602	EBI1_A18	OUT	OUTPUT
601	EBI1_A19	OUT	OUTPUT
600	EBI1_A20	OUT	OUTPUT
599	EBI1_A21	OUT	OUTPUT

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
598	EBI1_A22	OUT	OUTPUT
597	EBI1_NCS0	OUT	OUTPUT
596	EBI1_NCS0/EBI1_NRD/EBI1_NWR_NWR0/ EBI1_NWR_NWR1		CONTROL
595	EBI1_NRD	OUT	OUTPUT
594	EBI1_NWR_NWR0	IN/OUT	INPUT
593			OUTPUT
592	EBI1_NWR_NWR1	IN/OUT	INPUT
591			OUTPUT
590	EBI1_D0	IN/OUT	INPUT
589			OUTPUT
588			CONTROL
587	EBI1_D1	IN/OUT	INPUT
586			OUTPUT
585			CONTROL
584	EBI1_D2	IN/OUT	INPUT
583			OUTPUT
582			CONTROL
581	EBI1_D3	IN/OUT	INPUT
580			OUTPUT
579			CONTROL
578	EBI1_D4	IN/OUT	INPUT
577			OUTPUT
576			CONTROL
575	EBI1_D5	IN/OUT	INPUT
574			OUTPUT
573			CONTROL
572	EBI1_D6	IN/OUT	INPUT
571			OUTPUT
570			CONTROL
569	EBI1_D7	IN/OUT	INPUT
568			OUTPUT
567			CONTROL
566	EBI1_D8	IN/OUT	INPUT
565			OUTPUT
564			CONTROL

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
563	EBI1_D9	IN/OUT	INPUT
562			OUTPUT
561			CONTROL
560	EBI1_D10	IN/OUT	INPUT
559			OUTPUT
558			CONTROL
557	EBI1_D11	IN/OUT	INPUT
556			OUTPUT
555			CONTROL
554	EBI1_D12	IN/OUT	INPUT
553			OUTPUT
552			CONTROL
551	EBI1_D13	IN/OUT	INPUT
550			OUTPUT
549			CONTROL
548	EBI1_D14	IN/OUT	INPUT
547			OUTPUT
546			CONTROL
545	EBI1_D15	IN/OUT	INPUT
544			OUTPUT
543			CONTROL
542	PE20	IN/OUT	INPUT
541			OUTPUT
540			CONTROL
539	PE21	IN/OUT	INPUT
538			OUTPUT
537			CONTROL
536	PE22	IN/OUT	INPUT
535			OUTPUT
534			CONTROL
533	PE23	IN/OUT	INPUT
532			OUTPUT
531			CONTROL

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
530	PE24	IN/OUT	INPUT
529			OUTPUT
528			CONTROL
527	PE26	IN/OUT	INPUT
526			OUTPUT
525			CONTROL
524	PE25	IN/OUT	INPUT
523			OUTPUT
522			CONTROL
521	PE27	IN/OUT	INPUT
520			OUTPUT
519			CONTROL
518		internal	
517	EBI1_SDK	OUT	OUTPUT
516		internal	
515	PE28	IN/OUT	INPUT
514			OUTPUT
513			CONTROL
512	PE29	IN/OUT	INPUT
511			OUTPUT
510			CONTROL
509	PE30	IN/OUT	INPUT
508			OUTPUT
507			CONTROL
506	PE31	IN/OUT	INPUT
505			OUTPUT
504			CONTROL
503	RTCK	OUT	OUTPUT
502			CONTROL
501	PA0	IN/OUT	INPUT
500			OUTPUT
499			CONTROL
498	PA1	IN/OUT	INPUT
497			OUTPUT
496			CONTROL

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
495	PA2	IN/OUT	INPUT
494			OUTPUT
493			CONTROL
492	PA3	IN/OUT	INPUT
491			OUTPUT
490			CONTROL
489	PA4	IN/OUT	INPUT
488			OUTPUT
487			CONTROL
486	PA5	IN/OUT	INPUT
485			OUTPUT
484			CONTROL
483	PA6	IN/OUT	INPUT
482			OUTPUT
481			CONTROL
480	PA7	IN/OUT	INPUT
479			OUTPUT
478			CONTROL
477	PA8	IN/OUT	INPUT
476			OUTPUT
475			CONTROL
474	PA9	IN/OUT	INPUT
473			OUTPUT
472			CONTROL
471	PA10	IN/OUT	INPUT
470			OUTPUT
469			CONTROL
468	PA11	IN/OUT	INPUT
467			OUTPUT
466			CONTROL
465	PA12	IN/OUT	INPUT
464			OUTPUT
463			CONTROL

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
462	PA13	IN/OUT	INPUT
461			OUTPUT
460			CONTROL
459	PA14	IN/OUT	INPUT
458			OUTPUT
457			CONTROL
456	PA15	IN/OUT	INPUT
455			OUTPUT
454			CONTROL
453	PB0	IN/OUT	INPUT
452			OUTPUT
451			CONTROL
450	PB1	IN/OUT	INPUT
449			OUTPUT
448			CONTROL
447	PB2	IN/OUT	INPUT
446			OUTPUT
445			CONTROL
444	PB3	IN/OUT	INPUT
443			OUTPUT
442			CONTROL
441	PB4	IN/OUT	INPUT
440			OUTPUT
439			CONTROL
438	PB5	IN/OUT	INPUT
437			OUTPUT
436			CONTROL
435	PB6	IN/OUT	INPUT
434			OUTPUT
433			CONTROL
432	PB7	IN/OUT	INPUT
431			OUTPUT
430			CONTROL

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
429	PB8	IN/OUT	INPUT
428			OUTPUT
427			CONTROL
426	PB9	IN/OUT	INPUT
425			OUTPUT
424			CONTROL
423	PB10	IN/OUT	INPUT
422			OUTPUT
421			CONTROL
420	PB11	IN/OUT	INPUT
419			OUTPUT
418			CONTROL
417	PB12	IN/OUT	INPUT
416			OUTPUT
415			CONTROL
414	PB13	IN/OUT	INPUT
413			OUTPUT
412			CONTROL
411	PB14	IN/OUT	INPUT
410			OUTPUT
409			CONTROL
408	PB15	IN/OUT	INPUT
407			OUTPUT
406			CONTROL
405	PB16	IN/OUT	INPUT
404			OUTPUT
403			CONTROL
402	PB17	IN/OUT	INPUT
401			OUTPUT
400			CONTROL
399	PB18	IN/OUT	INPUT
398			OUTPUT
397			CONTROL

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
396	PB19	IN/OUT	INPUT
395			OUTPUT
394			CONTROL
393	PB20	IN/OUT	INPUT
392			OUTPUT
391			CONTROL
390	PB21	IN/OUT	INPUT
389			OUTPUT
388			CONTROL
387	PB22	IN/OUT	INPUT
386			OUTPUT
385			CONTROL
384	PB23	IN/OUT	INPUT
383			OUTPUT
382			CONTROL
381	PB24	IN/OUT	INPUT
380			OUTPUT
379			CONTROL
378	PB25	IN/OUT	INPUT
377			OUTPUT
376			CONTROL
375	PB26	IN/OUT	INPUT
374			OUTPUT
373			CONTROL
372	PB27	IN/OUT	INPUT
371			OUTPUT
370			CONTROL
369	PB28	IN/OUT	INPUT
368			OUTPUT
367			CONTROL
366	PB29	IN/OUT	INPUT
365			OUTPUT
364			CONTROL

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
363	PB30	IN/OUT	INPUT
362			OUTPUT
361			CONTROL
360	PB31	IN/OUT	INPUT
359			OUTPUT
358			CONTROL
357	PC0	IN/OUT	INPUT
356			OUTPUT
355			CONTROL
354	PC1	IN/OUT	INPUT
353			OUTPUT
352			CONTROL
351	PC2	IN/OUT	INPUT
350			OUTPUT
349			CONTROL
348	PC3	IN/OUT	INPUT
347			OUTPUT
346			CONTROL
345	PC4	IN/OUT	INPUT
344			OUTPUT
343			CONTROL
342	PC5	IN/OUT	INPUT
341			OUTPUT
340			CONTROL
339	PC6	IN/OUT	INPUT
338			OUTPUT
337			CONTROL
336	PC7	IN/OUT	INPUT
335			OUTPUT
334			CONTROL
333	PC8	IN/OUT	INPUT
332			OUTPUT
331			CONTROL

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
330	PC9	IN/OUT	INPUT
329			OUTPUT
328			CONTROL
327	PC10	IN/OUT	INPUT
326			OUTPUT
325			CONTROL
324	PC11	IN/OUT	INPUT
323			OUTPUT
322			CONTROL
321	PC12	IN/OUT	INPUT
320			OUTPUT
319			CONTROL
318	PC13	IN/OUT	INPUT
317			OUTPUT
316			CONTROL
315	PC14	IN/OUT	INPUT
314			OUTPUT
313			CONTROL
312	PC15	IN/OUT	INPUT
311			OUTPUT
310			CONTROL
309	PC16	IN/OUT	INPUT
308			OUTPUT
307			CONTROL
306	PC17	IN/OUT	INPUT
305			OUTPUT
304			CONTROL
303	PC18	IN/OUT	INPUT
302			OUTPUT
301			CONTROL
300	PC19	IN/OUT	INPUT
299			OUTPUT
298			CONTROL

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
297	PC20	IN/OUT	INPUT
296			OUTPUT
295			CONTROL
294	PC21	IN/OUT	INPUT
293			OUTPUT
292			CONTROL
291	PC22	IN/OUT	INPUT
290			OUTPUT
289			CONTROL
288	PC23	IN/OUT	INPUT
287			OUTPUT
286			CONTROL
285	PC24	IN/OUT	INPUT
284			OUTPUT
283			CONTROL
282	PC25	IN/OUT	INPUT
281			OUTPUT
280			CONTROL
279	PC26	IN/OUT	INPUT
278			OUTPUT
277			CONTROL
276	PC27	IN/OUT	INPUT
275			OUTPUT
274			CONTROL
273	PC28	IN/OUT	INPUT
272			OUTPUT
271			CONTROL
270	PC29	IN/OUT	INPUT
269			OUTPUT
268			CONTROL
267	PC30	IN/OUT	INPUT
266			OUTPUT
265			CONTROL

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
264	PC31	IN/OUT	INPUT
263			OUTPUT
262			CONTROL
261	PD0	IN/OUT	INPUT
260			OUTPUT
259			CONTROL
258	PD1	IN/OUT	INPUT
257			OUTPUT
256			CONTROL
255	PD2	IN/OUT	INPUT
254			OUTPUT
253			CONTROL
252	PD3	IN/OUT	INPUT
251			OUTPUT
250			CONTROL
249	PD4	IN/OUT	INPUT
248			OUTPUT
247			CONTROL
246	N.C.	OUT	OUTPUT
245	EBI0_A0_NBS0	OUT	OUTPUT
244	EBI0_A[7:0]		CONTROL
243	EBI0_A1_NBS2_NWR2	IN/OUT	INPUT
242			OUTPUT
241	EBI0_A2	OUT	OUTPUT
240	EBI0_A3	OUT	OUTPUT
239	EBI0_A4	OUT	OUTPUT
238	EBI0_A5	OUT	OUTPUT
237	EBI0_A6	OUT	OUTPUT
236	EBI0_A7	OUT	OUTPUT
235	EBI0_A8	OUT	OUTPUT
234	EBI0_A[15:8]		CONTROL
233	EBI0_A9	OUT	OUTPUT
232	EBI0_A10	OUT	OUTPUT
231	EBI0_SDA10	OUT	OUTPUT
230	EBI0_SDA10/SDCKE/RAS/CAS/ SDWE/NANDOE/NANDWE		CONTROL

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
229	EBI0_A11	OUT	OUTPUT
228	EBI0_A12	OUT	OUTPUT
227	EBI0_A13	OUT	OUTPUT
226	EBI0_A14	OUT	OUTPUT
225	EBI0_A15	OUT	OUTPUT
224	EBI0_A16_BA0	OUT	OUTPUT
223	EBI0_A[22:16]		CONTROL
222	EBI0_A17_BA1	OUT	OUTPUT
221	EBI0_A18	OUT	OUTPUT
220	EBI0_A19	OUT	OUTPUT
219	EBI0_A20	OUT	OUTPUT
218	EBI0_A21	OUT	OUTPUT
217	EBI0_A22	OUT	OUTPUT
216	EBI0_NCS0	OUT	OUTPUT
215	EBI0_NCS0/EBI0_NCS1_SDCS/EBI0_NRD/ EBI0_NWR_NWR0/EBI0_NBS1_NWR1/EBI0_NBS3_NWR3		CONTROL
214	EBI0_NCS1_SDCS	OUT	OUTPUT
213	EBI0_NRD	OUT	OUTPUT
212	EBI0_NWR_NWR0	IN/OUT	INPUT
211			OUTPUT
210	EBI0_NBS1_NWR1	IN/OUT	INPUT
209			OUTPUT
208	EBI0_NBS3_NWR3	IN/OUT	INPUT
207			OUTPUT
206		internal	
205	EBI0_SDCK	OUT	OUTPUT
204		internal	
203	EBI0_SDCKE	OUT	OUTPUT
202	EBI0_RAS	OUT	OUTPUT
201	EBI0_CAS	OUT	OUTPUT
200	EBI0_SDWE	OUT	OUTPUT
199	EBI0_NANDOE	OUT	OUTPUT
198	EBI0_NANDWE	OUT	OUTPUT
197	EBI0_D0	IN/OUT	INPUT
196			OUTPUT
195			CONTROL

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
194	EBI0_D1	IN/OUT	INPUT
193			OUTPUT
192			CONTROL
191	EBI0_D2	IN/OUT	INPUT
190			OUTPUT
189			CONTROL
188	EBI0_D3	IN/OUT	INPUT
187			OUTPUT
186			CONTROL
185	EBI0_D4	IN/OUT	INPUT
184			OUTPUT
183			CONTROL
182	EBI0_D5	IN/OUT	INPUT
181			OUTPUT
180			CONTROL
179	EBI0_D6	IN/OUT	INPUT
178			OUTPUT
177			CONTROL
176	EBI0_D7	IN/OUT	INPUT
175			OUTPUT
174			CONTROL
173	EBI0_D8	IN/OUT	INPUT
172			OUTPUT
171			CONTROL
170	EBI0_D9	IN/OUT	INPUT
169			OUTPUT
168			CONTROL
167	EBI0_D10	IN/OUT	INPUT
166			OUTPUT
165			CONTROL
164	EBI0_D11	IN/OUT	INPUT
163			OUTPUT
162			CONTROL

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
161	EBI0_D12	IN/OUT	INPUT
160			OUTPUT
159			CONTROL
158	EBI0_D13	IN/OUT	INPUT
157			OUTPUT
156			CONTROL
155	EBI0_D14	IN/OUT	INPUT
154			OUTPUT
153			CONTROL
152	EBI0_D15	IN/OUT	INPUT
151			OUTPUT
150			CONTROL
149	PD5	IN/OUT	INPUT
148			OUTPUT
147			CONTROL
146	PD6	IN/OUT	INPUT
145			OUTPUT
144			CONTROL
143	PD12	IN/OUT	INPUT
142			OUTPUT
141			CONTROL
140	PD7	IN/OUT	INPUT
139			OUTPUT
138			CONTROL
137	PD8	IN/OUT	INPUT
136			OUTPUT
135			CONTROL
134	PD9	IN/OUT	INPUT
133			OUTPUT
132			CONTROL
131	PD10	IN/OUT	INPUT
130			OUTPUT
129			CONTROL

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
128	PD11	IN/OUT	INPUT
127			OUTPUT
126			CONTROL
125	PD13	IN/OUT	INPUT
124			OUTPUT
123			CONTROL
122	PD14	IN/OUT	INPUT
121			OUTPUT
120			CONTROL
119	PD15	IN/OUT	INPUT
118			OUTPUT
117			CONTROL
116	PD16	IN/OUT	INPUT
115			OUTPUT
114			CONTROL
113	PD17	IN/OUT	INPUT
112			OUTPUT
111			CONTROL
110	PD18	IN/OUT	INPUT
109			OUTPUT
108			CONTROL
107	PD19	IN/OUT	INPUT
106			OUTPUT
105			CONTROL
104	PD20	IN/OUT	INPUT
103			OUTPUT
102			CONTROL
101	PD21	IN/OUT	INPUT
100			OUTPUT
99			CONTROL
98	PD22	IN/OUT	INPUT
97			OUTPUT
96			CONTROL

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
95	PD23	IN/OUT	INPUT
94			OUTPUT
93			CONTROL
92	PD24	IN/OUT	INPUT
91			OUTPUT
90			CONTROL
89	PD25	IN/OUT	INPUT
88			OUTPUT
87			CONTROL
86	PD26	IN/OUT	INPUT
85			OUTPUT
84			CONTROL
83	PD27	IN/OUT	INPUT
82			OUTPUT
81			CONTROL
80	PD28	IN/OUT	INPUT
79			OUTPUT
78			CONTROL
77	PD29	IN/OUT	INPUT
76			OUTPUT
75			CONTROL
74	PD30	IN/OUT	INPUT
73			OUTPUT
72			CONTROL
71	PD31	IN/OUT	INPUT
70			OUTPUT
69			CONTROL
68	PE0	IN/OUT	INPUT
67			OUTPUT
66			CONTROL
65	PE1	IN/OUT	INPUT
64			OUTPUT
63			CONTROL

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
62	PE2	IN/OUT	INPUT
61			OUTPUT
60			CONTROL
59	PE3	IN/OUT	INPUT
58			OUTPUT
57			CONTROL
56	PE4	IN/OUT	INPUT
55			OUTPUT
54			CONTROL
53	PE5	IN/OUT	INPUT
52			OUTPUT
51			CONTROL
50	PE6	IN/OUT	INPUT
49			OUTPUT
48			CONTROL
47	PE7	IN/OUT	INPUT
46			OUTPUT
45			CONTROL
44	PE8	IN/OUT	INPUT
43			OUTPUT
42			CONTROL
41	PE9	IN/OUT	INPUT
40			OUTPUT
39			CONTROL
38	PE10	IN/OUT	INPUT
37			OUTPUT
36			CONTROL
35	PE11	IN/OUT	INPUT
34			OUTPUT
33			CONTROL
32	PE12	IN/OUT	INPUT
31			OUTPUT
30			CONTROL

Table 12-3. AT91SAM9263 JTAG Boundary Scan Register (Continued)

Bit Number	Pin Name	Pin Type	Associated BSR Cells
29	PE13	IN/OUT	INPUT
28			OUTPUT
27			CONTROL
26	PE14	IN/OUT	INPUT
25			OUTPUT
24			CONTROL
23	PE15	IN/OUT	INPUT
22			OUTPUT
21			CONTROL
20	PE16	IN/OUT	INPUT
19			OUTPUT
18			CONTROL
17	PE17	IN/OUT	INPUT
16			OUTPUT
15			CONTROL
14	PE18	IN/OUT	INPUT
13			OUTPUT
12			CONTROL
11	PE19	IN/OUT	INPUT
10			OUTPUT
09			CONTROL
08	PA16	IN/OUT	INPUT
07			OUTPUT
06			CONTROL
05	PA17	IN/OUT	INPUT
04			OUTPUT
03			CONTROL
02	PA18	IN/OUT	INPUT
01			OUTPUT
00			CONTROL

12.5.7 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 0x05B0_C03F.

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

Product part Number is 0x5B0C

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

Set to 0x0.

13. AT91SAM9263 Boot Program

13.1 Overview

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

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

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

If the SD Card is not formatted or if boot.bin file is not found, NAND Flash Boot program is then executed.

The NAND Flash Boot program looks for a sequence of seven valid ARM exception vectors. If such a sequence is found, code is downloaded into the internal SRAM. This is followed by a remap and a jump to the first address of the SRAM.

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

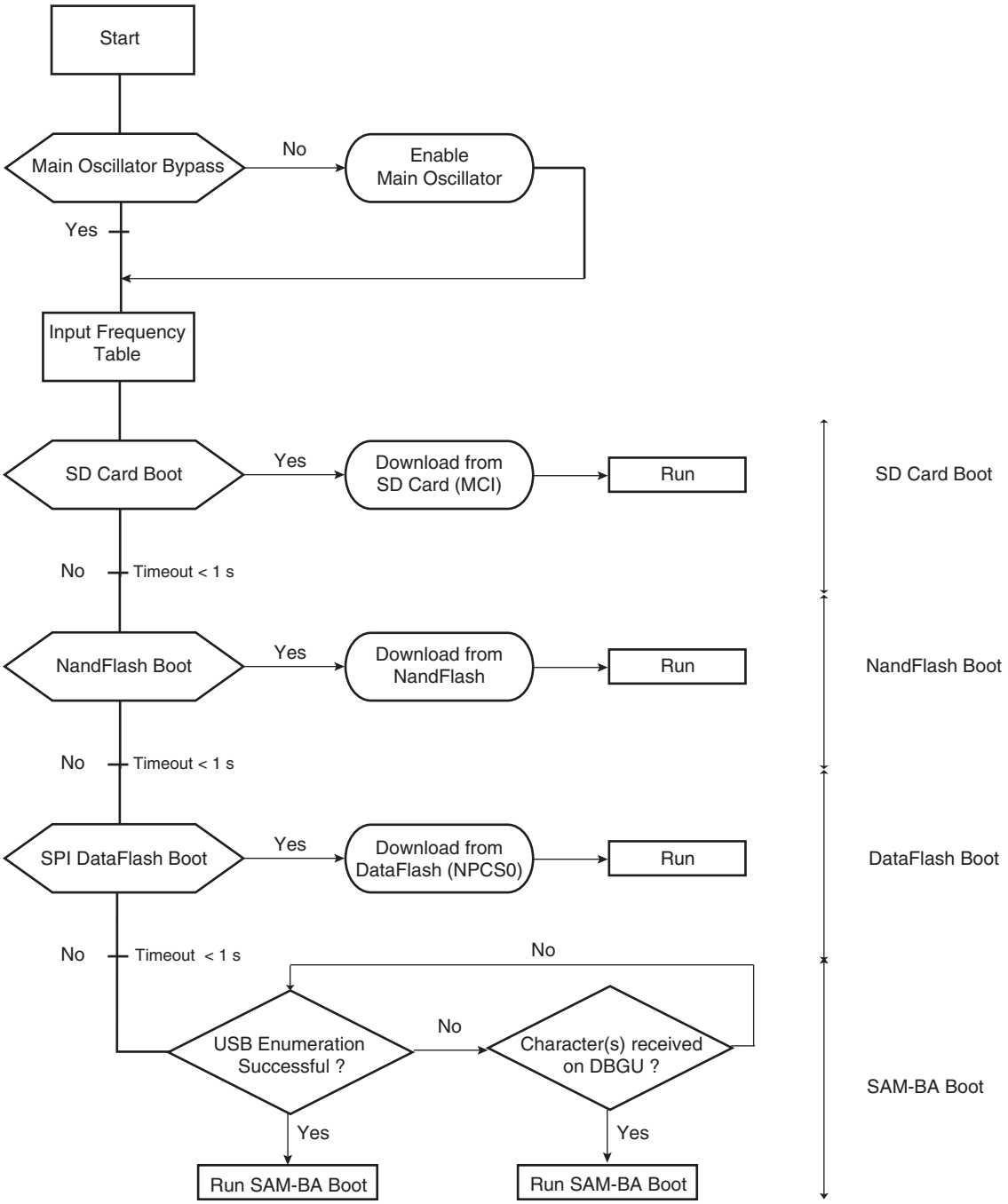
If a valid sequence is found, code is downloaded into the internal SRAM. This is followed by a remap and a jump to the first address of the SRAM.

If no boot.bin file is found, SAM-BA® Boot is then executed. It waits for transactions either on the USB device, or on the DBGU serial port.

13.2 Flow Diagram

The Boot Program implements the algorithm in Figure 13-1.

Figure 13-1. Boot Program Algorithm Flow Diagram



13.3 Device Initialization

Initialization follows the steps described below:

1. Stack setup for ARM supervisor mode
2. External Clock Detection
3. Switch Master Clock on Main Oscillator
4. C variable initialization
5. Main oscillator frequency detection
6. PLL setup: PLLB is initialized to generate a 48 MHz clock necessary to use the USB Device. A register located in the Power Management Controller (PMC) determines the frequency of the main oscillator and thus the correct factor for the PLLB.

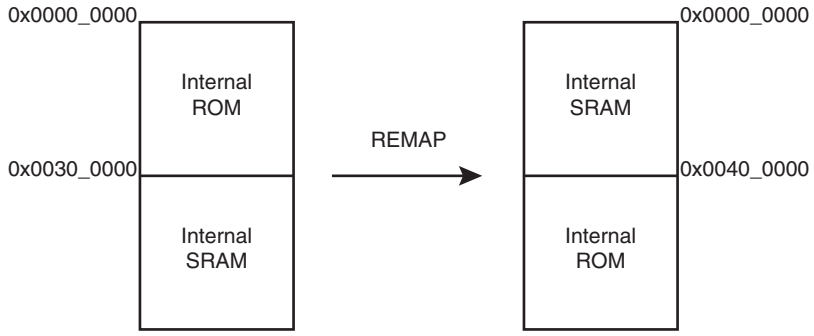
Table 13-1 defines the crystals supported by the Boot Program.

Table 13-1. Crystals Supported by Software Auto-detection (MHz)

3.0	3.2768	3.6864	3.84	4.0
4.433619	4.608	4.9152	5.0	5.24288
6.0	6.144	6.4	6.5536	7.159090
7.3728	7.864320	8.0	9.8304	10.0
11.05920	12.0	12.288	13	13.56
14.31818	14.7456	16.0	16.367667	17.734470
18.432	20.0	24	25	26
28.224	32	33	40	

7. Initialization of the DBGU serial port (115200 bauds, 8, N, 1)
8. Enable the User Reset
9. Jump to SD Card Boot sequence. If SD Card Boot succeeds, perform a remap and jump to 0x0.
10. Jump to NAND Flash Boot sequence. If NAND Flash Boot succeeds, perform a remap and jump to 0x0.
11. Jump to DataFlash Boot sequence through NPCS0. If DataFlash Boot succeeds, perform a remap and jump to 0x0.
12. Activation of the Instruction Cache
13. Jump to SAM-BA Boot sequence
14. Disable the WatchDog
15. Initialization of the USB Device Port

Figure 13-2. Remap Action after Download Completion



13.4 DataFlash Boot

The DataFlash Boot program searches for a valid application in the SPI DataFlash memory. If a valid application is found, this application is loaded into internal SRAM and executed by branching at address 0x0000_0000 after remap. This application may be the application code or a second-level bootloader.

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

After reset, the code in internal ROM is mapped at both addresses 0x0000_0000 and 0x0010_0000:

```
400000   ea000006   B      0x20              00ea000006B0x20
400004   eafffffe   B      0x04              04eafffffeB0x04
400008   ea00002f   B      _main             08ea00002fB_main
40000c   eafffffe   B      0x0c              0ceafffffeB0x0c
400010   eafffffe   B      0x10              10eafffffeB0x10
400014   eafffffe   B      0x14              14eafffffeB0x14
400018   eafffffe   B      0x18              18eafffffeB0x18
40001c   eafffffe   B      0x1c              1ceafffffeB0x1c
```

13.4.1 Valid Image Detection

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

The sixth vector, at offset 0x14, contains the size of the image to download. The user must replace this vector with his own vector (see “Structure of ARM Vector 6” on page 99).

Figure 13-3. LDR Opcode

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

Figure 13-4. B Opcode

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

Unconditional instruction: 0xE for bits 31 to 28.

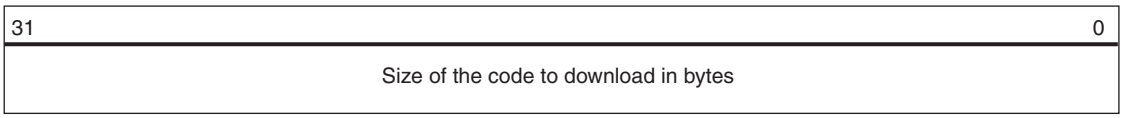
Load PC with PC relative addressing instruction:

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

13.4.2 Structure of ARM Vector 6

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

Figure 13-5. Structure of the ARM Vector 6



13.4.2.1 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
18    eafffffe  B       0x18
```

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

13.4.3 DataFlash Boot Sequence

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

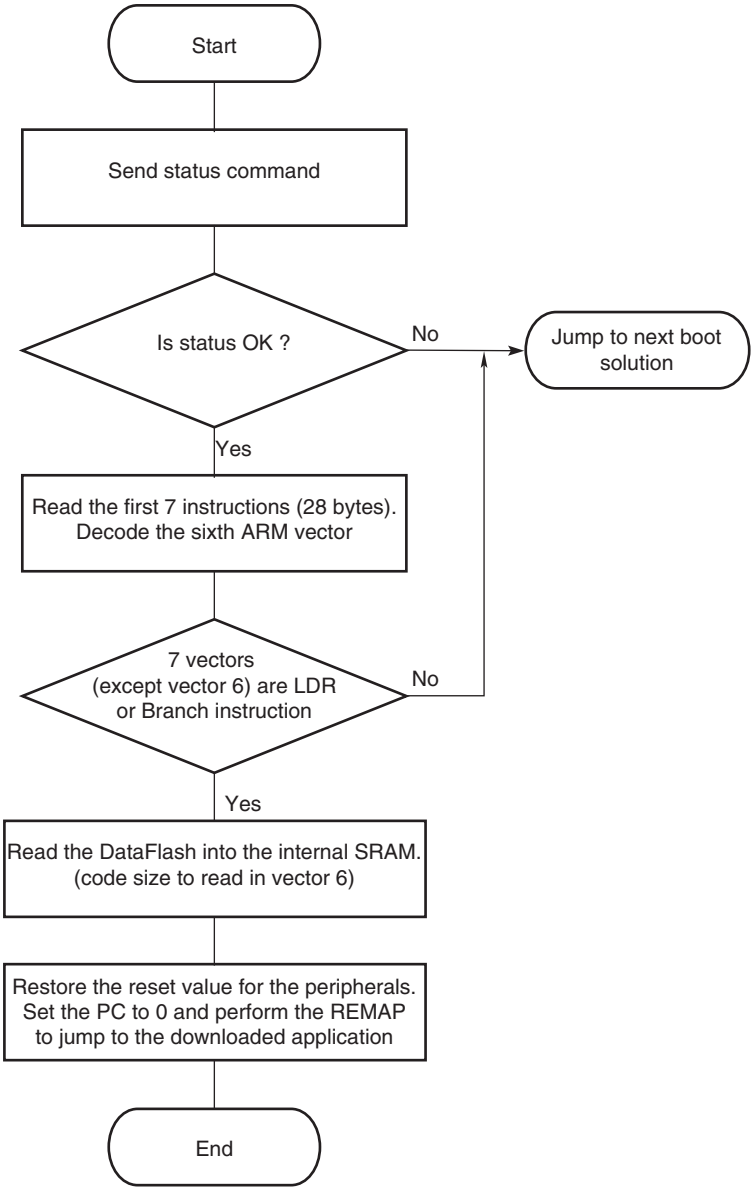
The DataFlash boot program supports all Atmel DataFlash devices. Table 13-2 summarizes the parameters to include in the ARM vector 6 for all devices.

Table 13-2. 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

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

Figure 13-6. Serial DataFlash Download



13.5 SD Card Boot

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

(Boot ROM does not support high capacity SDCards.)

It looks for a boot.bin file in the root directory of a FAT12/16/32 formatted SD Card. If a valid file is found, this application is loaded into internal SRAM and executed by branching at address 0x0000_0000 after remap. This application may be the application code or a second-level bootloader.

13.6 NAND Flash Boot

The NAND Flash Boot program searches for a valid application in the NAND Flash memory. If a valid application is found, this application is loaded into internal SRAM and executed by branching at address 0x0000_0000 after remap. See "DataFlash Boot" on page 98 for more information on Valid Image Detection.

13.6.1 Supported NAND Flash Devices

Any 8-bit or 16-bit NAND Flash device connected on EBI0 is supported.

13.7 SAM-BA Boot

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

The SAM-BA boot principle is to:

- Check if USB Device enumeration has occurred.
- Check if character(s) 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 13-3.

Table 13-3. Commands Available through the SAM-BA Boot

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 following by '>'
- Send a file (**S**): Send a file to a specified address
  - *Address*: Address in hexadecimal
  - *Output*: '>'.

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

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

13.7.1 DBGU Serial Port

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

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

13.7.2 Xmodem Protocol

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

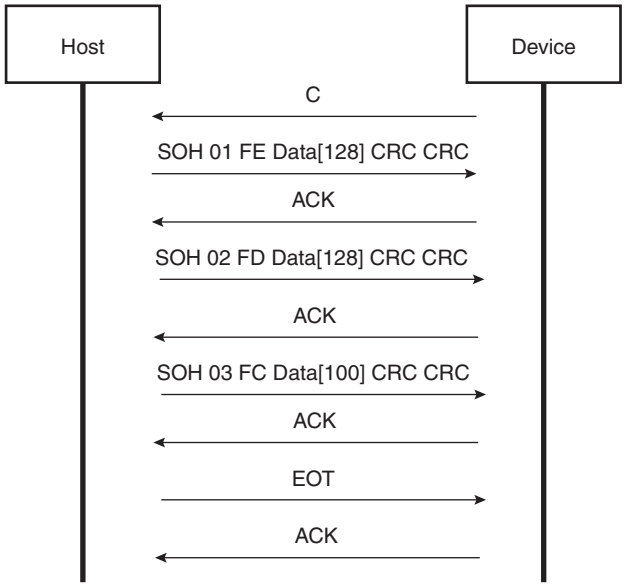
Xmodem protocol with CRC is accurate provided both sender and receiver report successful transmission. Each block of the transfer looks like:

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

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

Figure 13-7 shows a transmission using this protocol.

Figure 13-7. Xmodem Transfer Example



13.7.3 USB Device Port

A 48 MHz USB clock is necessary to use the USB Device port. It has been programmed earlier in the device initialization procedure with PLLB configuration.

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

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

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

13.7.3.1 Enumeration Process

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

Table 13-4. Handled Standard Requests

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

Table 13-4. Handled Standard Requests (Continued)

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

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

Table 13-5. 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.

13.7.3.2 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.

13.8 Hardware and Software Constraints

- SAM-BA boot disposes of two blocks of internal SRAM. The first block is available for user code. Its size is 73728 bytes. The second block is used for variables and stacks.

Table 13-6. User Area Address

Start Address	End Address	Size (bytes)
0x3000000	0x312000	73728

- The SD Card[1], NAND Flash and DataFlash downloaded code size must be inferior to 72 K bytes.
- The code is always downloaded from the DataFlash or NAND Flash device address 0x0000_0000 to the address 0x0000_0000 of the internal SRAM (after remap).
- The downloaded code must be position-independent or linked at address 0x0000_0000.
- The DataFlash must be connected to NPCS0 of the SPI.
- USB requirements:
  - Crystal or Input Frequencies supported by Software Auto-detection. See Table 13-1 on page 97 for more informations.

Note: 1. Boot ROM does not support high capacity SDCards.

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

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

Table 13-7 contains a list of pins that are driven during the boot program execution. These pins are driven during the boot sequence for a period of less than 1 second if no correct boot program is found.

For the DataFlash driven by the SPCK signal at 8 MHz, the time to download 72 KBytes is reduced to 200 ms.

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

Table 13-7. Pins Driven during Boot Program Execution

Peripheral	Pin	PIO Line
MCI1	MCCK	PIOA6
MCI1	MCCDA	PIOA7
MCI1	MCDA0	PIOA8
MCI1	MCDA1	PIOA9
MCI1	MCDA2	PIOA10
MCI1	MCDA3	PIOA11
SPI0	MOSI	PIOA1

Table 13-7. Pins Driven during Boot Program Execution (Continued)

Peripheral	Pin	PIO Line
SPI0	MISO	PIOA0
SPI0	SPCK	PIOA2
SPI0	NPCS0	PIOA5
PIOD	NANDCS	PIOD15
DBGU	DRXD	PIOC30
DBGU	DTXD	PIOC31

14. Reset Controller (RSTC)

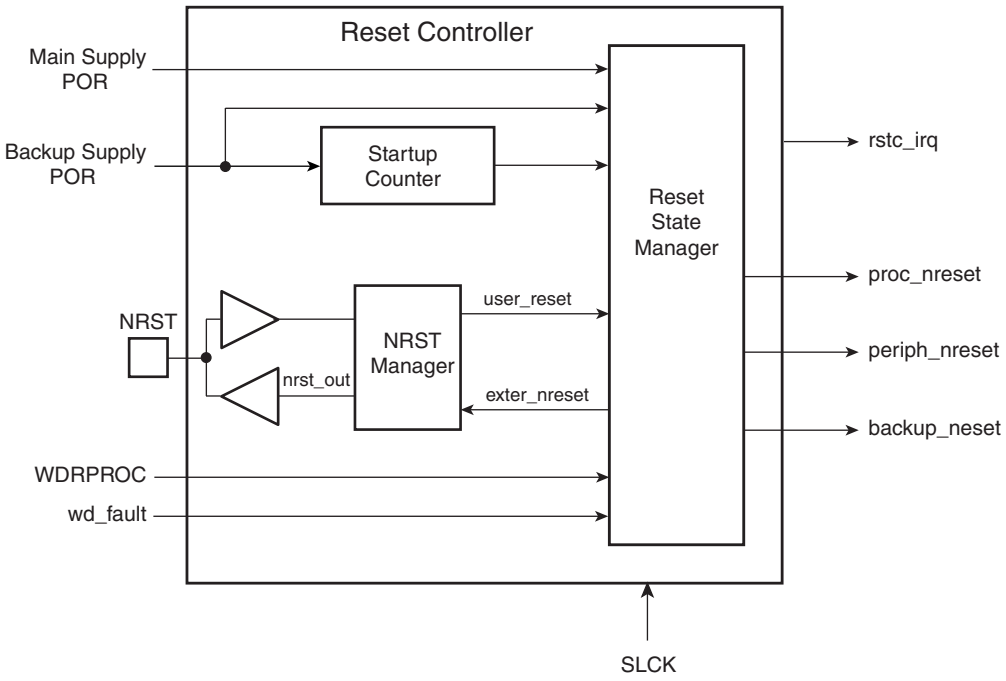
14.1 Overview

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

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

14.2 Block Diagram

Figure 14-1. Reset Controller Block Diagram



14.3 Functional Description

14.3.1 Reset Controller Overview

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

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

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

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

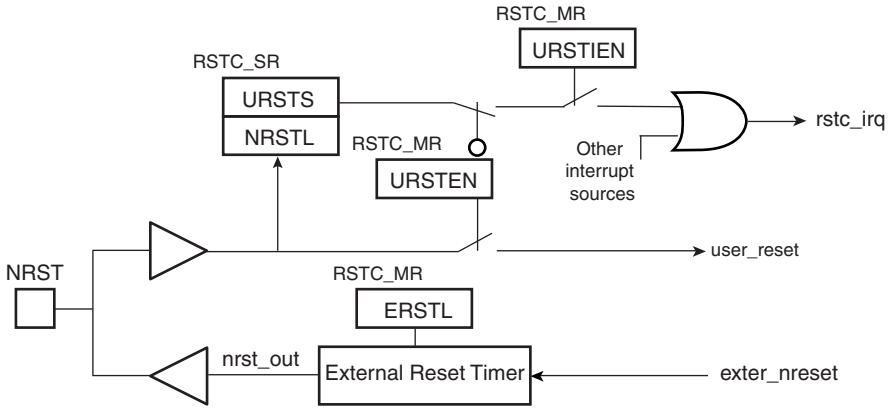
The startup counter waits for the complete crystal oscillator startup. The wait delay is given by the crystal oscillator startup time maximum value that can be found in the section Crystal Oscillator Characteristics in the Electrical Characteristics section of the product documentation.

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

14.3.2 NRST Manager

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

Figure 14-2. NRST Manager



14.3.2.1 NRST Signal or Interrupt

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

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

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

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

14.3.2.2 NRST External Reset Control

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

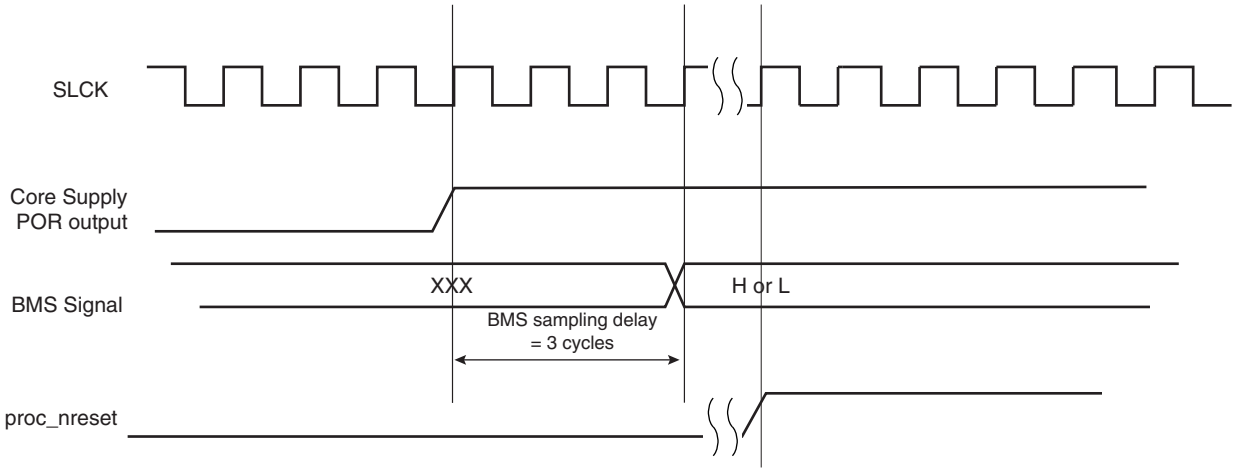
This feature allows the Reset Controller to shape the NRST pin level, and thus to guarantee that the NRST line is driven low for a time compliant with potential external devices connected on the system reset.

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

14.3.3 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 14-3. BMS Sampling



14.3.4 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.

14.3.4.1 General Reset

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

After this time, the processor clock is released at Slow Clock and all the other signals remain valid for 3 cycles for proper processor and logic reset. Then, all the reset signals are released and the field RSTTYP in RSTC_SR reports a General Reset. As the RSTC_MR is reset, the NRST line rises 2 cycles after the backup_nreset, as ERSTL defaults at value 0x0.

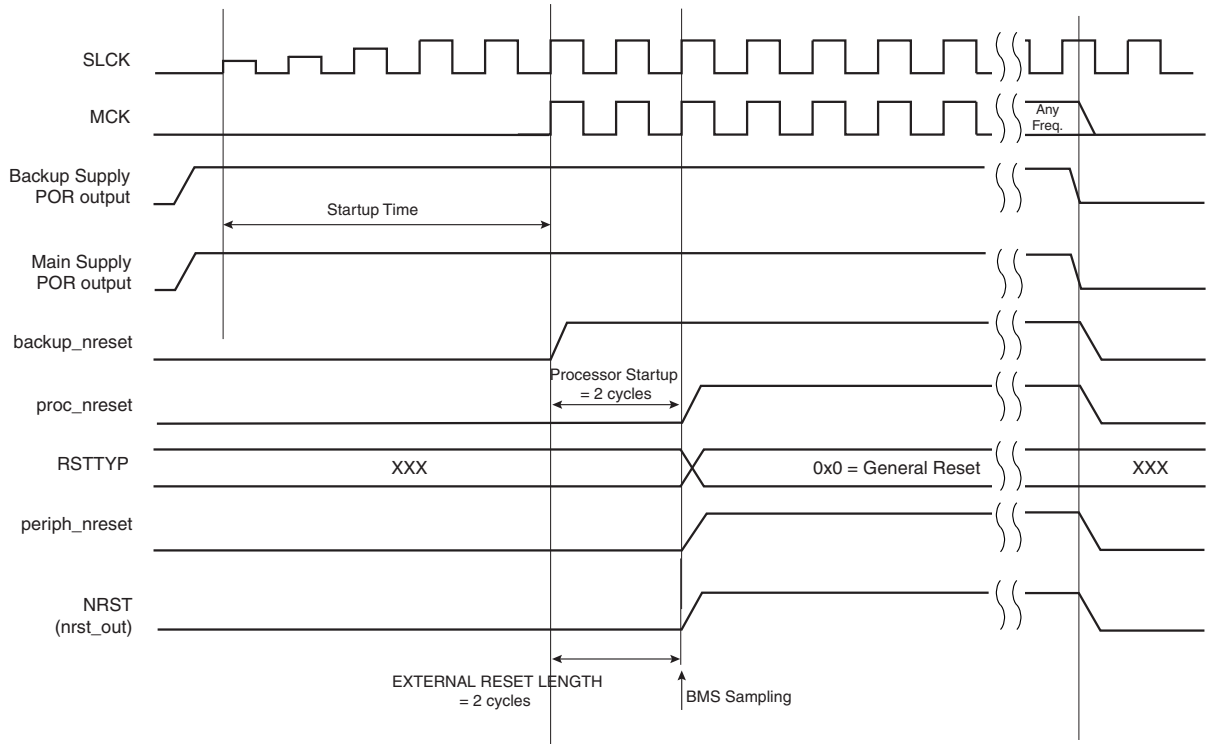
When VDDBU is detected low by the Backup Supply POR Cell, all resets signals are immediately asserted, even if the Main Supply POR Cell does not report a Main Supply shutdown.

VDDBU only activates the backup_nreset signal.

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

Figure 14-4 shows how the General Reset affects the reset signals.

Figure 14-4. General Reset State



14.3.4.2 *Wake-up Reset*

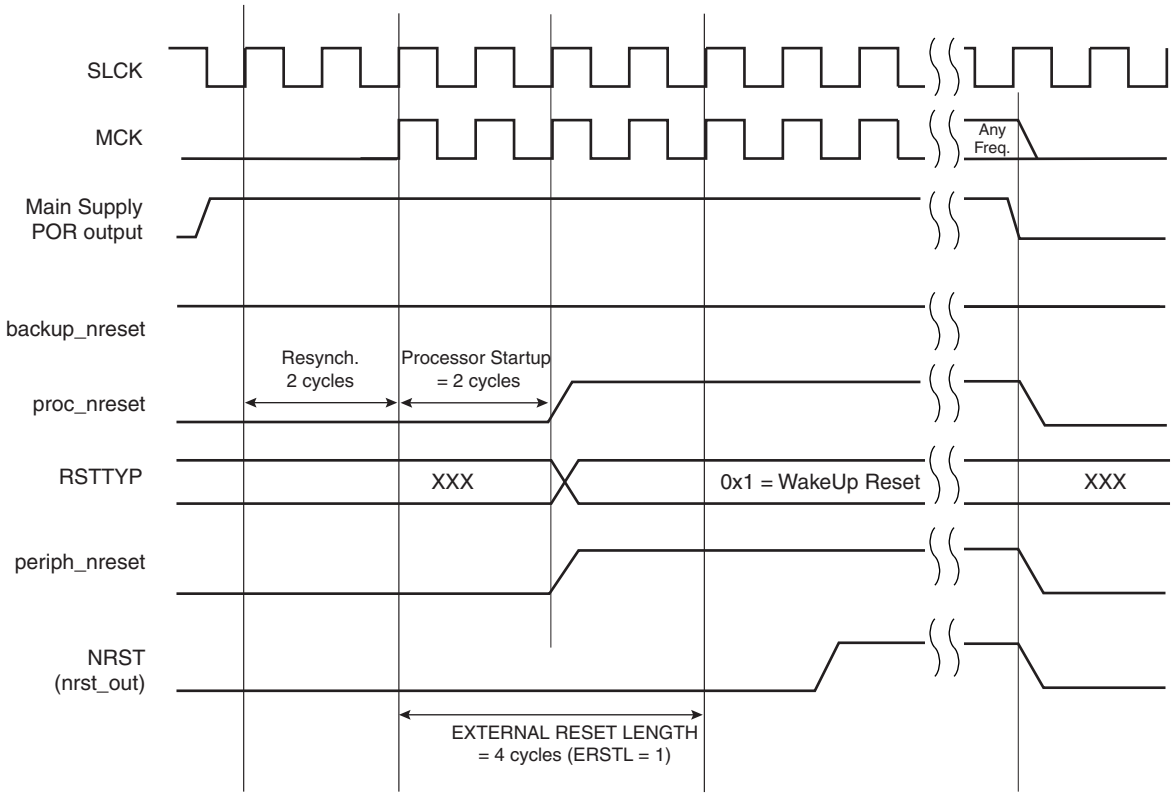
The Wake-up Reset occurs when the Main Supply is down. When the Main Supply POR output is active, all the reset signals are asserted except backup_nreset. When the Main Supply powers up, the POR output is resynchronized on Slow Clock. The processor clock is then re-enabled during 3 Slow Clock cycles, depending on the requirements of the ARM processor.

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

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

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

Figure 14-5. Wake-up State



14.3.4.3 User Reset

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

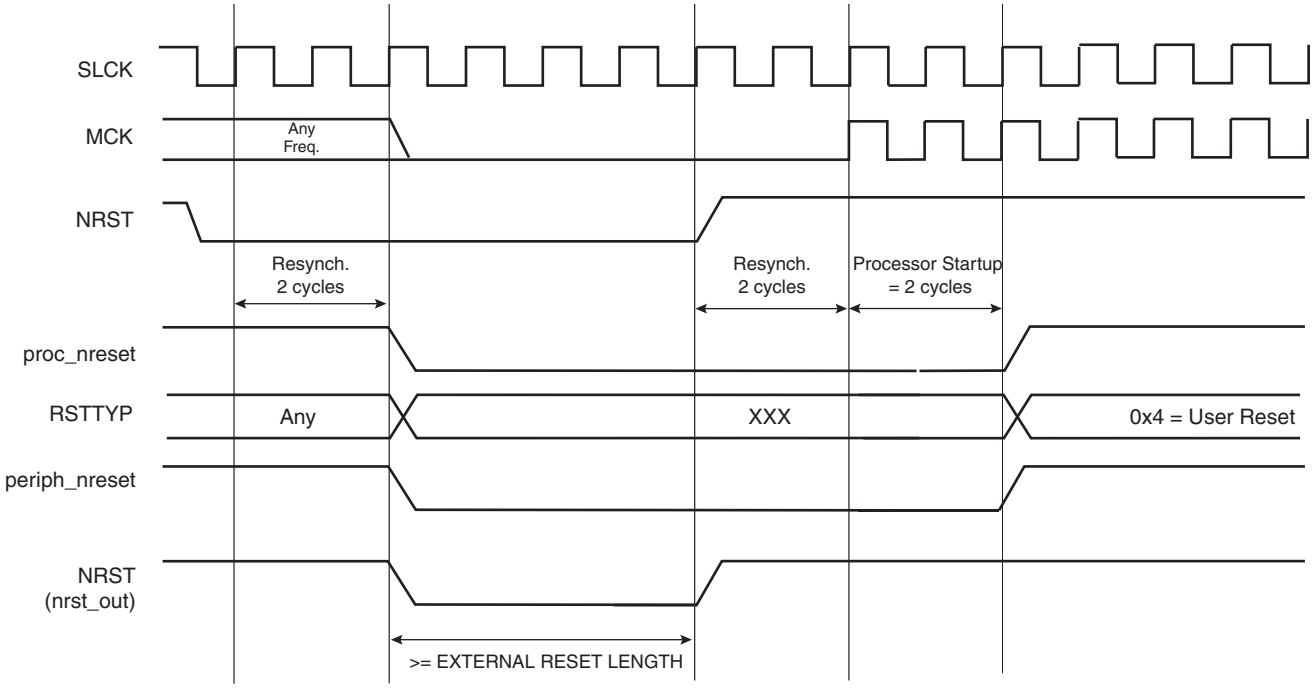
The User Reset is entered as soon as a low level is detected on NRST. The Processor Reset and the Peripheral Reset are asserted.

The User Reset is left when NRST rises, after a two-cycle resynchronization time and a 3-cycle processor startup. The processor clock is re-enabled as soon as NRST is confirmed high.

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

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

Figure 14-6. User Reset State



14.3.4.4 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 3 Slow Clock cycles.

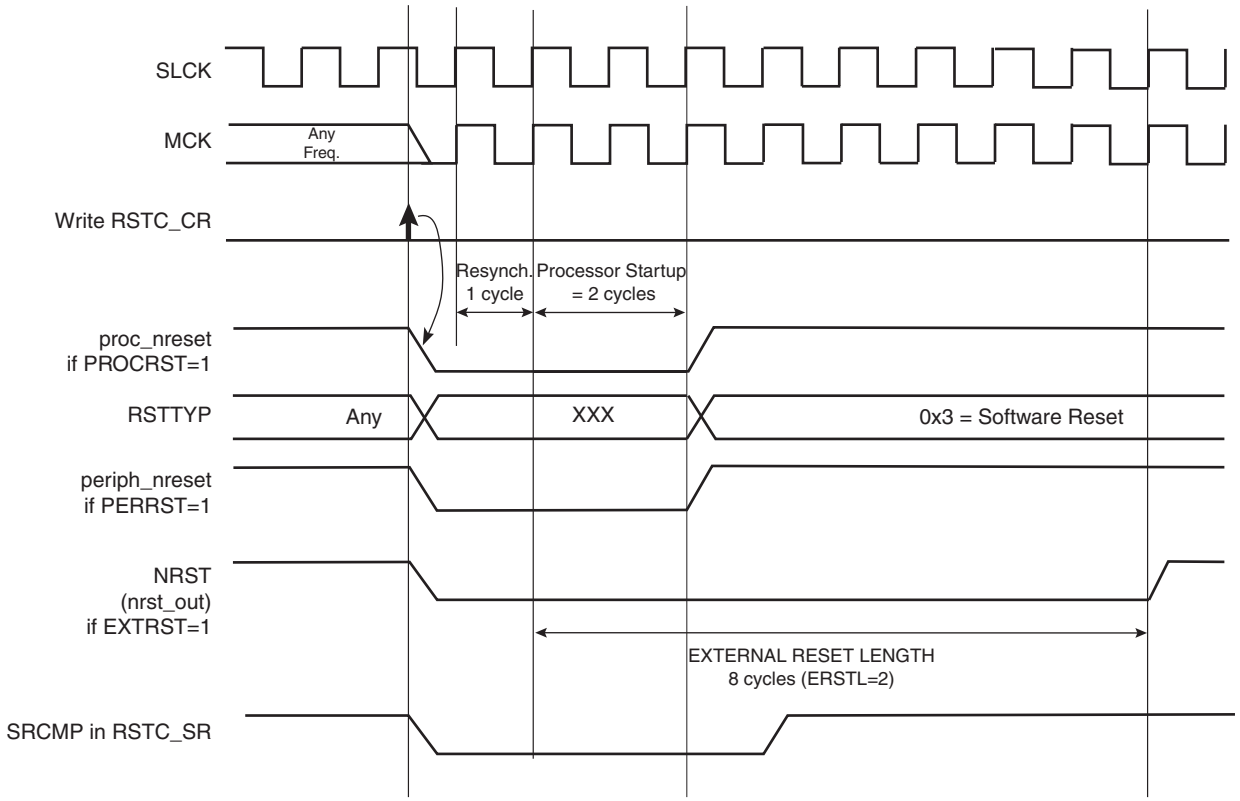
The internal reset signals are asserted as soon as the register write is performed. This is detected on the Master Clock (MCK). They are released when the software reset is left, i.e.; synchronously to SLCK.

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

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

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

Figure 14-7. Software Reset



14.3.4.5 *Watchdog Reset*

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

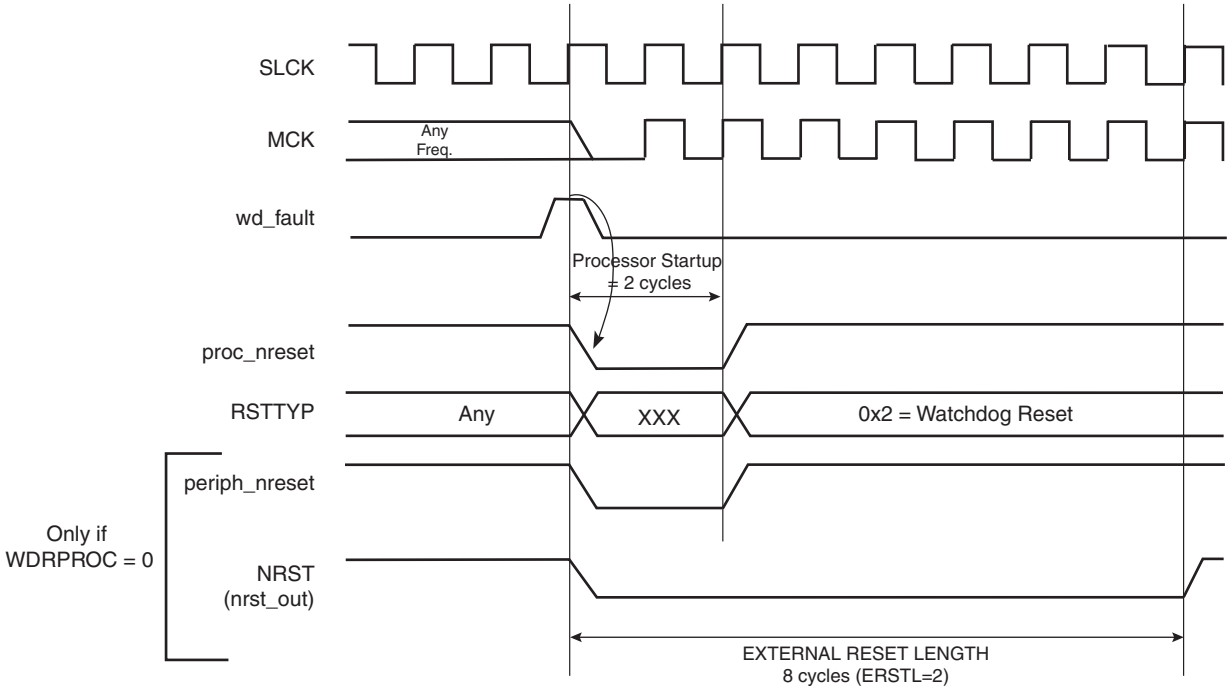
When in Watchdog Reset, assertion of the reset signals depends on the WDRPROC bit in WDT_MR:

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

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

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

Figure 14-8. Watchdog Reset



14.3.5 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:

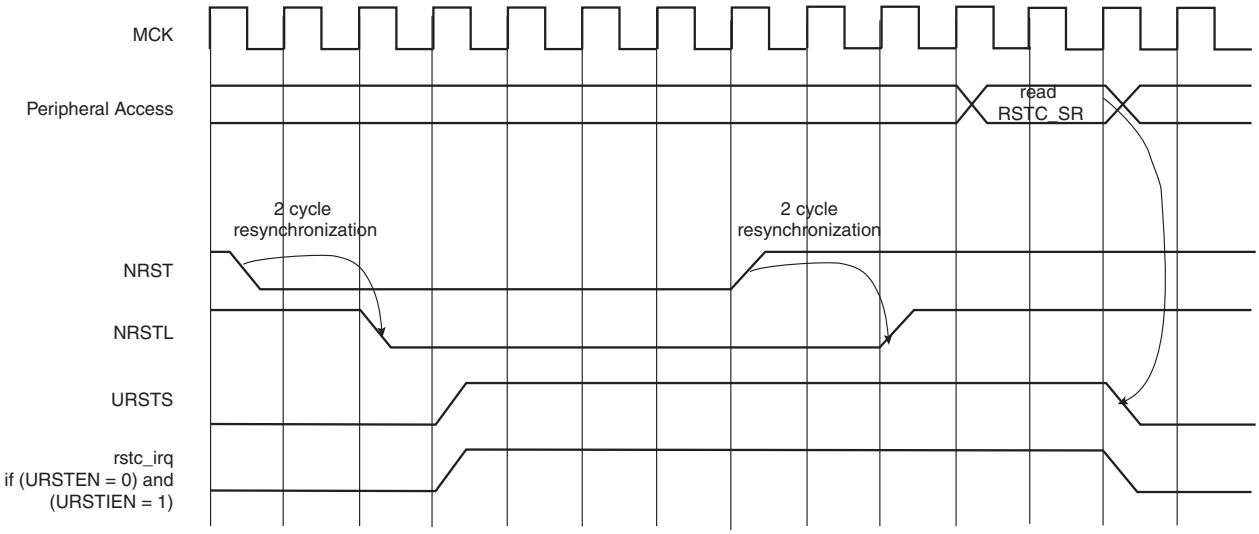
- When in User Reset:
  - A watchdog event is impossible because the Watchdog Timer is being reset by the proc_nreset signal.
  - A software reset is impossible, since the processor reset is being activated.
- When in Software Reset:
  - A watchdog event has priority over the current state.
  - The NRST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

14.3.6 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 14-9). 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 14-9. Reset Controller Status and Interrupt



14.4 Reset Controller (RSTC) User Interface

Table 14-1. 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.

14.4.1 Reset Controller Control Register

Name: RSTC_CR

Address: 0xFFFFFD00

Access Type: Write-only

31	30	29	28	27	26	25	24
KEY							

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–		–

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

- **PROCRST: Processor Reset**

0 = No effect.

1 = If KEY is correct, resets the processor.

- **PERRST: Peripheral Reset**

0 = No effect.

1 = If KEY is correct, resets the peripherals.

- **EXTRST: External Reset**

0 = No effect.

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

- **KEY: Password**

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

14.4.2 Reset Controller Status Register

Name: RSTC_SR

Address: 0xFFFFFD04

Access Type: Read-only

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

- **URSTS: User Reset Status**

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

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

- **RSTTYP: Reset Type**

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

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

- **NRSTL: NRST Pin Level**

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

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

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

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

14.4.3 Reset Controller Mode Register

Name: RSTC_MR

Address: 0xFFFFFD08

Access Type: Read-write

31	30	29	28	27	26	25	24
KEY							

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

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

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

- **URSTEN: User Reset Enable**

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

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

- **URSTIEN: User Reset Interrupt Enable**

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

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

- **ERSTL: External Reset Length**

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

- **KEY: Password**

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

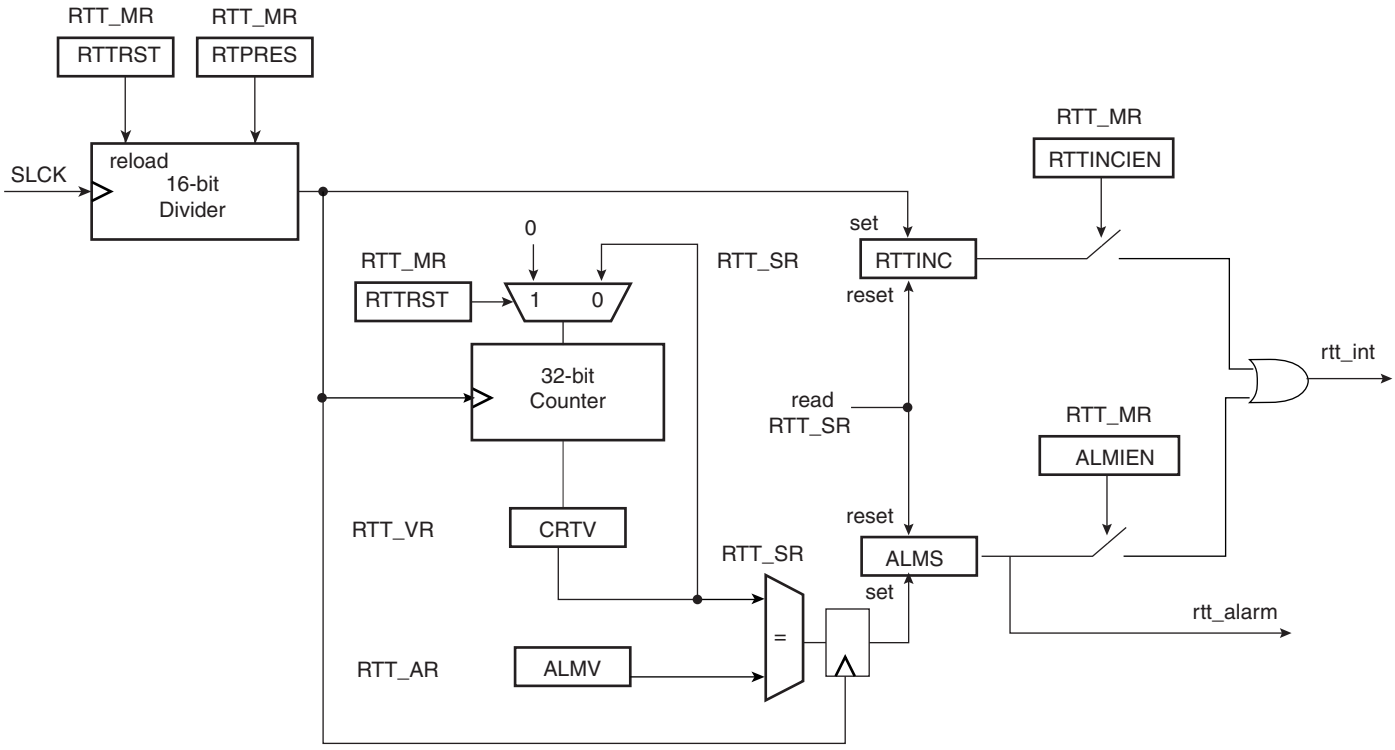
15. Real-Time Timer (RTT)

15.1 Description

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

15.2 Block Diagram

Figure 15-1. Real-time Timer



15.3 Functional Description

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

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

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

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

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

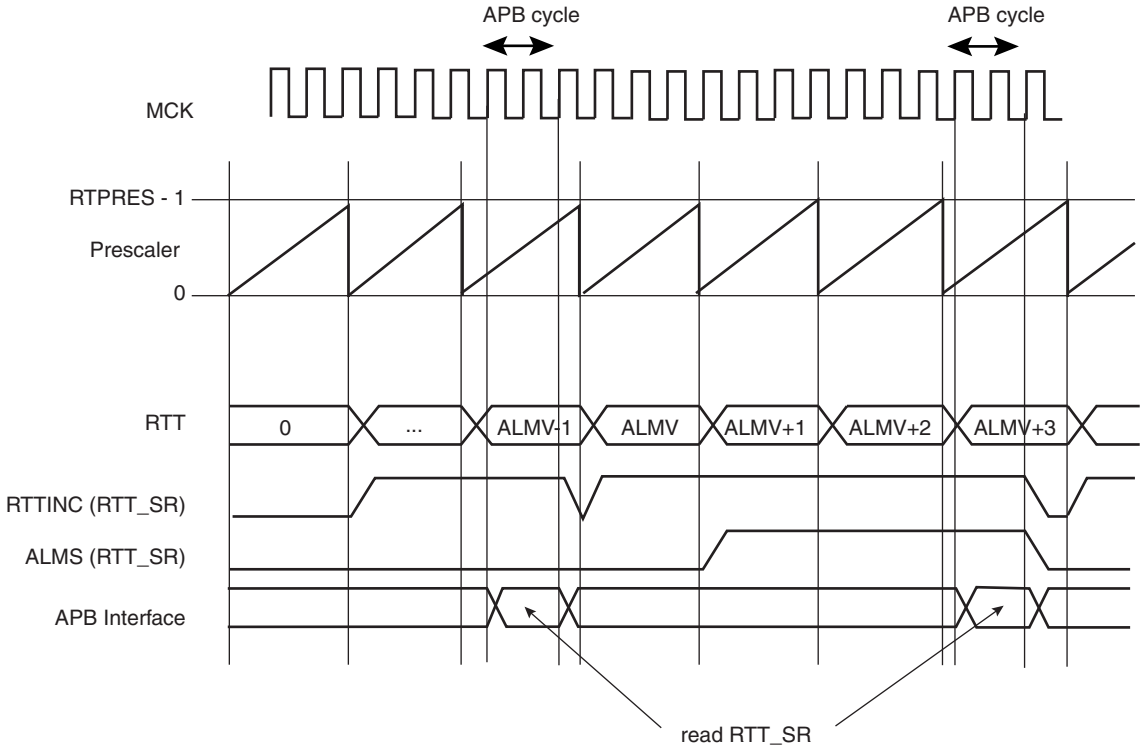
The bit RTTINC in RTT_SR is set each time the Real-time Timer counter is incremented. This bit can be used to start a periodic interrupt, the period being one second when the RTPRES is programmed with 0x8000 and Slow Clock equal to 32.768 Hz.

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

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

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

Figure 15-2. RTT Counting



15.4 Real-time Timer (RTT) User Interface

Table 15-1. 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

15.4.1 Real-time Timer Mode Register

Register Name: RTT_MR

Addresses: 0xFFFFFD20 (0), 0xFFFFFD50 (1)

Access Type: Read/Write

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

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

15	14	13	12	11	10	9	8
RTPRES							

7	6	5	4	3	2	1	0
RTPRES							

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

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

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

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

- **ALMIEN: Alarm Interrupt Enable**

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

1 = The bit ALMS in RTT_SR asserts interrupt.

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

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

1 = The bit RTTINC in RTT_SR asserts interrupt.

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

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

15.4.2 Real-time Timer Alarm Register

Register Name: RTT_AR

Addresses: 0xFFFFFD24 (0), 0xFFFFFD54 (1)

Access Type: Read/Write

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

- **ALMV: Alarm Value**

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

15.4.3 Real-time Timer Value Register

Register Name: RTT_VR

Addresses: 0xFFFFFD28 (0), 0xFFFFFD58 (1)

Access Type: Read-only

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

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

Returns the current value of the Real-time Timer.

15.4.4 Real-time Timer Status Register

Register Name: RTT_SR

Addresses: 0xFFFFFD2C (0), 0xFFFFFD5C (1)

Access Type: Read-only

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

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

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

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

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

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

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

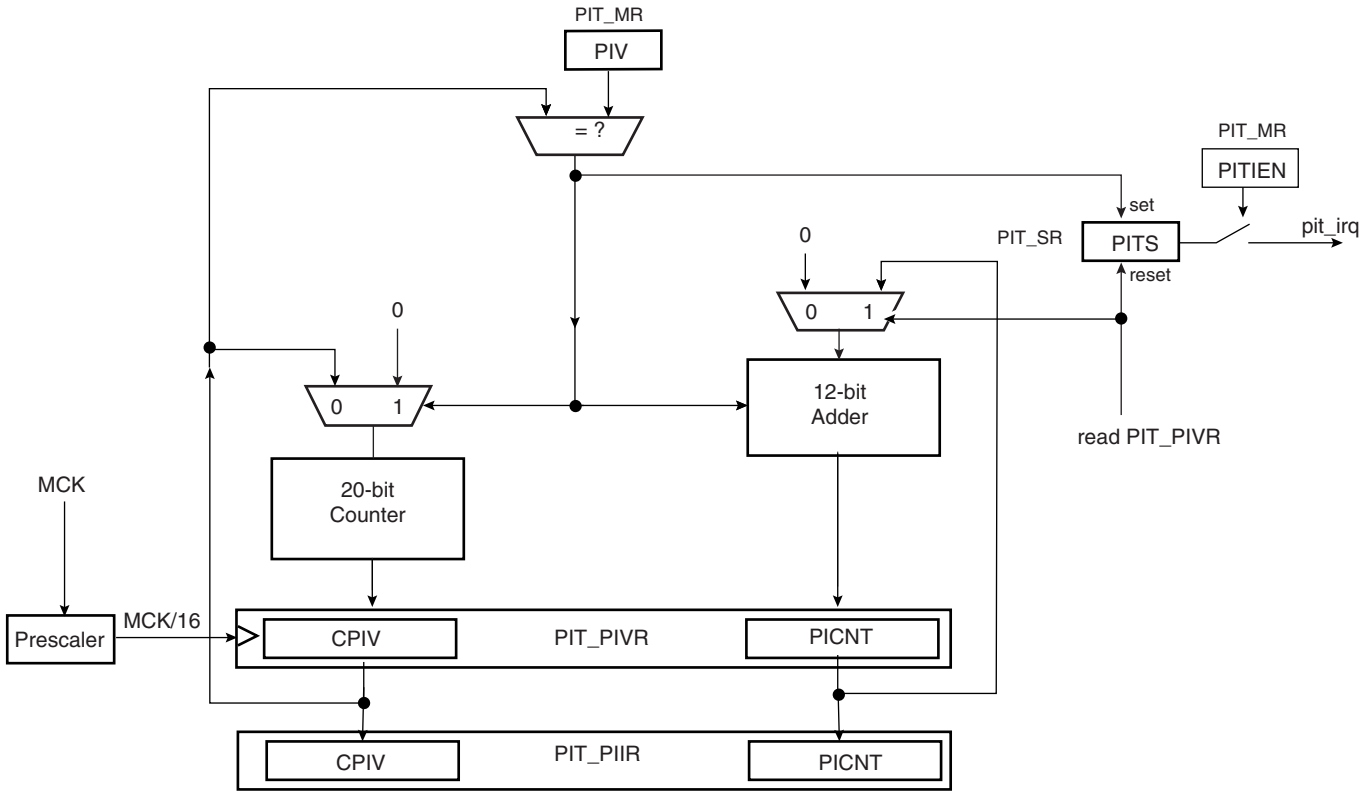
16. Periodic Interval Timer (PIT)

16.1 Overview

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

16.2 Block Diagram

Figure 16-1. Periodic Interval Timer



16.3 Functional Description

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

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

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

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

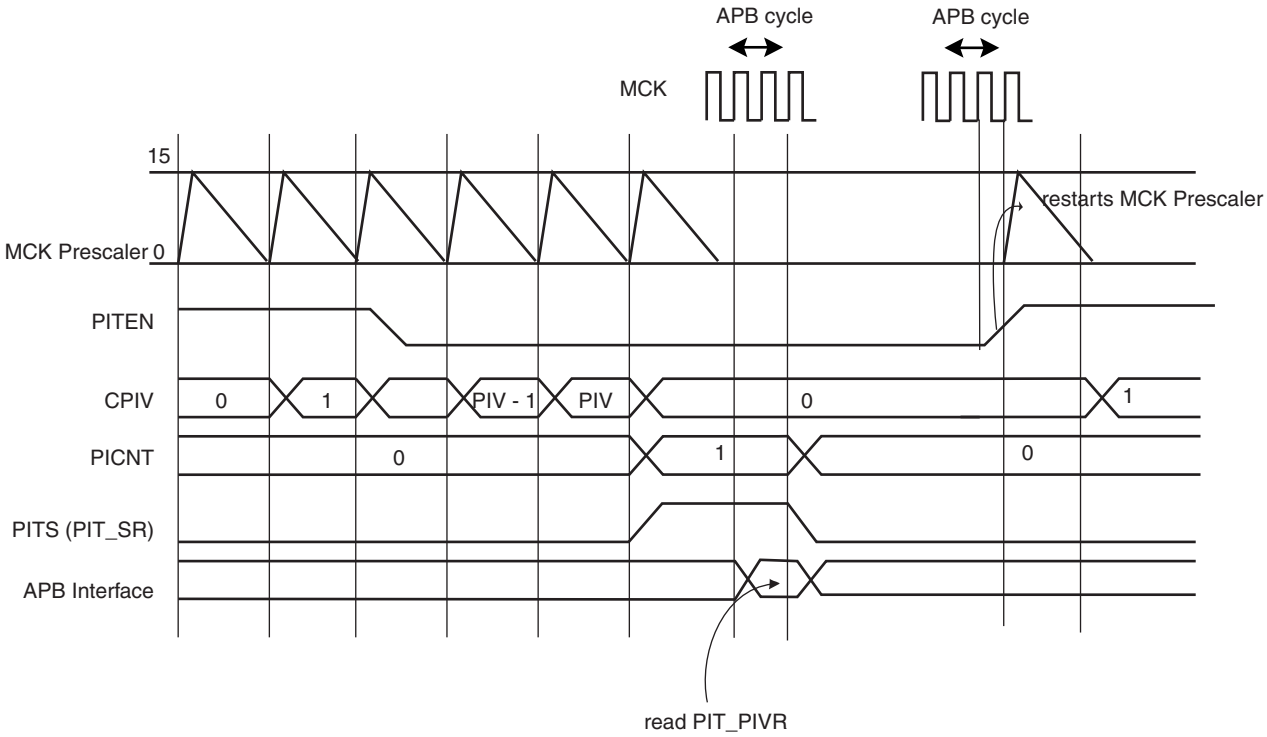
When CPIV and PICNT values are obtained by reading the Periodic Interval Value Register (PIT_PIVR), the overflow counter (PICNT) is reset and the PITS is cleared, thus acknowledging the interrupt. The value of PICNT gives the number of periodic intervals elapsed since the last read of PIT_PIVR.

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

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

The PIT is stopped when the core enters debug state.

Figure 16-2. Enabling/Disabling PIT with PITEN



16.4 Periodic Interval Timer (PIT) User Interface

Table 16-1. 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

16.4.1 Periodic Interval Timer Mode Register

Register Name: PIT_MR

Address: 0xFFFFFD30

Access Type: Read/Write

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

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

15	14	13	12	11	10	9	8
PIV							

7	6	5	4	3	2	1	0
PIV							

- **PIV: Periodic Interval Value**

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

- **PITEN: Period Interval Timer Enabled**

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

1 = The Periodic Interval Timer is enabled.

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

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

1 = The bit PITS in PIT_SR asserts interrupt.

16.4.2 Periodic Interval Timer Status Register

Register Name: PIT_SR

Address: 0xFFFFFD34

Access Type: Read-only

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

- **PITS: Periodic Interval Timer Status**

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

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

16.4.3 Periodic Interval Timer Value Register

Register Name: PIT_PIVR

Address: 0xFFFFFD38

Access Type: Read-only

31	30	29	28	27	26	25	24
PICNT							

23	22	21	20	19	18	17	16
PICNT				CPIV			

15	14	13	12	11	10	9	8
CPIV							

7	6	5	4	3	2	1	0
CPIV							

Reading this register clears PITS in PIT_SR.

- **CPIV: Current Periodic Interval Value**

Returns the current value of the periodic interval timer.

- **PICNT: Periodic Interval Counter**

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

16.4.4 Periodic Interval Timer Image Register

Register Name: PIT_PIIR

Address: 0xFFFFFD3C

Access Type: Read-only

31	30	29	28	27	26	25	24
PICNT							

23	22	21	20	19	18	17	16
PICNT				CPIV			

15	14	13	12	11	10	9	8
CPIV							

7	6	5	4	3	2	1	0
CPIV							

- **CPIV: Current Periodic Interval Value**

Returns the current value of the periodic interval timer.

- **PICNT: Periodic Interval Counter**

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

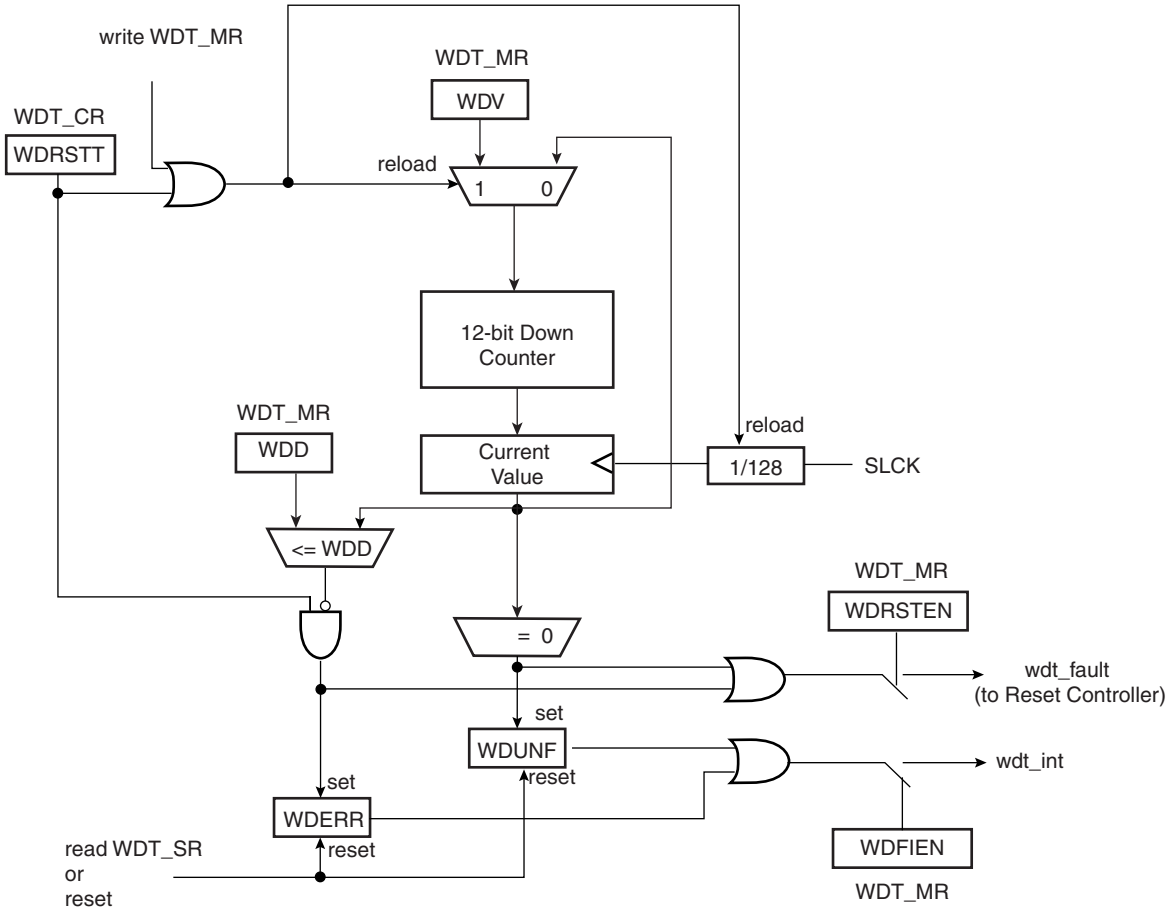
17. Watchdog Timer (WDT)

17.1 Overview

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

17.2 Block Diagram

Figure 17-1. Watchdog Timer Block Diagram



17.3 Functional Description

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

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

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

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

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

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

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

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

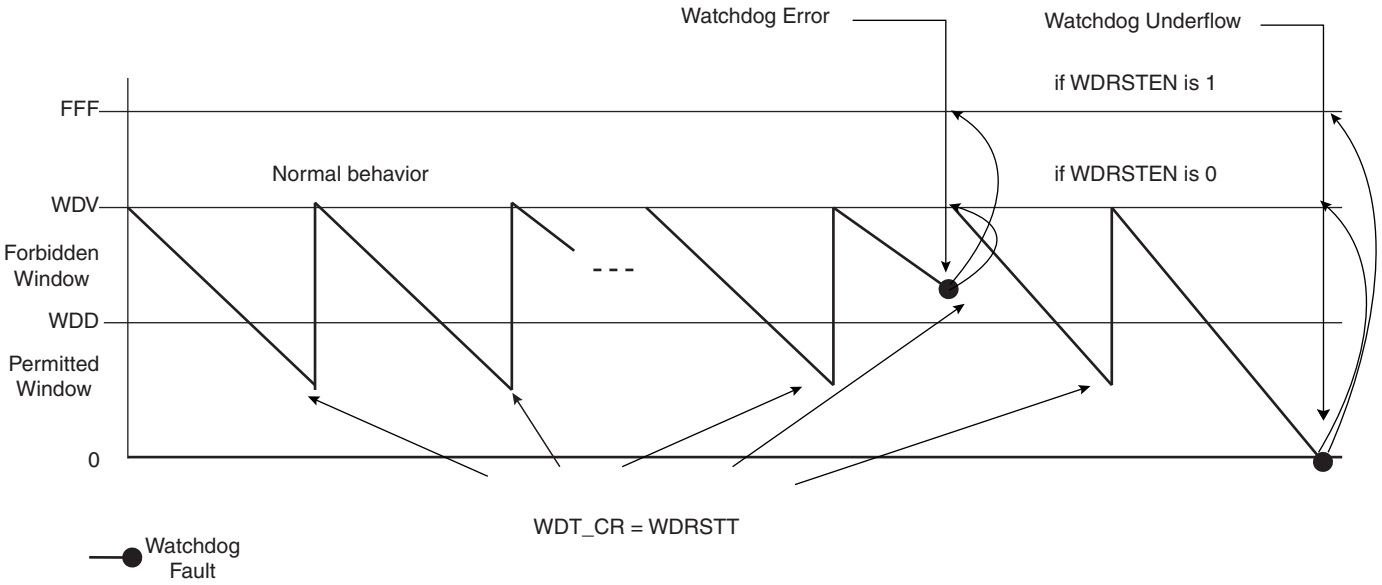
The status bits WDUNF (Watchdog Underflow) and WDERR (Watchdog Error) trigger an interrupt, provided the bit WDFIEN is set in the mode register. The signal “wdt_fault” to the reset controller causes a Watchdog reset if the WDRSTEN bit is set as already explained in the reset controller programmer Datasheet.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

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

Writing the WDT_MR reloads and restarts the down counter.

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

Figure 17-2. Watchdog Behavior



17.4 Watchdog Timer (WDT) User Interface

Table 17-1. 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

17.4.1 Watchdog Timer Control Register

Register Name: WDT_CR

Address: 0xFFFFFD40

Access Type: Write-only

31	30	29	28	27	26	25	24
KEY							

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WDRSTT

- **WDRSTT: Watchdog Restart**

0: No effect.

1: Restarts the Watchdog.

- **KEY: Password**

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

17.4.2 Watchdog Timer Mode Register

Register Name: WDT_MR

Address: 0xFFFFFD44

Access Type: Read-write Once

31	30	29	28	27	26	25	24
		WDIDLEHLT	WDDBGHLT	WDD			

23	22	21	20	19	18	17	16
WDD							

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

7	6	5	4	3	2	1	0
WDV							

- **WDV: Watchdog Counter Value**

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

- **WDFIEN: Watchdog Fault Interrupt Enable**

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

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

- **WDRSTEN: Watchdog Reset Enable**

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

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

- **WDRPROC: Watchdog Reset Processor**

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

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

- **WDD: Watchdog Delta Value**

Defines the permitted range for reloading the Watchdog Timer.

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

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

- **WDDBGHLT: Watchdog Debug Halt**

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

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

- **WDIDLEHLT: Watchdog Idle Halt**

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

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

- **WDDIS: Watchdog Disable**

0: Enables the Watchdog Timer.

1: Disables the Watchdog Timer.

17.4.3 Watchdog Timer Status Register

Register Name: WDT_SR

Address: 0xFFFFFD48

Access Type: Read-only

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

- **WDUNF: Watchdog Underflow**

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

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

- **WDERR: Watchdog Error**

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

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

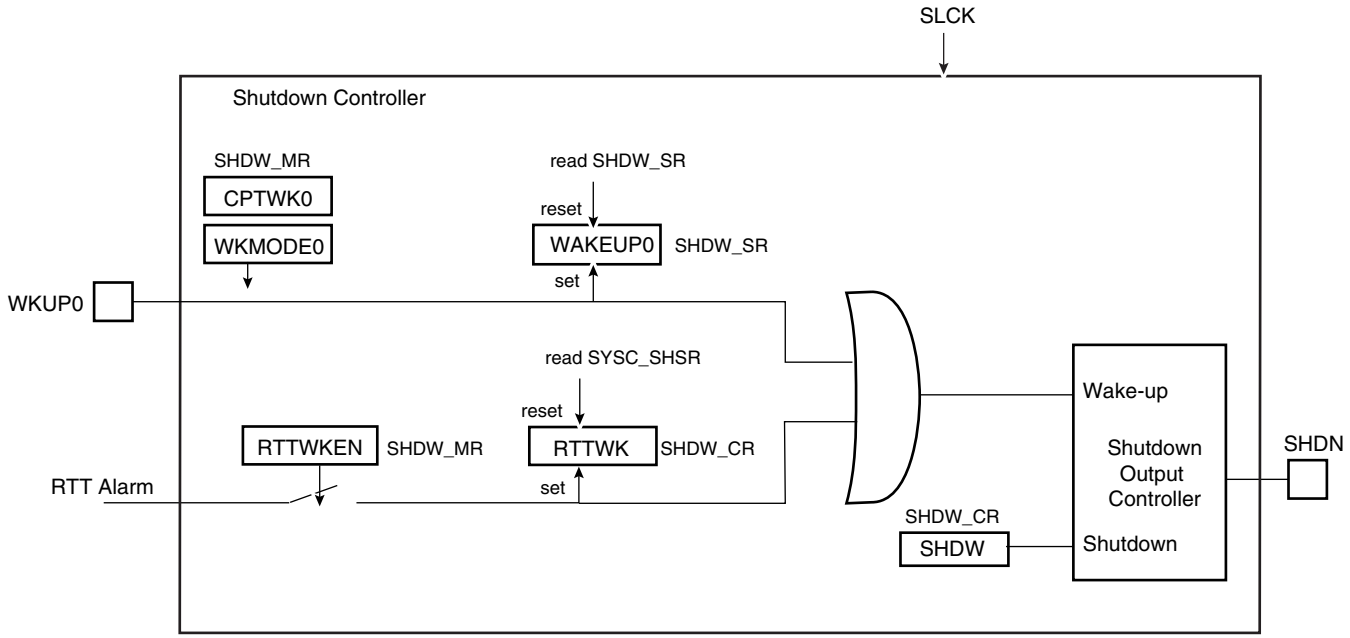
18. Shutdown Controller (SHDWC)

18.1 Overview

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

18.2 Block Diagram

Figure 18-1. Shutdown Controller Block Diagram



18.3 I/O Lines Description

Table 18-1. I/O Lines Description

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

18.4 Product Dependencies

18.4.1 Power Management

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

18.5 Functional Description

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

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

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

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

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

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

18.6 Shutdown Controller (SHDWC) User Interface

Table 18-2. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Shutdown Control Register	SHDW_CR	Write-only	-
0x04	Shutdown Mode Register	SHDW_MR	Read-write	0x0000_0103
0x08	Shutdown Status Register	SHDW_SR	Read-only	0x0000_0000

18.6.1 Shutdown Control Register

Register Name: SHDW_CR

Address: 0xFFFFFD10

Access Type: Write-only

31	30	29	28	27	26	25	24
KEY							

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	SHDW

- **SHDW: Shutdown Command**

0 = No effect.

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

- **KEY: Password**

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

18.6.2 Shutdown Mode Register

Register Name: SHDW_MR

Address: 0xFFFFFD14

Access Type: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	RTTWKEN
15	14	13	12	11	10	9	8
–				–	–	–	
7	6	5	4	3	2	1	0
CPTWK0				–	–	WKMODE0	

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

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

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

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

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

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

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

18.6.3 Shutdown Status Register

Register Name: SHDW_SR

Address: 0xFFFFFD18

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	RTTWK
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WAKEUP0

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

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

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

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

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

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

19. General Purpose Backup Registers (GPBR)

19.1 Overview

The System Controller embeds 20 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	–
...	...	...	...	...
0x4C	General Purpose Backup Register 19	SYS_GPBR 19	Read-write	–

19.2.1 General Purpose Backup Register x

Register Name:SYS_GPBRx

Addresses: 0xFFFFFD60 [0], 0xFFFFFD64 [1], 0xFFFFFD68 [2], 0xFFFFFD6C [3], 0xFFFFFD70 [4], 0xFFFFFD74 [5], 0xFFFFFD78 [6], 0xFFFFFD7C [7], 0xFFFFFD80 [8], 0xFFFFFD84 [9], 0xFFFFFD88 [10], 0xFFFFFD8C [11], 0xFFFFFD90 [12], 0xFFFFFD94 [13], 0xFFFFFD98 [14], 0xFFFFFD9C [15], 0xFFFFFDA0 [16], 0xFFFFFDA4 [17], 0xFFFFFDA8 [18], 0xFFFFFDAC [19]

Access Type: 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. AT91SAM9263 Bus Matrix

20.1 Description

Bus Matrix implements a multi-layer AHB, based on AHB-Lite protocol, that enables parallel access paths between multiple AHB masters and slaves in a system, which increases the overall bandwidth. Bus Matrix interconnects 9 AHB Masters to 7 AHB Slaves. The normal latency to connect a master to a slave is one cycle except for the default master of the accessed slave which is connected directly (zero cycle latency).

The Bus Matrix user interface is compliant with ARM Advanced Peripheral Bus and provides a Chip Configuration User Interface with Registers that allow the Bus Matrix to support application specific features.

20.2 Memory Mapping

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

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

20.3 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.

20.3.1 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.

20.3.2 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.

20.3.3 Fixed Default Master

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

To change from one kind of default master to another, the Bus Matrix user interface provides the Slave Configuration Registers, one for each slave, that allow to set a default master for each slave. The Slave Configuration Register contains two fields:

DEFMSTR_TYPE and FIXED_DEFMSTR. The 2-bit DEFMSTR_TYPE field allows to choose the default master type (no default, last access master, fixed default master) whereas the 4-bit

FIXED_DEFMSTR field allows to choose a fixed default master provided that DEFMSTR_TYPE is set to fixed default master. Please refer to the Bus Matrix user interface description.

20.4 Arbitration

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

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

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

This choice is given through the field ARBT of the Slave Configuration Registers (MATRIX_SCFG).

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

When a re-arbitration has to be done, it is realized only under some specific conditions detailed in the following paragraph.

20.4.1 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 20.4.1.1 "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 20.4.1.2 "Slot Cycle Limit Arbitration".)

20.4.1.1 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).

20.4.1.2 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.

20.4.2 Round-Robin Arbitration

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

There are three round-robin algorithm implemented:

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

20.4.2.1 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.

20.4.2.2 Round-Robin Arbitration with Last Access Master

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

20.4.2.3 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.

20.4.3 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).

20.5 Bus Matrix (MATRIX) User Interface

Table 20-1. 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 - 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
0x0060 - 0x007C	Reserved	–	–	–
0x0080	Priority Register A for Slave 0	MATRIX_PRAS0	Write-only	0x00000000
0x0084	Priority Register B for Slave 0	MATRIX_PRBS0	Write-only	0x00000000
0x0088	Priority Register A for Slave 1	MATRIX_PRAS1	Write-only	0x00000000
0x008C	Priority Register B for Slave 1	MATRIX_PRBS1	Write-only	0x00000000
0x0090	Priority Register A for Slave 2	MATRIX_PRAS2	Write-only	0x00000000
0x0094	Priority Register B for Slave 2	MATRIX_PRBS2	Write-only	0x00000000
0x0098	Priority Register A for Slave 3	MATRIX_PRAS3	Write-only	0x00000000
0x009C	Priority Register B for Slave 3	MATRIX_PRBS3	Write-only	0x00000000
0x00A0	Priority Register A for Slave 4	MATRIX_PRAS4	Write-only	0x00000000
0x00A4	Priority Register B for Slave 4	MATRIX_PRBS4	Write-only	0x00000000
0x00A8	Priority Register A for Slave 5	MATRIX_PRAS5	Write-only	0x00000000
0x00AC	Priority Register B for Slave 5	MATRIX_PRBS5	Write-only	0x00000000
0x00B0	Priority Register A for Slave 6	MATRIX_PRAS6	Write-only	0x00000000
0x00B4	Priority Register B for Slave 6	MATRIX_PRBS6	Write-only	0x00000000
0x00C0 - 0x00FC	Reserved	–	–	–
0x0100	Master Remap Control Register	MATRIX_MRCR	Read-write	0x00000000
0x0104 - 0x010C	Reserved	–	–	–

20.5.1 Bus Matrix Master Configuration Registers

Register Name:MATRIX_MCFG0...MATRIX_MCFG8

Address: 0xFFFFEC00

Access Type: Read-write

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

- **ULBT: Undefined Length Burst Type**

0: Infinite Length Burst

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

1: Single Access

The undefined length burst is treated as a succession of single accesses, allowing 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.

20.5.2 Bus Matrix Slave Configuration Registers

Register Name: MATRIX_SCFG0...MATRIX_SCFG6

Address: 0xFFFFEC40

Access Type: Read-write

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

23	22	21	20	19	18	17	16
–		FIXED_DEFMSTR				DEFMSTR_TYPE	

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
SLOT_CYCLE							

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

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

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

Note that an unreasonably small value breaks every burst and the Bus Matrix then arbitrates without performing any data transfer. 16 cycles is a reasonable value for SLOT_CYCLE.

- **DEFMASTR_TYPE: Default Master Type**

0: No Default Master

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

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

1: Last Default Master

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

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

2: Fixed Default Master

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

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

- **FIXED_DEFMSTR: Fixed Default Master**

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

- **ARBT: Arbitration Type**

0: Round-Robin Arbitration

1: Fixed Priority Arbitration

2: Reserved

3: Reserved

20.5.3 Bus Matrix Priority Registers A For Slaves

Register Name:MATRIX_PRAS0...MATRIX_PRAS6

Addresses: 0xFFFFEC80 [0], 0xFFFFEC88 [1], 0xFFFFEC90 [2], 0xFFFFEC98 [3], 0xFFFFECA0 [4], 0xFFFFECA8 [5], 0xFFFFECB0 [6]

Access Type: Write-only

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.

20.5.4 Bus Matrix Priority Registers B For Slaves

Register Name:MATRIX_PRBS0...MATRIX_PRBS6

Addresses: 0xFFFFEC84 [0], 0xFFFFEC8C [1], 0xFFFFEC94 [2], 0xFFFFEC9C [3], 0xFFFFECA4 [4], 0xFFFFECAC [5], 0xFFFFECB4 [6]

Access Type: Write -only

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	M8PR	

- **M8PR: Master 8 Priority**

Fixed priority of Master 8 for accessing to the selected slave. The higher the number, the higher the priority.

20.5.5 Bus Matrix Master Remap Control Register

Register Name: MATRIX_MRCR

Address: 0xFFFFED00

Access Type: Read-write

Reset: 0x0000_0000

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

RCBx	Master
RCB0	ARM926 Instruction
RCB1	ARM926 Data
RCB2	Peripheral DMA Controller
RCB3	LCD Controller
RCB4	Ethernet EMAC
RCB5	DMA Controller
RCB6	Two D Graphic Controller
RCB7	Image Sensor Interface
RCB8	OHCI USB Host Controller

20.6 Chip Configuration User Interface

Table 20-2. Chip Configuration User Interface

Offset	Register	Name	Access	Reset
0x0110	Reserved	–	–	–
0x0114	Bus Matrix TCM Configuration Register	MATRIX_TCMR	Read-write	0x00000000
0x0118 - 0x011C	Reserved	–	–	–
0x0120	EBI0 Chip Select Assignment Register	EBI0_CSA	Read-write	0x00010000
0x0124	EBI1 Chip Select Assignment Register	EBI1_CSA	Read-write	0x00010000
0x0128 - 0x01FC	Reserved	–	–	–

20.6.1 Bus Matrix TCM Configuration Register

Register Name:MATRIX_TCR

Access Type: Read-write

Reset: 0x0000_0000

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
DTCM_SIZE				ITCM_SIZE			

- **ITCM_SIZE: Size of ITCM enabled memory block**

0000: 0 KB (No ITCM Memory)

0101: 16 KB

0110: 32 KB

Others: Reserved

- **DTCM_SIZE: Size of DTCM enabled memory block**

0000: 0 KB (No DTCM Memory)

0101: 16 KB

0110: 32 KB

Others: Reserved

20.6.2 EBI0 Chip Select Assignment Register

Register Name: EBI0_CSA

Access Type: Read-write

Reset: 0x0001_0000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	VDDIOMSEL
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	EBI0_DBPUC
7	6	5	4	3	2	1	0
–	–	EBI0_CS5A	EBI0_CS4A	EBI0_CS3A	–	EBI0_CS1A	–

- **EBI0_CS1A: EBI0 Chip Select 1 Assignment**

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

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

- **EBI0_CS3A: EBI0 Chip Select 3 Assignment**

0 = EBI0 Chip Select 3 is only assigned to the Static Memory Controller and EBI0_NCS3 behaves as defined by the SMC.

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

- **EBI0_CS4A: EBI0 Chip Select 4 Assignment**

0 = EBI0 Chip Select 4 is only assigned to the Static Memory Controller and EBI0_NCS4 behaves as defined by the SMC.

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

- **EBI0_CS5A: EBI0 Chip Select 5 Assignment**

0 = EBI0 Chip Select 5 is only assigned to the Static Memory Controller and EBI0_NCS5 behaves as defined by the SMC.

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

- **EBI0_DBPUC: EBI0 Data Bus Pull-Up Configuration**

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

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

- **VDDIOMSEL: Memory voltage selection**

0 = Memories are 1.8V powered.

1 = Memories are 3.3V powered.

20.6.3 EBI1 Chip Select Assignment Register

Register Name: EBI1_CSA

Access Type: Read-write

Reset: 0x0001_0000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	VDDIOMSEL
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	EBI1_DBPUC
7	6	5	4	3	2	1	0
–	–	–	–	EBI1_CS2A	–	EBI1_CS1A	–

- **EBI1_CS1A: EBI1 Chip Select 1 Assignment**

0 = EBI1 Chip Select 1 is assigned to the Static Memory Controller.

1 = EBI1 Chip Select 1 is assigned to the SDRAM Controller.

- **EBI1_CS2A: EBI1 Chip Select 2 Assignment**

0 = EBI1 Chip Select 2 is only assigned to the Static Memory Controller and EBI1_NCS2 behaves as defined by the SMC.

1 = EBI1 Chip Select 2 is assigned to the Static Memory Controller and the SmartMedia Logic is activated.

- **EBI1_DBPUC: EBI1 Data Bus Pull-Up Configuration**

0 = EBI1 D0 - D15 Data Bus bits are internally pulled-up to the VDDIOM1 power supply.

1 = EBI1 D0 - D15 Data Bus bits are not internally pulled-up.

- **VDDIOMSEL: Memory voltage selection**

0 = Memories are 1.8V powered.

1 = Memories are 3.3V powered.

21. External Bus Interface (EBI)

21.1 Overview

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

The EBI0 also supports the CompactFlash and the NAND Flash protocols via integrated circuitry that greatly reduces the requirements for external components. Furthermore, the EBI0 handles data transfers with up to six external devices, each assigned to six address spaces defined by the embedded Memory Controller. Data transfers are performed through a 16-bit or 32-bit data bus, an address bus of up to 26 bits, up to six chip select lines (NCS[5:0]) and several control pins that are generally multiplexed between the different external Memory Controllers.

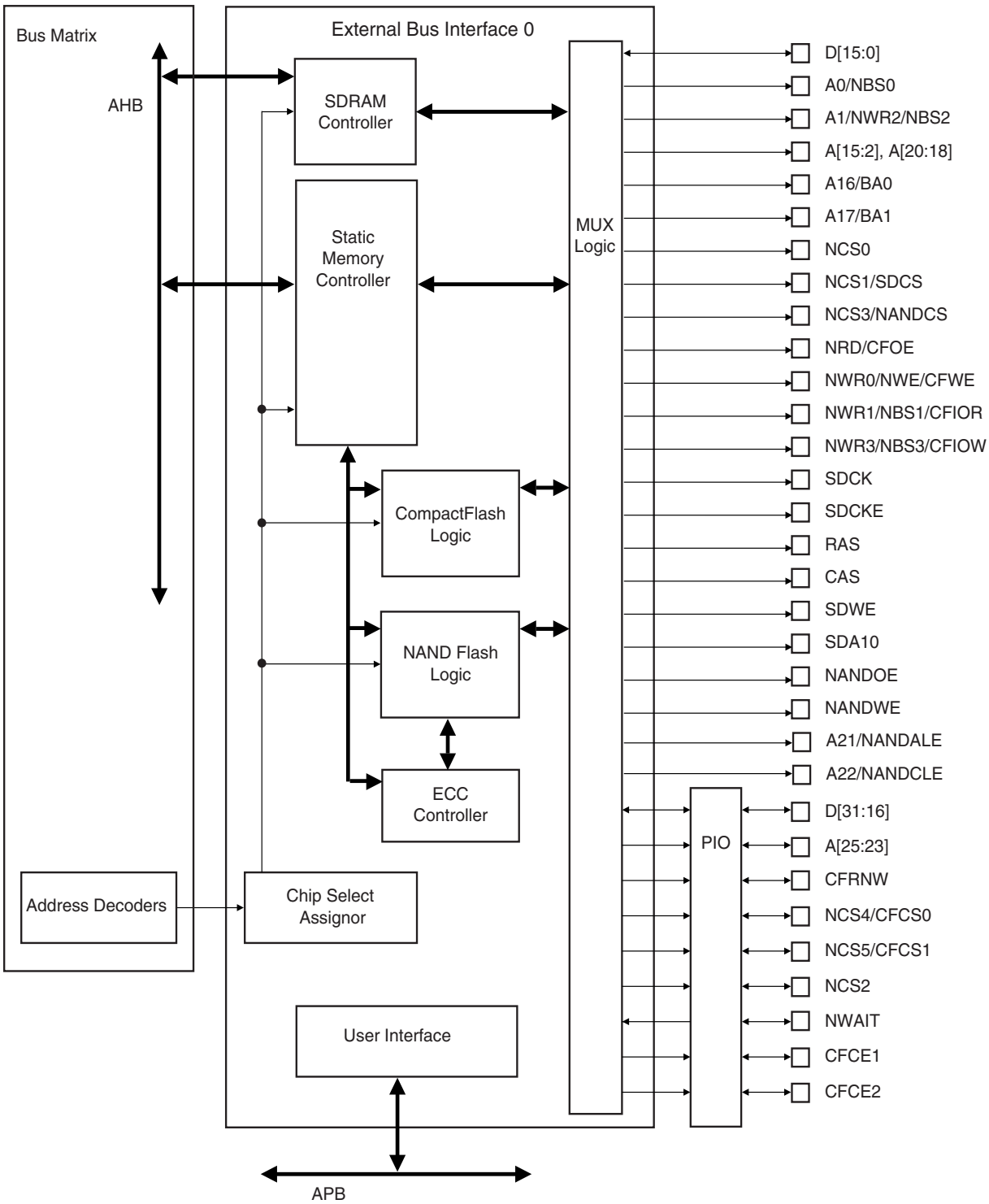
The EBI1 also supports the NAND Flash protocols via integrated circuitry that greatly reduces the requirements for external components. Furthermore, the EBI1 handles data transfers with up to three external devices, each assigned to three address spaces defined by the embedded Memory Controller. Data transfers are performed through a 16-bit or 32-bit data bus, an address bus of up to 23 bits, up to three chip select lines (NCS[2:0]) and several control pins that are generally multiplexed between the different external Memory Controllers.

21.2 Block Diagram

21.2.1 External Bus Interface 0

Figure 21-1 shows the organization of the External Bus Interface 0.

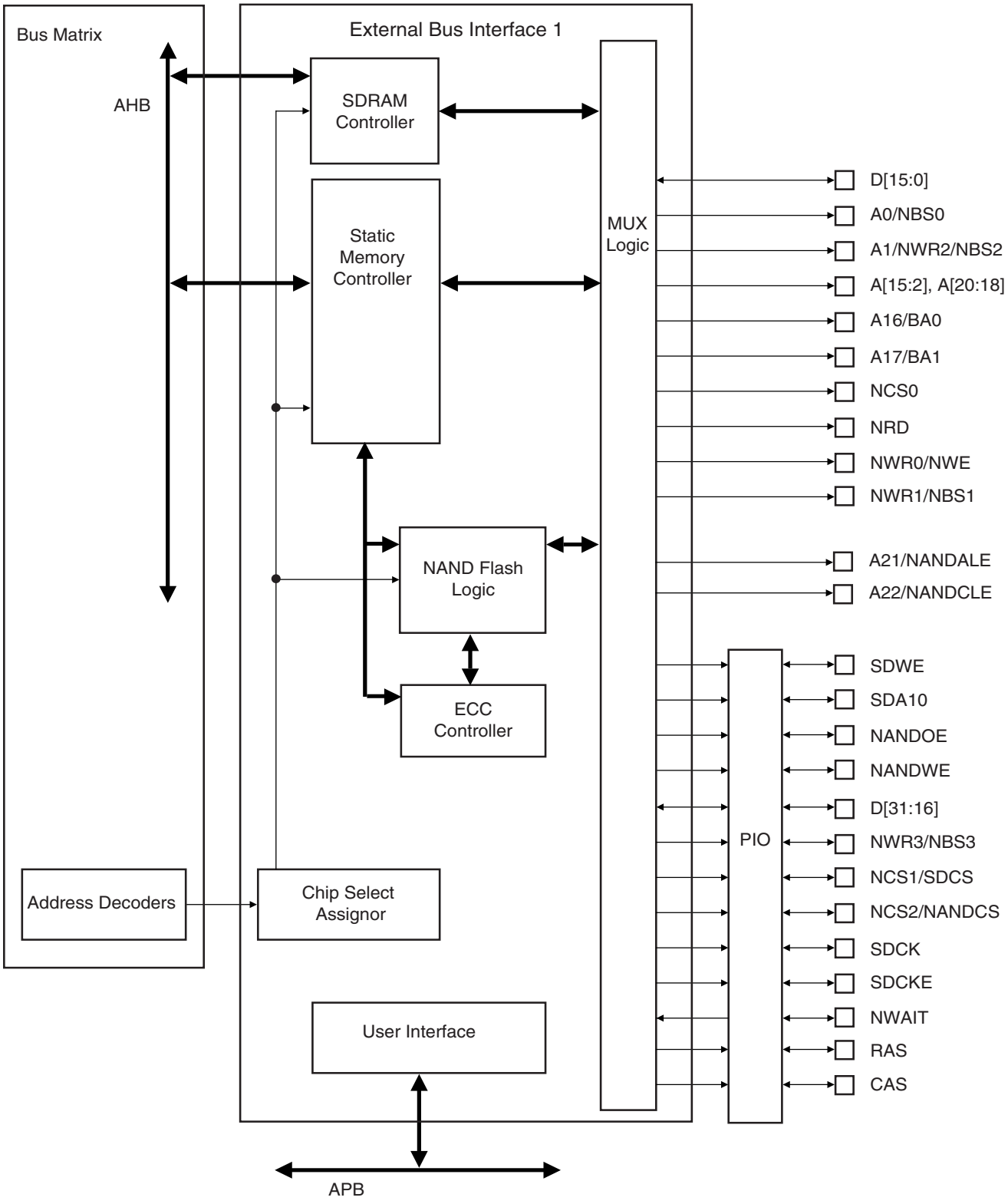
Figure 21-1. Organization of the External Bus Interface 0



21.2.2 External Bus Interface 1

Figure 21-2 shows the organization of the External Bus Interface 1.

Figure 21-2. Organization of the External Bus Interface 1



21.3 I/O Lines Description

Table 21-1. EBI0 I/O Lines Description

Name	Function	Type	Active Level
EBI			
EBI0_D0 - EBI0_D31	Data Bus	I/O	
EBI0_A0 - EBI0_A25	Address Bus	Output	
EBI0_NWAIT	External Wait Signal	Input	Low
SMC			
EBI0_NCS0 - EBI0_NCS5	Chip Select Lines	Output	Low
EBI0_NWR0 - EBI0_NWR3	Write Signals	Output	Low
EBI0_NRD	Read Signal	Output	Low
EBI0_NWE	Write Enable	Output	Low
EBI0_NBS0 - EBI0_NBS3	Byte Mask Signals	Output	Low
EBI for CompactFlash Support			
EBI0_CFCE1 - EBI0_CFCE2	CompactFlash Chip Enable	Output	Low
EBI0_CFOE	CompactFlash Output Enable	Output	Low
EBI0_CFWE	CompactFlash Write Enable	Output	Low
EBI0_CFIOR	CompactFlash I/O Read Signal	Output	Low
EBI0_CFIOW	CompactFlash I/O Write Signal	Output	Low
EBI0_CFRNW	CompactFlash Read Not Write Signal	Output	
EBI0_CFCS0 - EBI0_CFCS1	CompactFlash Chip Select Lines	Output	Low
EBI for NAND Flash Support			
EBI0_NANDCS	NAND Flash Chip Select Line	Output	Low
EBI0_NANDOE	NAND Flash Output Enable	Output	Low
EBI0_NANDWE	NAND Flash Write Enable	Output	Low
SDRAM Controller			
EBI0_SDCK	SDRAM Clock	Output	
EBI0_SDCKE	SDRAM Clock Enable	Output	High
EBI0_SDCS	SDRAM Controller Chip Select Line	Output	Low
EBI0_BA0 - EBI0_BA1	Bank Select	Output	
EBI0_SDWE	SDRAM Write Enable	Output	Low
EBI0_RAS - EBI0_CAS	Row and Column Signal	Output	Low
EBI0_NWR0 - EBI0_NWR3	Write Signals	Output	Low
EBI0_NBS0 - EBI0_NBS3	Byte Mask Signals	Output	Low
EBI0_SDA10	SDRAM Address 10 Line	Output	

Table 21-2. EBI1 I/O Lines Description

Name	Function	Type	Active Level
EBI			
EBI1_D0 - EBI1_D31	Data Bus	I/O	
EBI1_A0 - EBI1_A22	Address Bus	Output	
EBI1_NWAIT	External Wait Signal	Input	Low
SMC			
EBI1_NCS0 - EBI1_NCS2	Chip Select Lines	Output	Low
EBI1_NWR0 - EBI1_NWR3	Write Signals	Output	Low
EBI1_NRD	Read Signal	Output	Low
EBI1_NWE	Write Enable	Output	Low
EBI1_NBS0 - EBI1_NBS3	Byte Mask Signals	Output	Low
EBI for NAND Flash Support			
EBI1_NANDCS	NAND Flash Chip Select Line	Output	Low
EBI1_NANDOE	NAND Flash Output Enable	Output	Low
EBI1_NANDWE	NAND Flash Write Enable	Output	Low
SDRAM Controller			
EBI1_SDCK	SDRAM Clock	Output	
EBI1_SDCKE	SDRAM Clock Enable	Output	High
EBI1_SDCS	SDRAM Controller Chip Select Line	Output	Low
EBI1_BA0 - EBI1_BA1	Bank Select	Output	
EBI1_SDWE	SDRAM Write Enable	Output	Low
EBI1_RAS - EBI1_CAS	Row and Column Signal	Output	Low
EBI1_NWR0 - EBI1_NWR3	Write Signals	Output	Low
EBI1_NBS0 - EBI1_NBS3	Byte Mask Signals	Output	Low
EBI1_SDA10	SDRAM Address 10 Line	Output	

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

Table 21-3 on page 168 details the connections between the two Memory Controllers and the EBI pins.

Table 21-3. EBIx Pins and Memory Controllers I/O Lines Connections

EBIx Pins[1]	SDRAMC I/O Lines	SMC I/O Lines
EBIx_NWR1/NBS1/CFIOR	NBS1	NWR1/NUB
EBIx_A0/NBS0	Not Supported	SMC_A0/NLB
EBIx_A1/NBS2/NWR2	Not Supported	SMC_A1
EBIx_A[11:2]	SDRAMC_A[9:0]	SMC_A[11:2]
EBIx_SDA10	SDRAMC_A10	Not Supported
EBIx_A12	Not Supported	SMC_A12
EBIx_A[14:13]	SDRAMC_A[12:11]	SMC_A[14:13]
EBIx_A[22:15]	Not Supported	SMC_A[22:15]
EBIx_A[25:23][2]	Not Supported	SMC_A[25:23]
EBIx_D[31:0]	D[31:0]	D[31:0]

Notes: 1. x indicates 0 or 1
2. Only for EBI0

21.3.1 Hardware Interface

Table 21-4 details the connections to be applied between the EBI pins and the external devices for each Memory Controller.

Table 21-4. EBI Pins and External Static Devices Connections

Signals: EBI0_, EBI1_	Pins of the Interfaced Device					
	8-bit Static Device	2 x 8-bit Static Devices	16-bit Static Device	4 x 8-bit Static Devices	2 x 16-bit Static Devices	32-bit Static Device
Controller	**SMC**					
D0 - D7	D0 - D7	D0 - D7	D0 - D7	D0 - D7	D0 - D7	D0 - D7
D8 - D15	–	D8 - D15	D8 - D15	D8 - D15	D8 - 15	D8 - 15
D16 - D23	–	–	–	D16 - D23	D16 - D23	D16 - D23
D24 - D31	–	–	–	D24 - D31	D24 - D31	D24 - D31
A0/NBS0	A0	–	NLB	–	NLB(3)	BE0(6)
A1/NWR2/NBS2	A1	A0	A0	WE(2)	NLB(4)	BE2(6)
A2 - A22	A[2:22]	A[1:21]	A[1:21]	A[0:20]	A[0:20]	A[0:20]
A23 - A25(5)	A[23:25]	A[22:24]	A[22:24]	A[21:23]	A[21:23]	A[21:23]
NCS0	CS	CS	CS	CS	CS	CS
NCS1/SDCS	CS	CS	CS	CS	CS	CS
NCS2(5)	CS	CS	CS	CS	CS	CS
NCS2/NANDCS(7)	CS	CS	CS	CS	CS	CS
NCS3/NANDCS(5)	CS	CS	CS	CS	CS	CS
NCS4/CFCS0(5)	CS	CS	CS	CS	CS	CS
NCS5/CFCS1(5)	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(6)
NWR3/NBS3	–	–	–	WE(2)	NUB(4)	BE3(6)

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. EBI0 signals only.
6. BEx: Byte x Enable (x = 0,1,2 or 3).
7. EBI1 signals only.

Table 21-5. EBI Pins and External Devices Connections

Signals: EBI0_, EBI1_	Pins of the Interfaced Device			
	SDRAM	CompactFlash (EBI0 only)	CompactFlash True IDE Mode (EBI0 only)	NAND Flash
Controller	**SDRAMC**	**SMC**		
D0 - D7	D0 - D7	D0 - D7	D0 - D7	I/O0-I/O7
D8 - D15	D8 - D15	D8 - 15	D8 - 15	I/O8-I/O15[6]
D16 - D31	D16 - D31	–	–	–
A0/NBS0	DQM0	A0	A0	–
A1/NWR2/NBS2	DQM2	A1	A1	–
A2 - A10	A[0:8]	A[2:10]	A[2:10]	–
A11	A9	–	–	–
SDA10	A10	–	–	–
A12	–	–	–	–
A13 - A14	A[11:12]	–	–	–
A15	–	–	–	–
A16/BA0	BA0	–	–	–
A17/BA1	BA1	–	–	–
A18 - A20	–	–	–	–
A21/NANDALE	–	–	–	ALE
A22/NANDCLE	–	REG	REG	CLE
A23 - A24[3]	–	–	–	–
A25[3]	–	CFRNW[1]	CFRNW[1]	–
NCS0	–	–	–	–
NCS1/SDCS	CS	–	–	–
NCS2[3]	–	–	–	–
NCS2/NANDCS[4]	–	–	–	–
NCS3/NANDCS[3]	–	–	–	CE[5]
NCS4/CFCS0[3]	–	CFCS0[1]	CFCS0[1]	–
NCS5/CFCS1[3]	–	CFCS1[1]	CFCS1[1]	–
NANDOE	–	–	–	OE
NANDWE	–	–	–	WE
NRD/CFOE	–	OE	–	–
NWR0/NWE/CFWE	–	WE	WE	–
NWR1/NBS1/CFIOR	DQM1	IOR	IOR	–
NWR3/NBS3/CFIOW	DQM3	IOW	IOW	–
CFCE1[3]	–	CE1	CS0	–
CFCE2[3]	–	CE2	CS1	–

Table 21-5. EBI Pins and External Devices Connections (Continued)

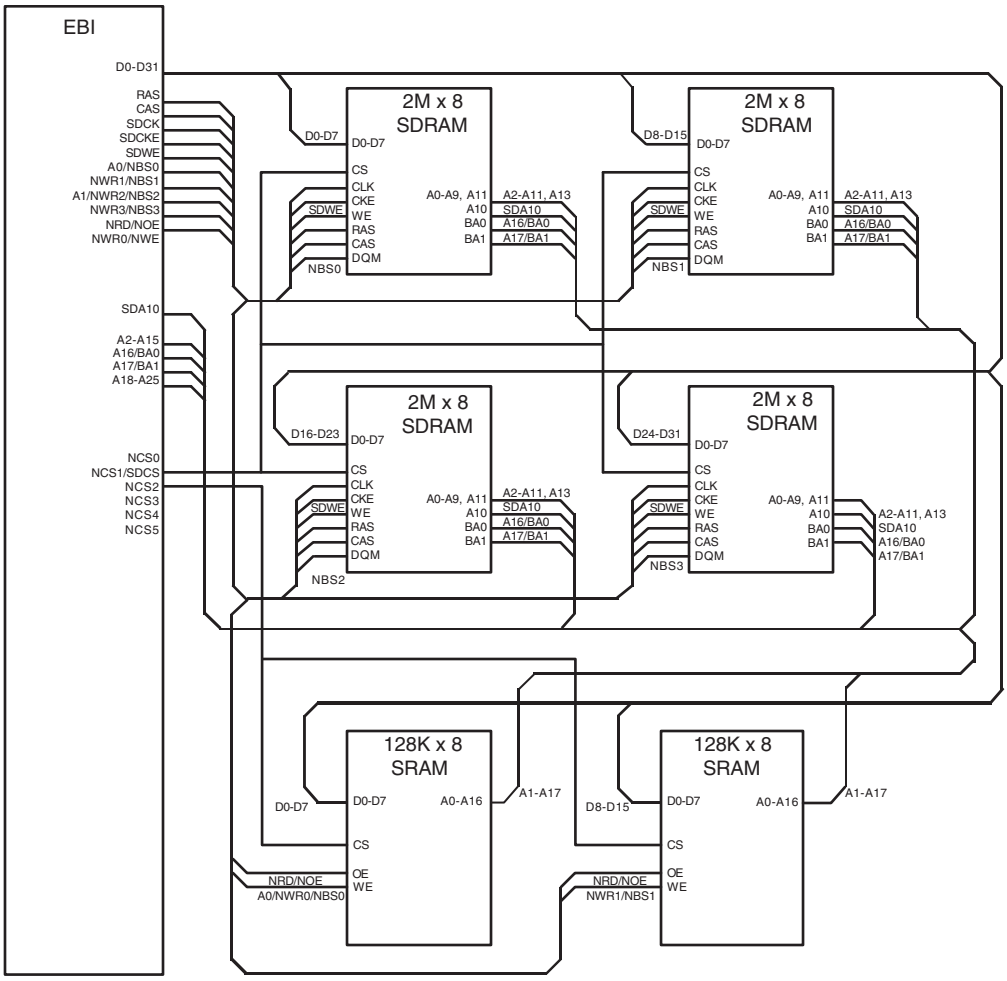
Signals: EBI0_, EBI1_	Pins of the Interfaced Device			
	SDRAM	CompactFlash (EBI0 only)	CompactFlash True IDE Mode (EBI0 only)	NAND Flash
Controller	**SDRAMC**	**SMC**		
SDCK	CLK	–	–	–
SDCKE	CKE	–	–	–
RAS	RAS	–	–	–
CAS	CAS	–	–	–
SDWE	WE	–	–	–
NWAIT[7]	–	WAIT	WAIT	–
Pxx[2]	–	CD1 or CD2	CD1 or CD2	–
Pxx[2]	–	–	–	CE[5]
Pxx[2]	–	–	–	RDY

Notes:
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. EBI0 signals only
4. EBI1 signals only
5. CE connection depends on the NAND Flash.
   For standard NAND Flash devices, it must be connected to any free PIO line.
   For "CE don't care" NAND Flash devices, it can be either connected to NCS3/NANDCS or to any free PIO line.
6. I/O8 - !/O15 pins used only for 16-bit NAND Flash device.
7. EBI0_NWAIT signal is multiplexed with PD5. EBI1_NWAIT signal is multiplexed with PE20.

21.3.2 Connection Examples

Figure 21-3 shows an example of connections between the EBI and external devices.

Figure 21-3. EBI Connections to Memory Devices



21.4 Product Dependencies

21.4.1 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.

21.5 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 SDRAM Controller (SDRAMC)
- the ECC Controller (ECC)
- a chip select assignment feature that assigns an AHB address space to the external devices
- a multiplex controller circuit that shares the pins between the different Memory Controllers
- programmable CompactFlash support logic (EBI0 only)
- programmable NAND Flash support logic

21.5.1 Bus Multiplexing

The EBI0 and EBI1 offers a complete set of control signals that share the 32-bit data lines, the address lines of up to 26 bits and the control signals through a multiplex logic operating in function of the memory area requests.

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

21.5.2 Pull-up Control

The EBI0_CSA and EBI1_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 EBIx_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.

21.5.3 Static Memory Controller

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

21.5.4 SDRAM Controller

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

21.5.5 ECC Controller

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

21.5.6 CompactFlash Support (EBI0 only)

The External Bus Interface 0 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 EBI0_CS4A and/or EBI0_CS5A bit of the EBI0_CSA Register in the Chip Configuration User Interface to the appropriate value enables this logic. For details on this register, refer to the in the Bus Matrix Section. Access to an external CompactFlash device is then made by accessing the address space reserved to NCS4 and/or NCS5 (i.e., between 0x5000 0000 and 0x5FFF FFFF for NCS4 and between 0x6000 0000 and 0x6FFF FFFF for NCS5).

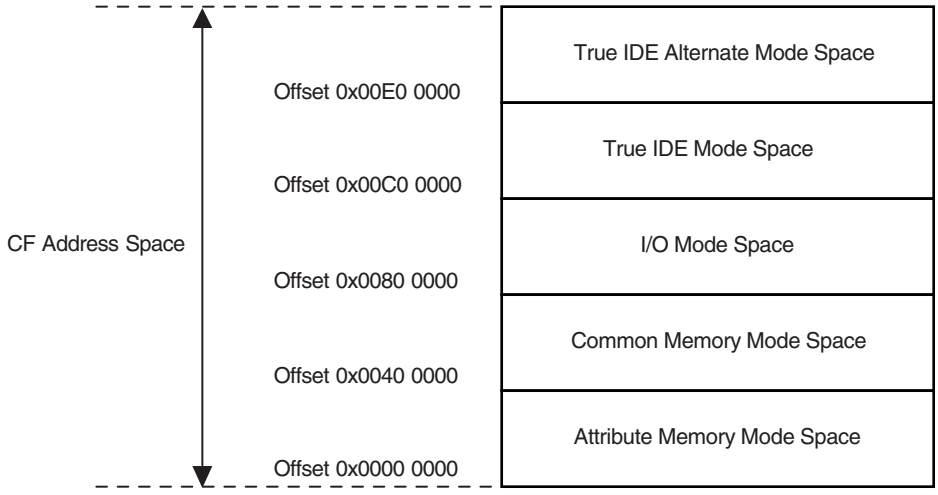
All CompactFlash modes (Attribute Memory, Common Memory, I/O and True IDE) are supported but the signals _IOIS16 (I/O and True IDE modes) and _ATA SEL (True IDE mode) are not handled.

21.5.6.1 I/O Mode, Common Memory Mode, Attribute Memory Mode and True IDE Mode

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

The different modes are accessed through a specific memory mapping as illustrated on Figure 21-4. A[23:21] bits of the transfer address are used to select the desired mode as described in Table 21-6 on page 175.

Figure 21-4. CompactFlash Memory Mapping



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

Table 21-6. 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

21.5.6.2 CFCE1 and CFCE2 Signals

To cover all types of access, the SMC must be alternatively set to drive 8-bit data bus or 16-bit data bus. The odd byte access on the D[7:0] bus is only possible when the SMC is configured to drive 8-bit memory devices on the corresponding NCS pin (NCS4 or NCS5). The Chip Select Register (DBW field in the corresponding Chip Select Register) of the NCS4 and/or NCS5 address space must be set as shown in Table 21-7 to enable the required access type.

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

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

Table 21-7. CFCE1 and CFCE2 Truth Table

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

21.5.6.3 Read/Write Signals

In I/O mode and True IDE mode, the CompactFlash logic drives the read and write command signals of the SMC on CFIOR and 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 21-5 on page 176 demonstrates a schematic representation of this logic.

Attribute memory mode, common memory mode and I/O mode are supported by setting the address setup and hold time on the NCS4 (and/or NCS5) chip select to the appropriate values. For details on these signal waveforms, please refer to the section: Setup and Hold Cycles of the Static Memory Controller section.

Figure 21-5. CompactFlash Read/Write Control Signals

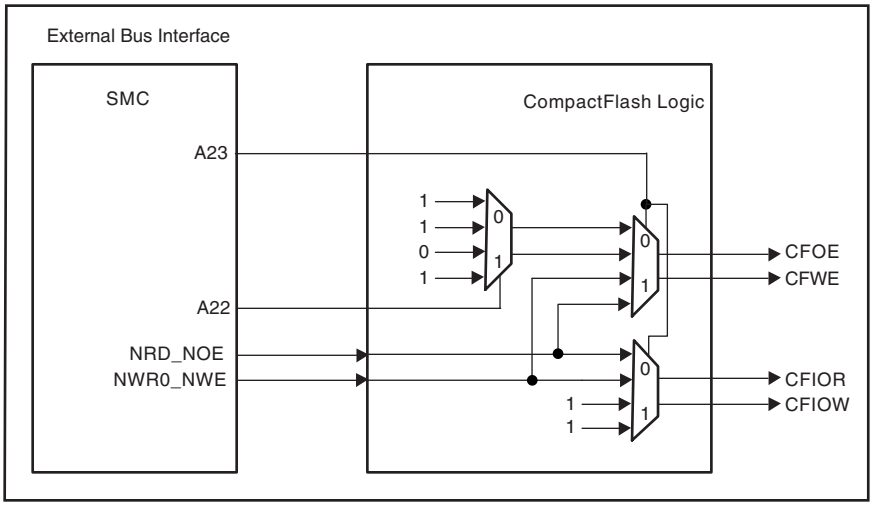


Table 21-8. CompactFlash Mode Selection

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

21.5.6.4 *Multiplexing of CompactFlash Signals on EBI Pins*

Table 21-9 on page 177 and Table 21-10 on page 177 illustrate the multiplexing of the CompactFlash logic signals with other EBI signals on the EBI pins. The EBI pins in Table 21-9 are strictly dedicated to the CompactFlash interface as soon as the EBI0_CS4A and/or EBI0_CS5A field of the EBI0_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 21-10 on page 177 remain shared between all memory areas when the corresponding CompactFlash interface is enabled (EBI0_CS4A = 1 and/or EBI0_CS5A = 1).

Table 21-9. Dedicated CompactFlash Interface Multiplexing

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

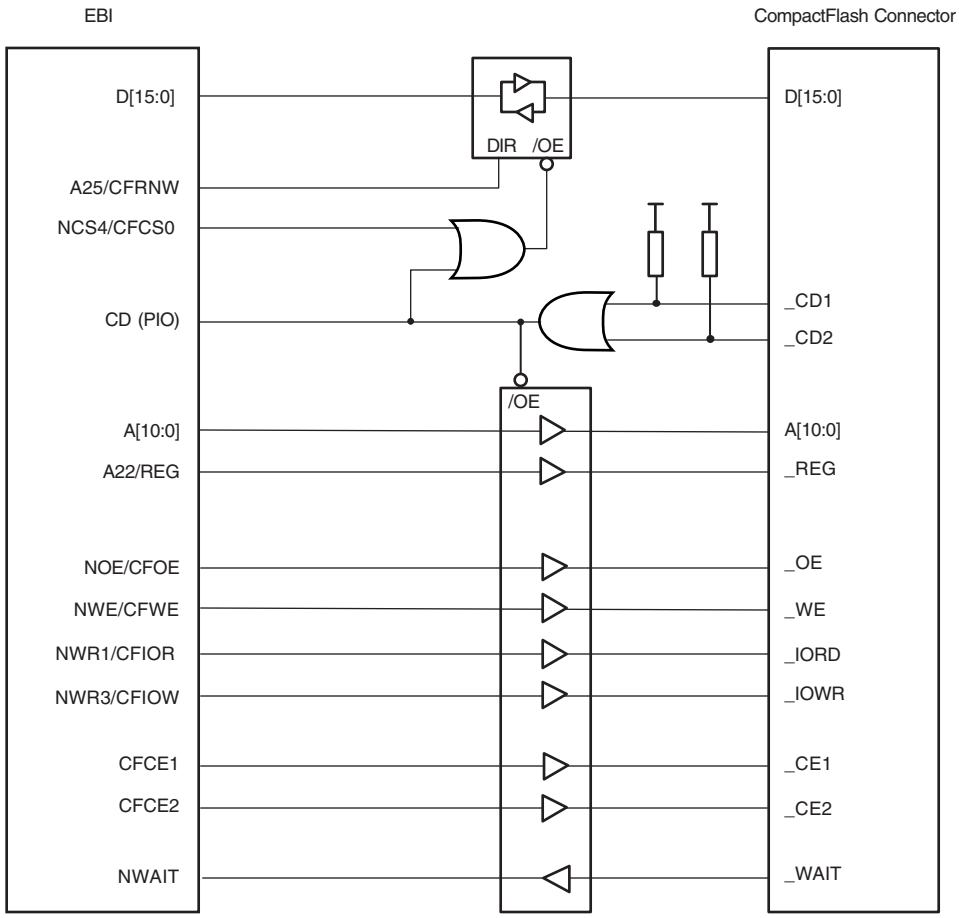
Table 21-10. Shared CompactFlash Interface Multiplexing

	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

21.5.6.5 Application Example

Figure 21-6 on page 178 illustrates an example of a CompactFlash application. CFCS0 and CFRNW signals are not directly connected to the CompactFlash slot 0, but do control the direction and the output enable of the buffers between the EBI and the CompactFlash Device. The timing of the CFCS0 signal is identical to the NCS4 signal. Moreover, the CFRNW signal remains valid throughout the transfer, as does the address bus. The CompactFlash _WAIT signal is connected to the NWAIT input of the Static Memory Controller. For details on these waveforms and timings, refer to the Static Memory Controller Section.

Figure 21-6. CompactFlash Application Example



21.5.7 NAND Flash Support

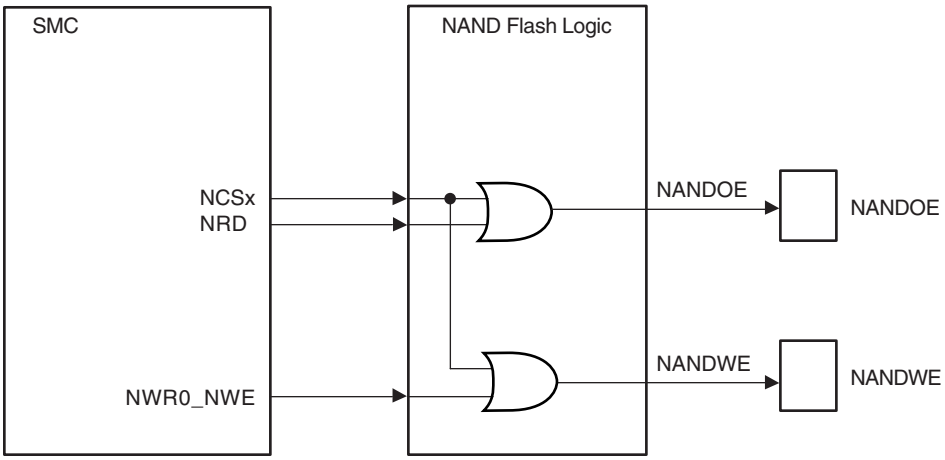
External Bus Interfaces 0 and 1 integrate circuitry that interfaces to NAND Flash devices.

21.5.7.1 External Bus Interface 0

The NAND Flash logic is driven by the Static Memory Controller on the NCS3 address space. Programming the EBI0_CS3A field in the EBI0_CSA Register in the Chip Configuration User Interface to the appropriate value enables the NAND Flash logic. For details on this register, refer to the Bus Matrix Section. Access to an external NAND Flash device is then made by accessing the address space reserved to NCS3 (i.e., between 0x4000 0000 and 0x4FFF FFFF).

The NAND Flash Logic drives the read and write command signals of the SMC on the NANDOE and NANDWE signals when the NCS3 signal is active. NANDOE and NANDWE are invalidated as soon as the transfer address fails to lie in the NCS3 address space. See Figure "NAND Flash Signal Multiplexing on EBI Pins" on page 179 for more information. For details on these waveforms, refer to the Static Memory Controller section.

Figure 21-7. NAND Flash Signal Multiplexing on EBI Pins



21.5.7.2 External Bus Interface 1

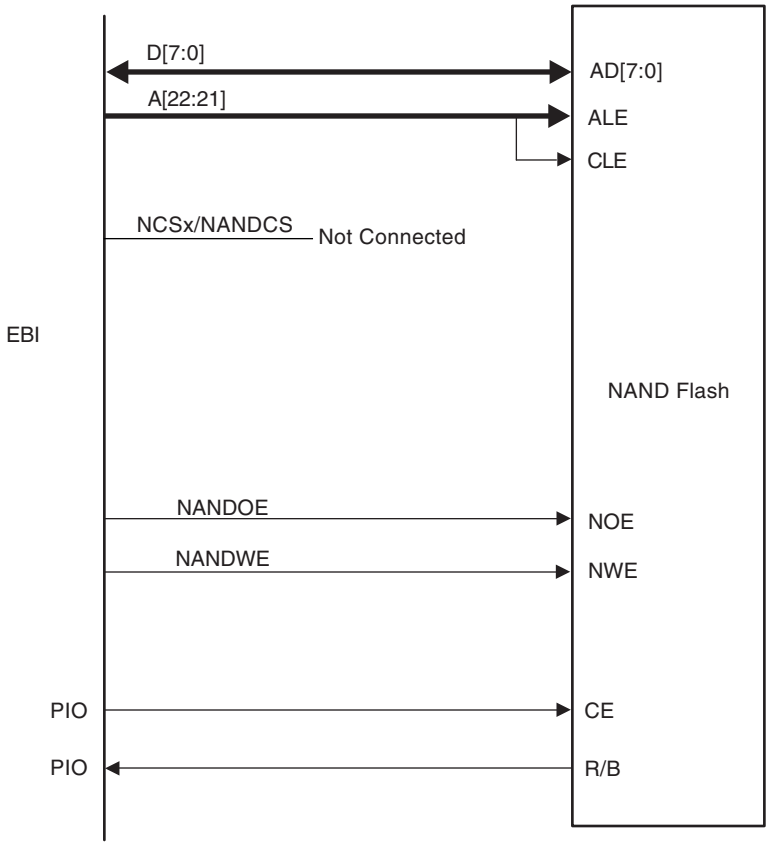
The NAND Flash logic is driven by the Static Memory Controller on the NCS2 address space. Programming the EBI1_CS2A field in the EBI1_CSA Register in the Chip Configuration User Interface to the appropriate value enables the NAND Flash logic. For details on this register, refer to the Bus Matrix Section. Access to an external NAND Flash device is then made by accessing the address space reserved to NCS2 (i.e., between 0x9000 0000 and 0x9FFF FFFF).

The NAND Flash Logic drives the read and write command signals of the SMC on the NANDOE and NANDWE signals when the NCS2 signal is active. NANDOE and NANDWE are invalidated as soon as the transfer address fails to lie in the NCS2 address space. See Figure 21-7 on page 179 for more information. For details on these waveforms, refer to the Static Memory Controller section.

21.5.7.3 NAND Flash Signals

The address latch enable and command latch enable signals on the NAND Flash device are driven by address bits A22 and A21 of the EBI address bus. The command, address or data words on the data bus of the NAND Flash device are distinguished by using their address within the NCSx address space. The chip enable (CE) signal of the device and the ready/busy (R/B) signals are connected to PIO lines. The CE signal then remains asserted even when NCSx is not selected, preventing the device from returning to standby mode.

Figure 21-8. NAND Flash Application Example



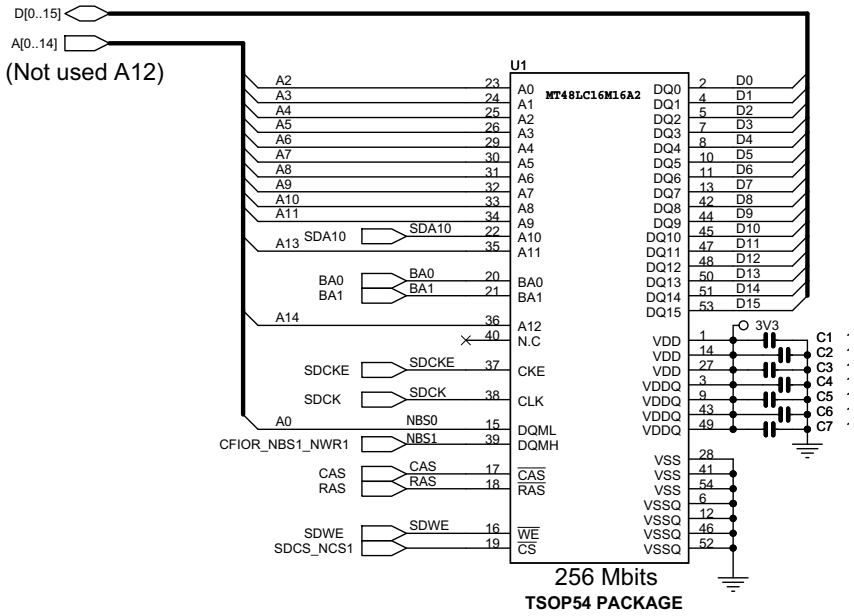
Note: The External Bus Interfaces 0 and 1 are also able to support 16-bit devices.

21.6 Implementation Examples

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

21.6.1 16-bit SDRAM

Figure 21-9. Hardware Configuration



21.6.1.1 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.

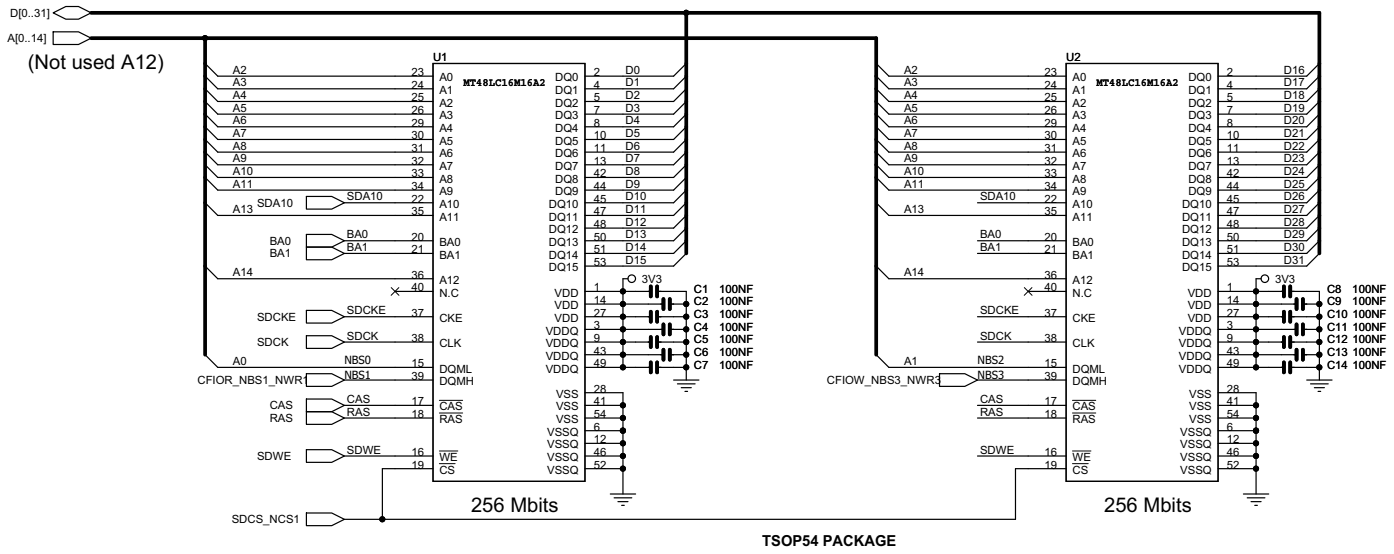
The Data Bus Width is to be programmed to 16 bits.

EBI1 SDCS, SDWE, SDCKE, SDA10, RAS and CAS signals are multiplexed with PIO lines and thus the dedicated PIOs must be programmed in peripheral mode in the PIO controller.

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

21.6.2 32-bit SDRAM

21.6.2.1 Hardware Configuration



21.6.2.2 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.

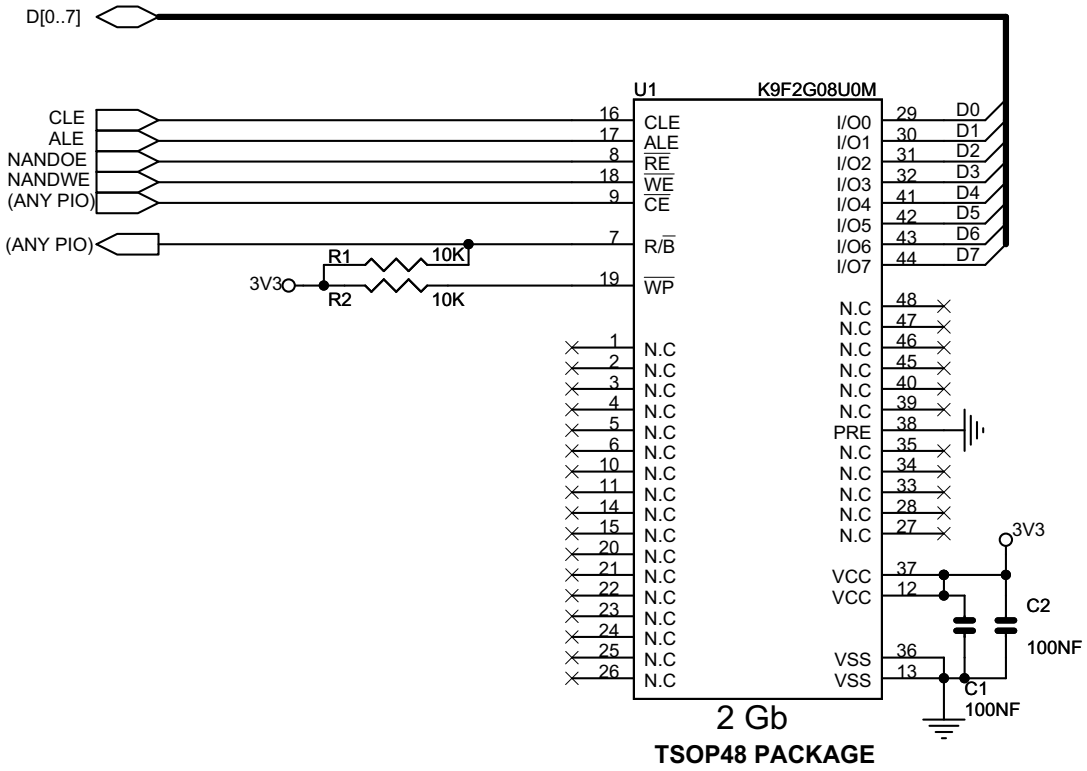
The Data Bus Width is to be programmed to 32 bits. The data lines D[16..31] are multiplexed with PIO lines and thus the dedicated PIOs must be programmed in peripheral mode in the PIO controller.

EBI1 SDCS, SDWE, SDCKE, SDA10, RAS and CAS signals are multiplexed with PIO lines and thus the dedicated PIOs must be programmed in peripheral mode in the PIO controller.

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

21.6.3 8-bit NAND Flash

21.6.3.1 Hardware Configuration



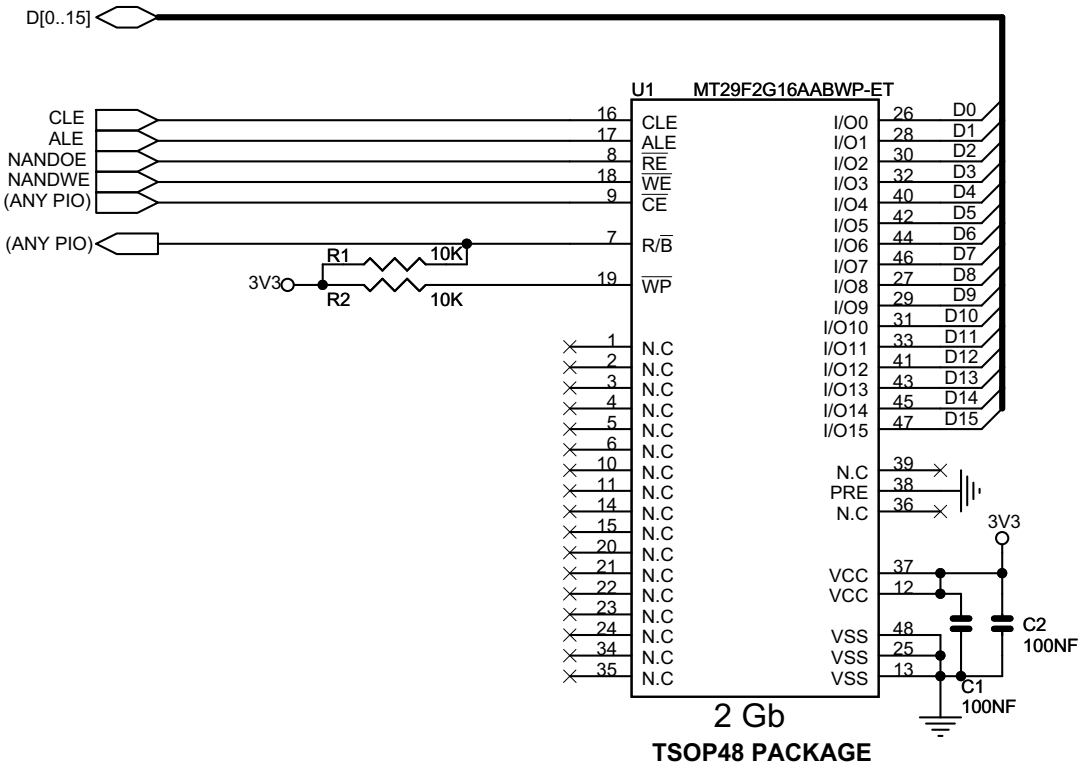
21.6.3.2 Software Configuration

The following configuration has to be performed:

- Assign the EBI CS3 to the NAND Flash by setting the bit EBI_CS3A in the EBI Chip Select Assignment Register located in the bus matrix memory space
- Reserve A21 / A22 for ALE / CLE functions. Address and Command Latches are controlled respectively by setting to 1 the address bit A21 and A22 during accesses.
- EBI1 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.

21.6.4 16-bit NAND Flash

21.6.4.1 Hardware Configuration

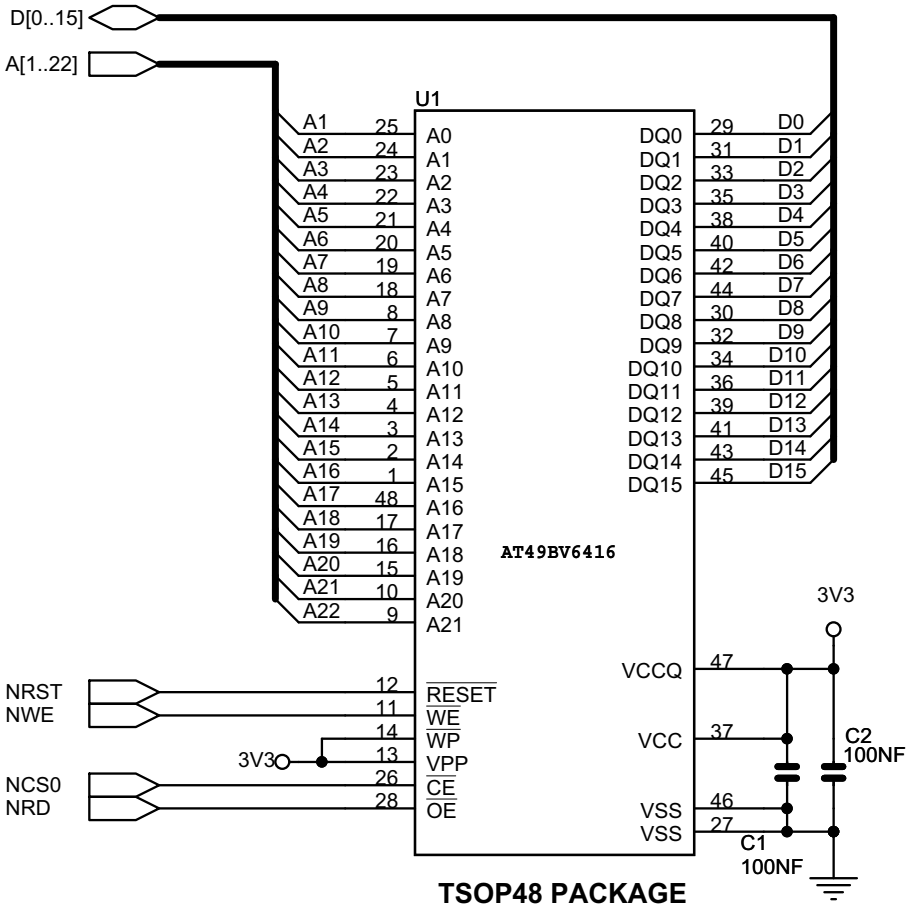


21.6.4.2 Software Configuration

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

21.6.5 NOR Flash on NCS0

21.6.5.1 Hardware Configuration



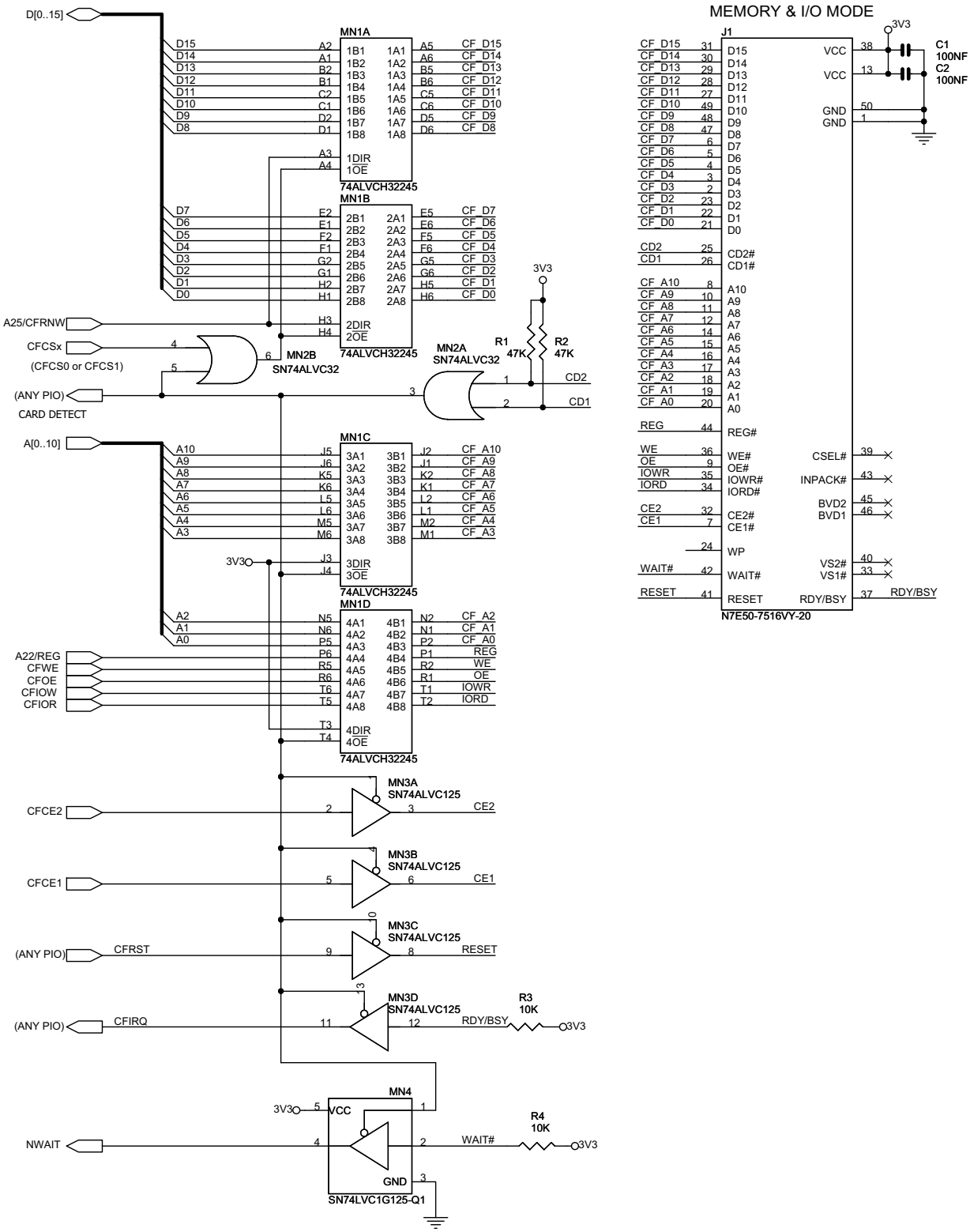
21.6.5.2 Software Configuration

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

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

21.6.6 Compact Flash

21.6.6.1 Hardware Configuration



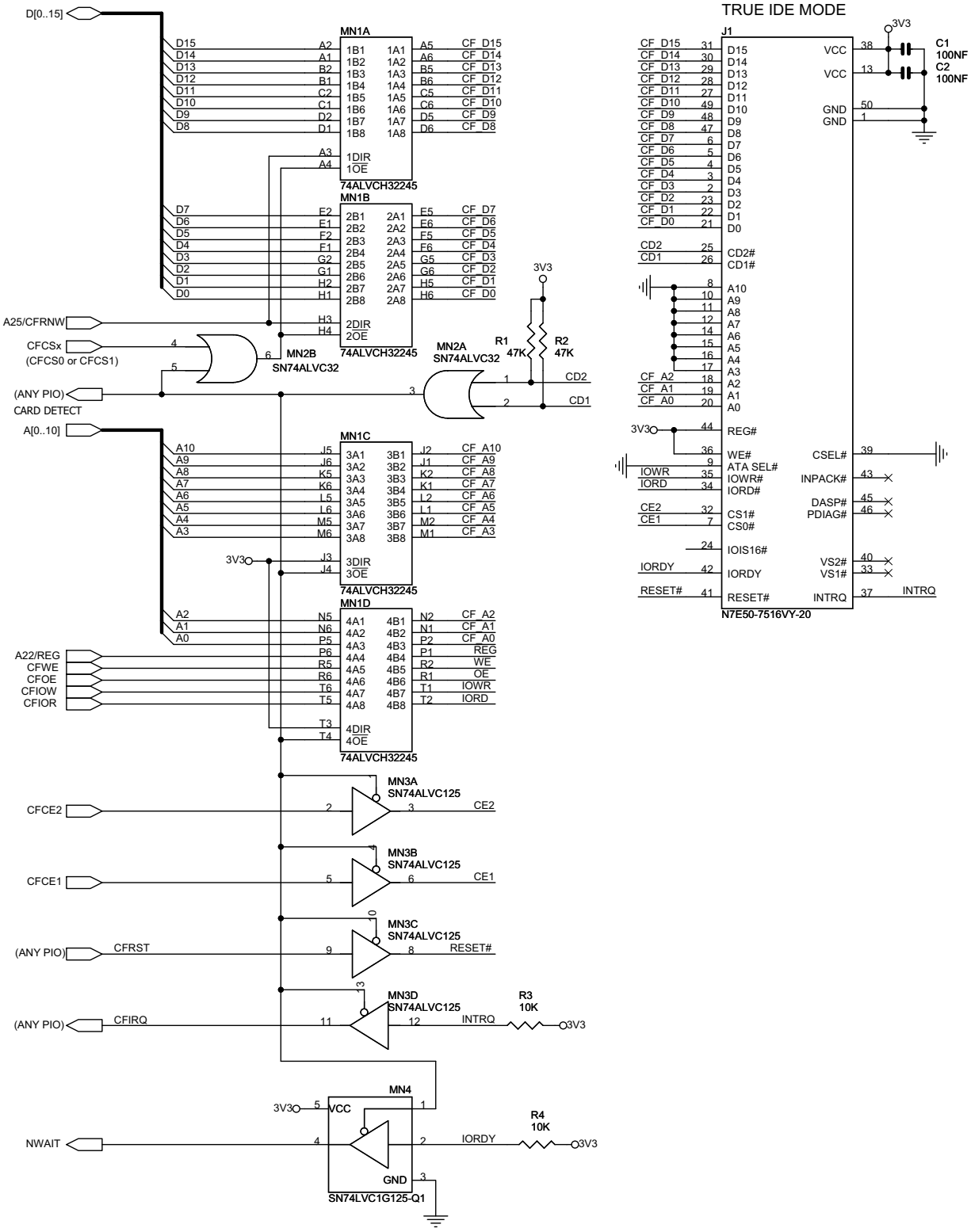
21.6.6.2 Software Configuration

The following configuration has to be performed:

- Assign the EBI CS4 and/or EBI_CS5 to the CompactFlash Slot 0 or/and Slot 1 by setting the bit EBI_CS4A or/and EBI_CS5A in the EBI Chip Select Assignment Register located in the bus matrix memory space.
- The address line A23 is to select I/O (A23=1) or Memory mode (A23=0) and the address line A22 for REG function.
- A23, CFRNW, CFS0, CFCS1, CFCE1 and CFCE2 signals are multiplexed with PIO lines and thus the dedicated PIOs must be programmed in peripheral mode in the PIO controller.
- Configure a PIO line as an output for CFRST and two others as an input for CFIRQ and CARD DETECT functions respectively.
- Configure SMC CS4 and/or SMC_CS5 (for Slot 0 or 1) Setup, Pulse, Cycle and Mode accordingly to Compact Flash timings and system bus frequency.

21.6.7 Compact Flash True IDE

21.6.7.1 Hardware Configuration



21.6.7.2 Software Configuration

The following configuration has to be performed:

- Assign the EBI CS4 and/or EBI_CS5 to the CompactFlash Slot 0 or/and Slot 1 by setting the bit EBI_CS4A or/and EBI_CS5A in the EBI Chip Select Assignment Register located in the bus matrix memory space.
- The address line A21 is to select Alternate True IDE (A21=1) or True IDE (A21=0) modes.
- CFRNW, CFS0, CFCS1, CFCE1 and CFCE2 signals are multiplexed with PIO lines and thus the dedicated PIOs must be programmed in peripheral mode in the PIO controller.
- Configure a PIO line as an output for CFRST and two others as an input for CFIRQ and CARD DETECT functions respectively.
- Configure SMC CS4 and/or SMC_CS5 (for Slot 0 or 1) Setup, Pulse, Cycle and Mode accordingly to Compact Flash timings and system bus frequency.

22. Static Memory Controller (SMC)

22.1 Overview

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

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.

22.2 I/O Lines Description

Table 22-1. 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

22.3 Multiplexed Signals

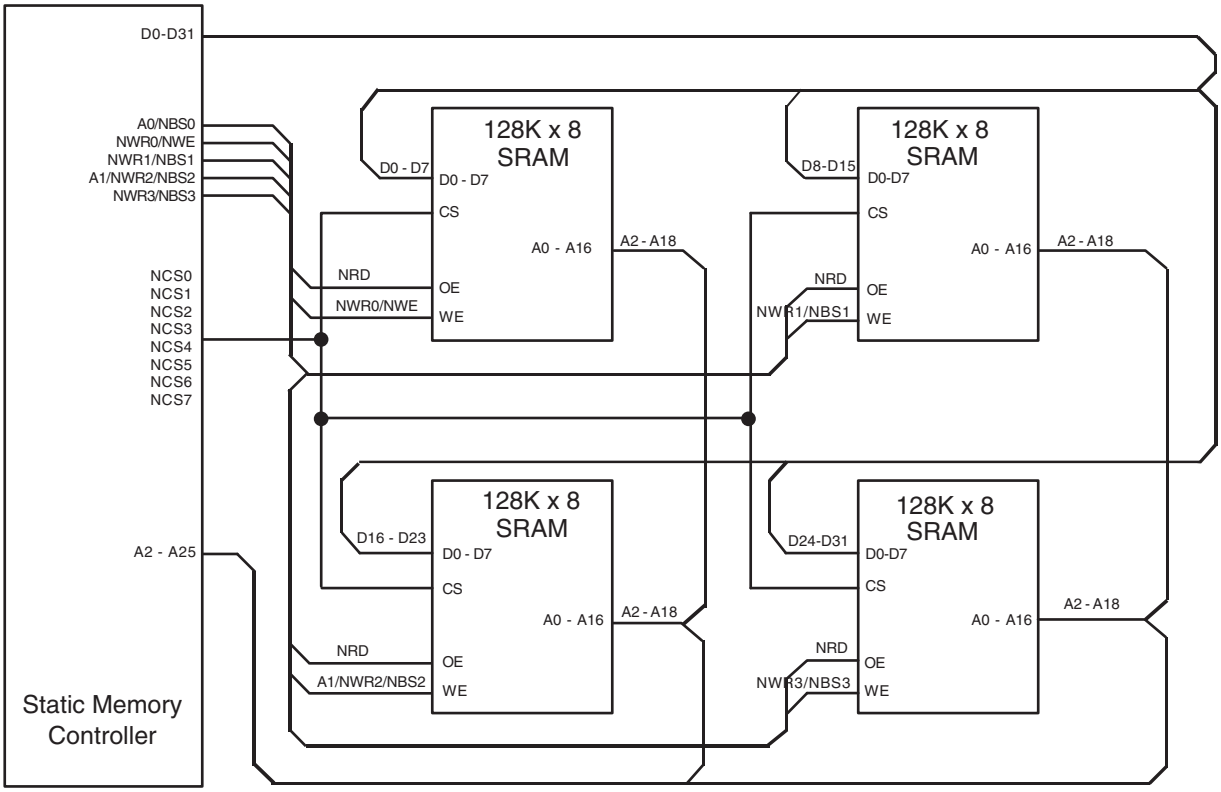
Table 22-2. Static Memory Controller (SMC) Multiplexed Signals

Multiplexed Signals			Related Function
NWR0	NWE		Byte-write or byte-select access, see “Byte Write or Byte Select Access” on page 194
A0	NBS0		8-bit or 16-/32-bit data bus, see “Data Bus Width” on page 194
NWR1	NBS1		Byte-write or byte-select access see “Byte Write or Byte Select Access” on page 194
A1	NWR2	NBS2	8-/16-bit or 32-bit data bus, see “Data Bus Width” on page 194. Byte-write or byte-select access, see “Byte Write or Byte Select Access” on page 194
NWR3	NBS3		Byte-write or byte-select access see “Byte Write or Byte Select Access” on page 194

22.4 Application Example

22.4.1 Hardware Interface

Figure 22-1. SMC Connections to Static Memory Devices



22.5 Product Dependencies

22.5.1 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.

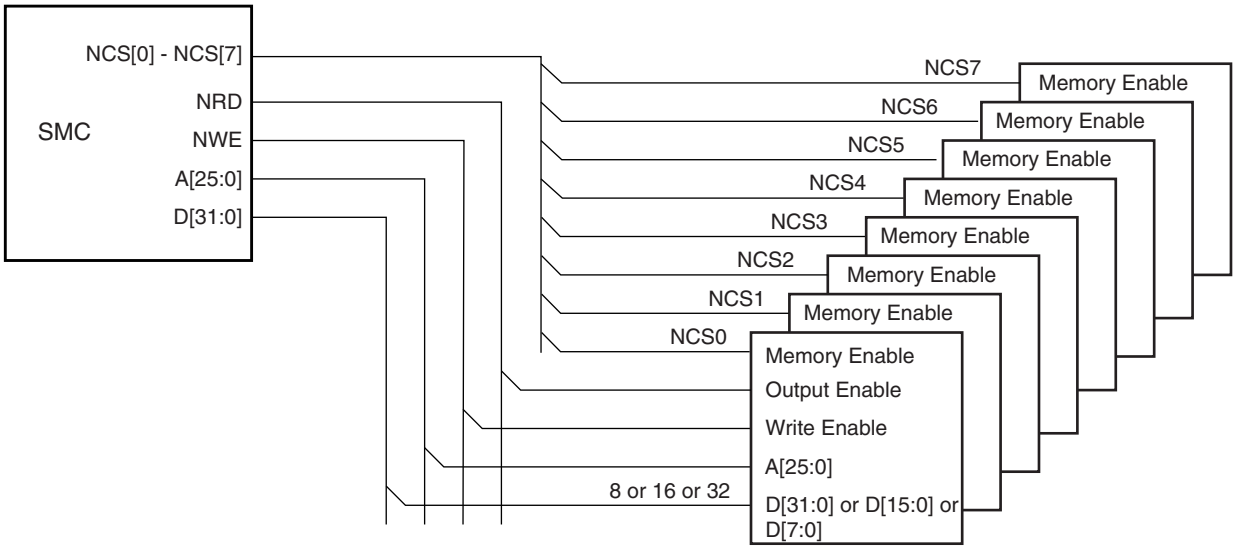
22.6 External Memory Mapping

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 22-2).

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 22-2. Memory Connections for Eight External Devices



22.7 Connection to External Devices

22.7.1 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 22-3 shows how to connect a 512K x 8-bit memory on NCS2. Figure 22-4 shows how to connect a 512K x 16-bit memory on NCS2. Figure 22-5 shows two 16-bit memories connected as a single 32-bit memory

22.7.2 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 22-3. Memory Connection for an 8-bit Data Bus

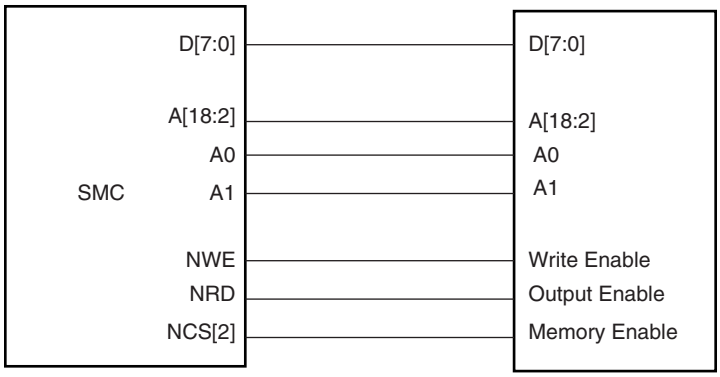


Figure 22-4. Memory Connection for a 16-bit Data Bus

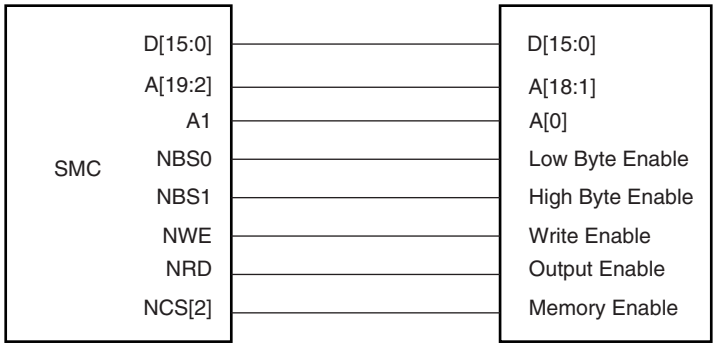
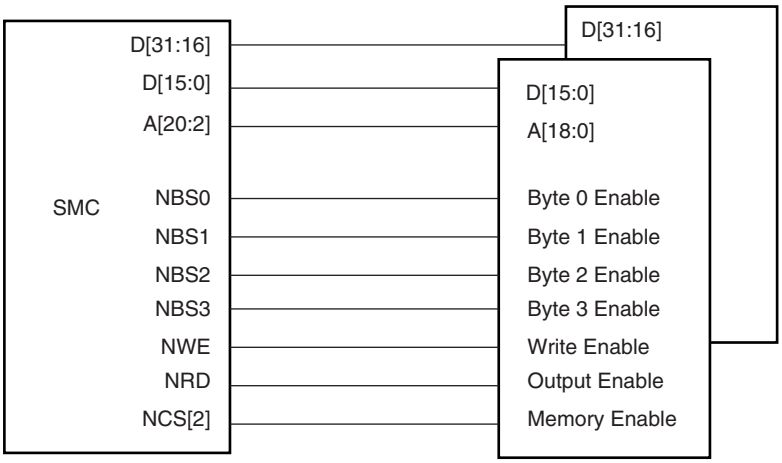


Figure 22-5. Memory Connection for a 32-bit Data Bus



22.7.2.1 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 22-6.

22.7.2.2 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.

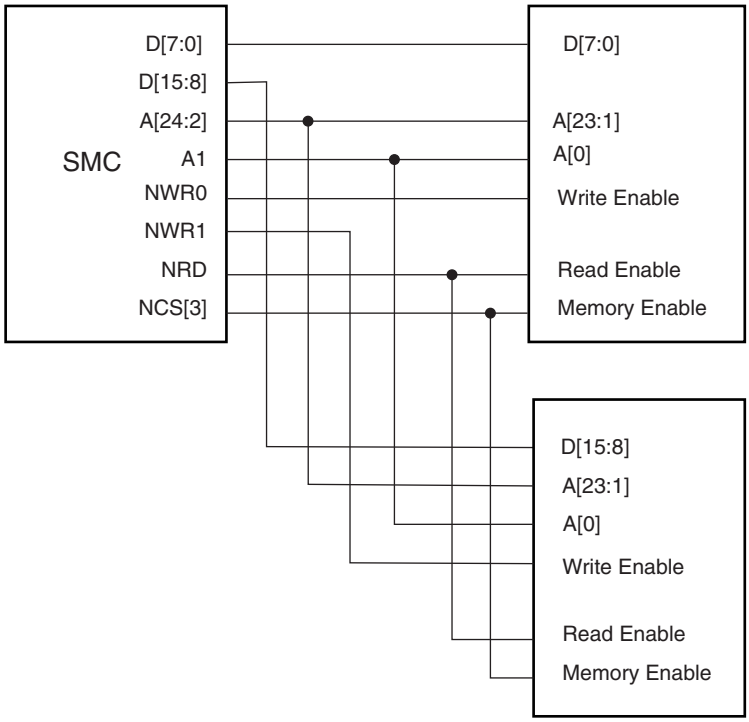
- For 16-bit devices: the SMC provides NBS0 and NBS1 selection signals for respectively byte0 (lower byte) and byte1 (upper byte) of a 16-bit bus.

  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 22-7 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 22-6. Connection of 2 x 8-bit Devices on a 16-bit Bus: Byte Write Option



22.7.2.3 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 22-3 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 22-7. Connection of 2x16-bit Data Bus on a 32-bit Data Bus (Byte Select Option)

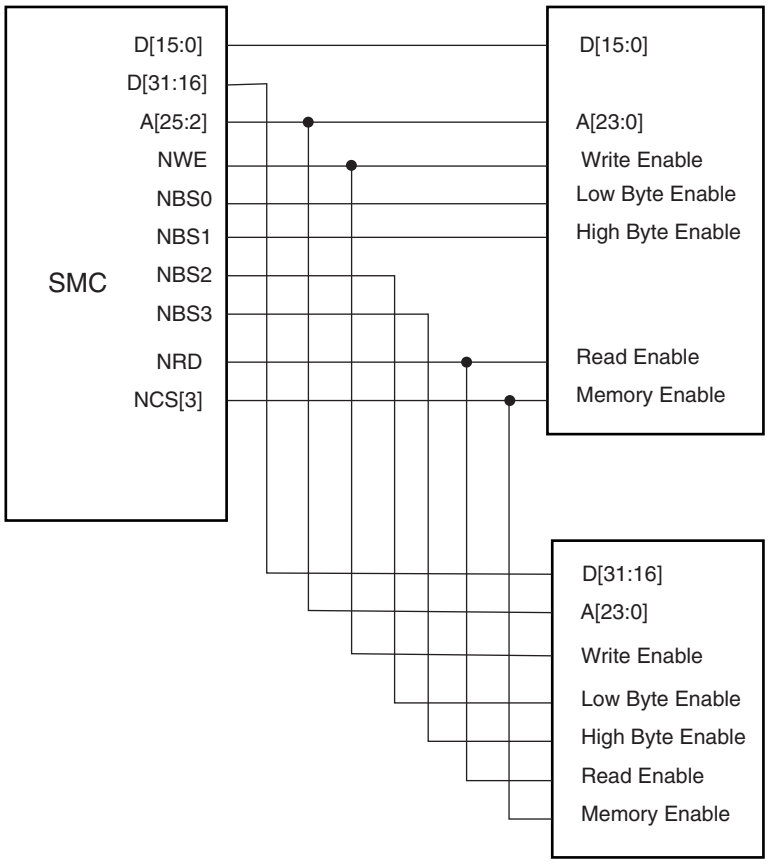


Table 22-3. SMC Multiplexed Signal Translation

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			

22.8 Standard Read and Write Protocols

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

22.8.1 Read Waveforms

The read cycle is shown on Figure 22-8.

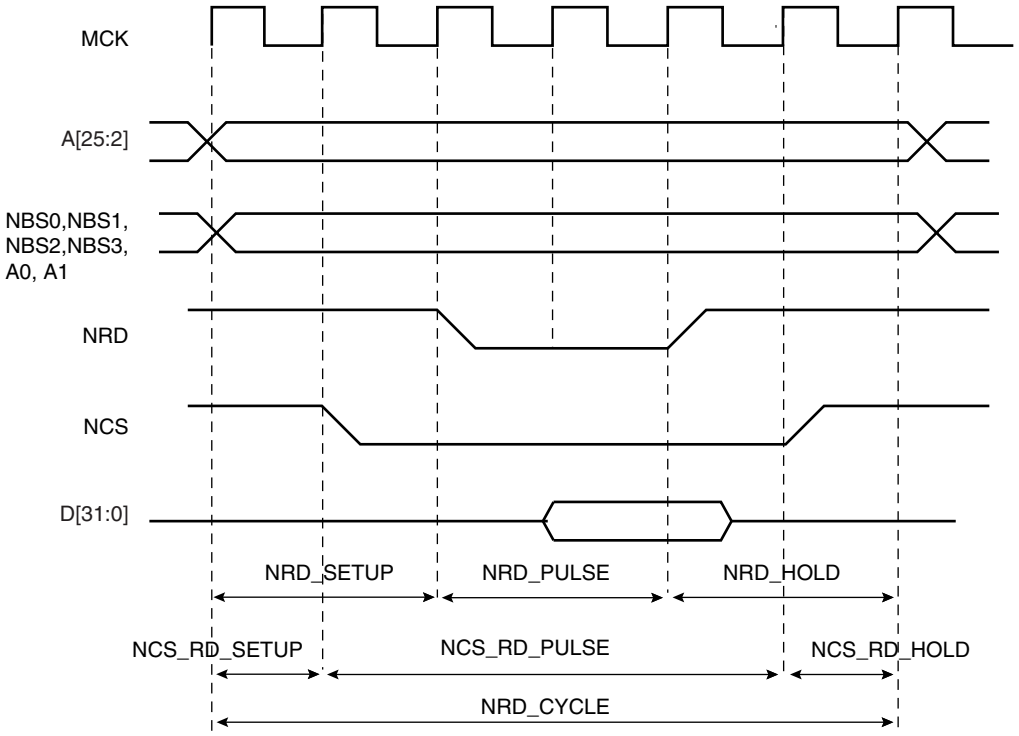
The read cycle starts with the address setting on the memory address bus, i.e.:

{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 22-8. Standard Read Cycle



22.8.1.1 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.

22.8.1.2 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.

22.8.1.3 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:

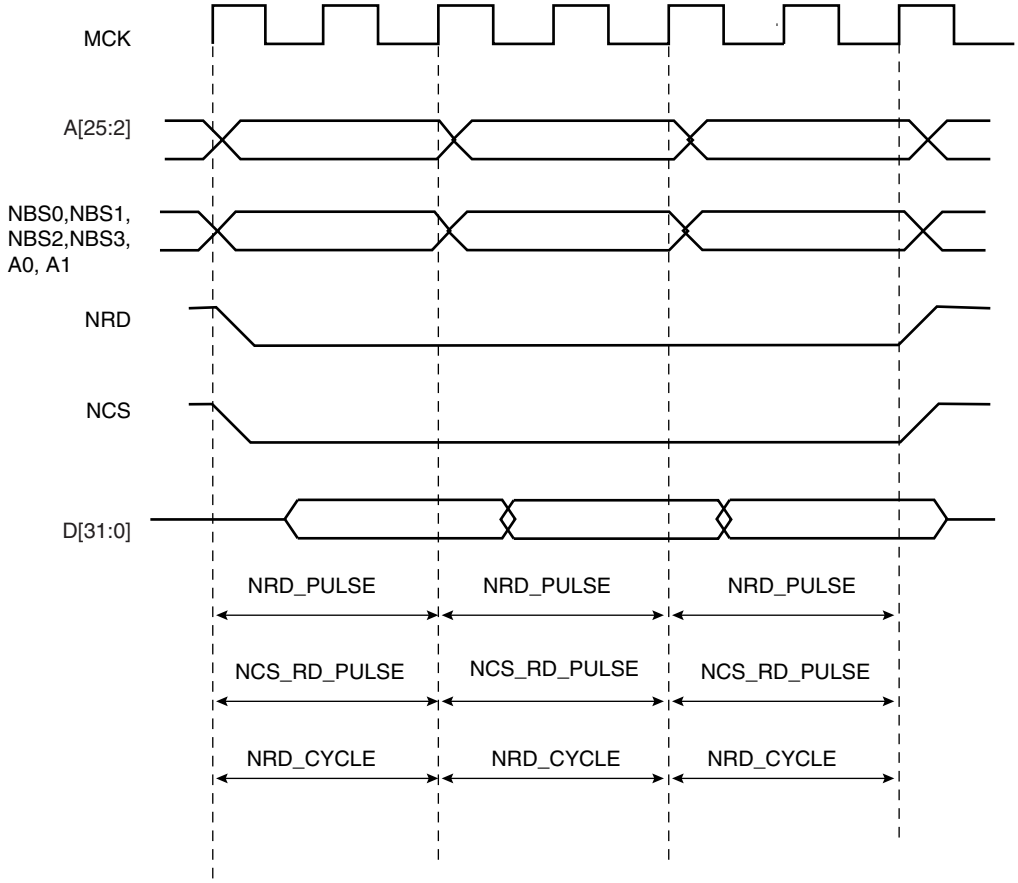
NRD_HOLD = NRD_CYCLE - NRD SETUP - NRD PULSE

NCS_RD_HOLD = NRD_CYCLE - NCS_RD_SETUP - NCS_RD_PULSE

22.8.1.4 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 22-9).

Figure 22-9. No Setup, No Hold On NRD and NCS Read Signals



22.8.1.5 Null Pulse

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

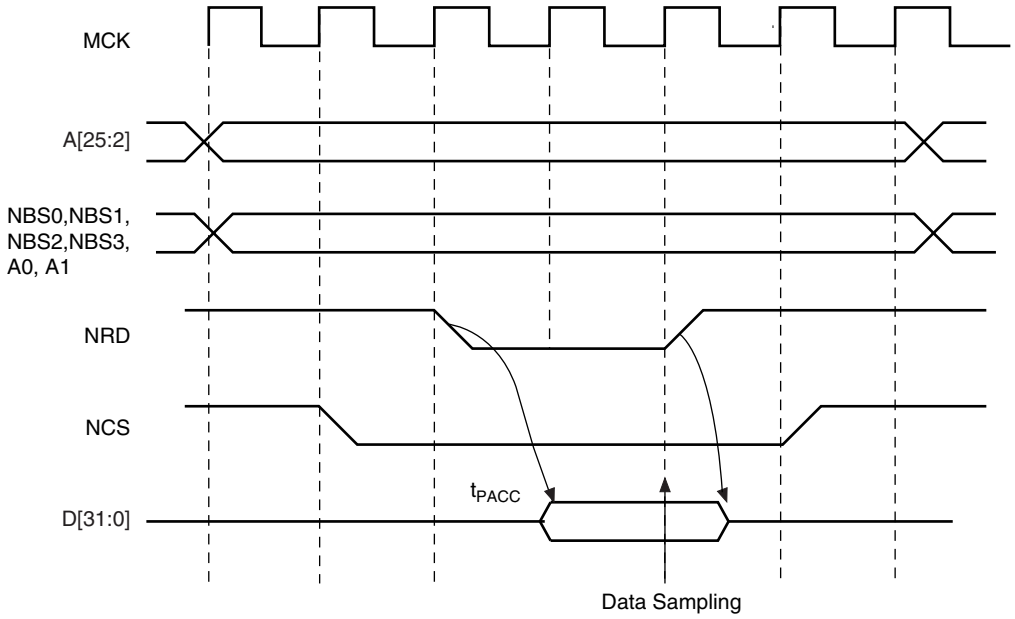
22.8.2 Read Mode

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.

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

Figure 22-10 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.

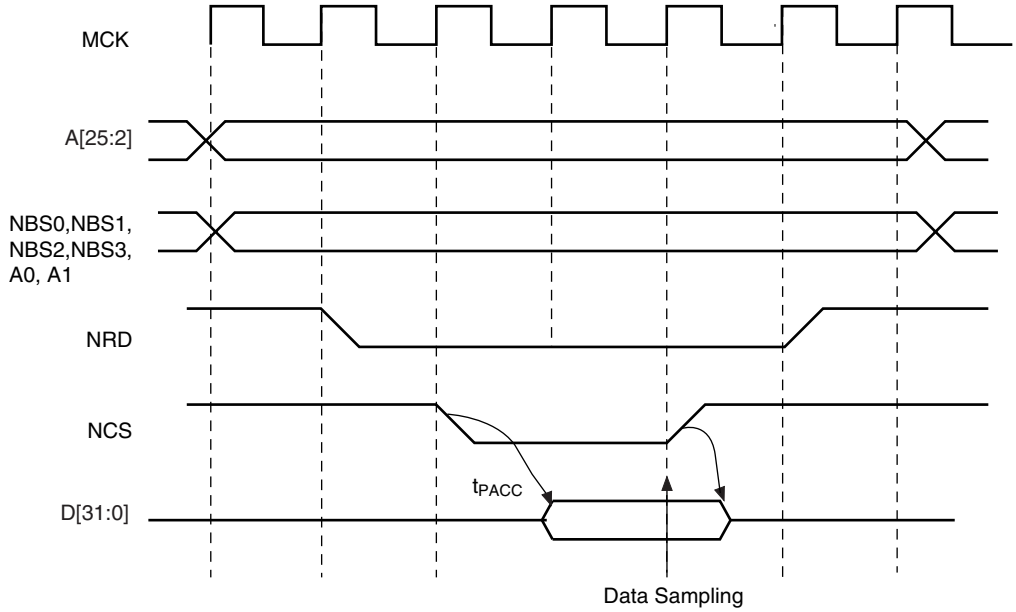
Figure 22-10. READ_MODE = 1: Data is sampled by SMC before the rising edge of NRD



22.8.2.2 Read is Controlled by NCS (READ_MODE = 0)

Figure 22-11 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 22-11. READ_MODE = 0: Data is sampled by SMC before the rising edge of NCS



22.8.3 Write Waveforms

The write protocol is similar to the read protocol. It is depicted in Figure 22-12. The write cycle starts with the address setting on the memory address bus.

22.8.3.1 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.

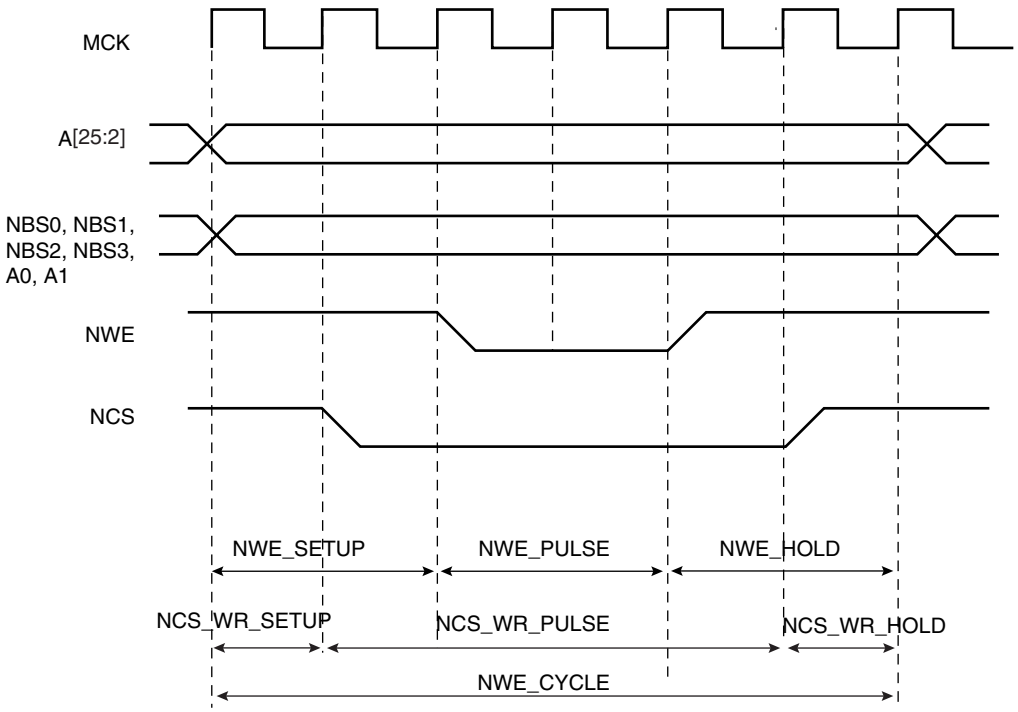
The NWE waveforms apply to all byte-write lines in Byte Write access mode: NWR0 to NWR3.

22.8.3.2 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 22-12. Write Cycle



22.8.3.3 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:

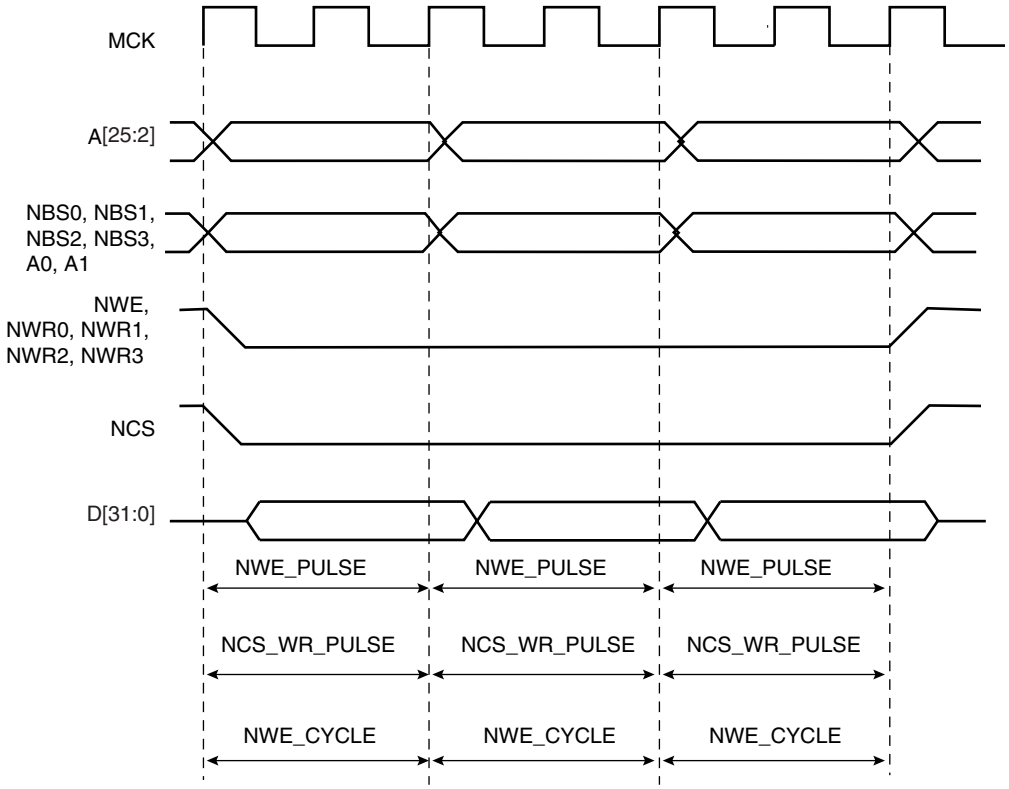
NWE_HOLD = NWE_CYCLE - NWE_SETUP - NWE_PULSE

NCS_WR_HOLD = NWE_CYCLE - NCS_WR_SETUP - NCS_WR_PULSE

22.8.3.4 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 22-13). 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 22-13. Null Setup and Hold Values of NCS and NWE in Write Cycle



22.8.3.5 Null Pulse

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

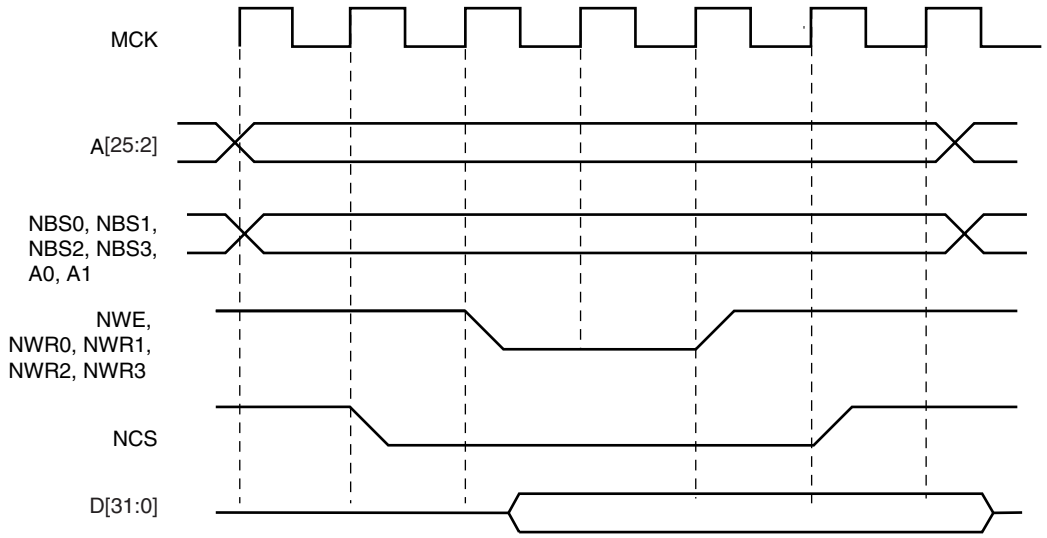
22.8.4 Write Mode

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

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

Figure 22-14 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.

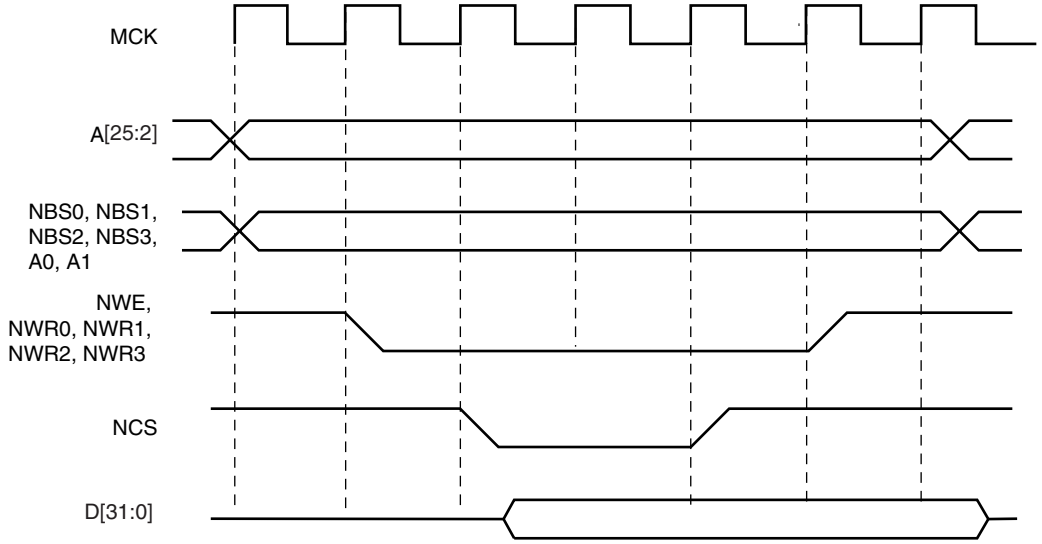
Figure 22-14. WRITE_MODE = 1. The write operation is controlled by NWE



22.8.4.2 Write is Controlled by NCS (WRITE_MODE = 0)

Figure 22-15 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 22-15. WRITE_MODE = 0. The write operation is controlled by NCS



•

22.8.5 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 22-4 shows how the timing parameters are coded and their permitted range.

Table 22-4. 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

22.8.6 Reset Values of Timing Parameters

Table 22-5 gives the default value of timing parameters at reset.

Table 22-5. 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

22.8.7 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 207.

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.

22.9 Automatic Wait States

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

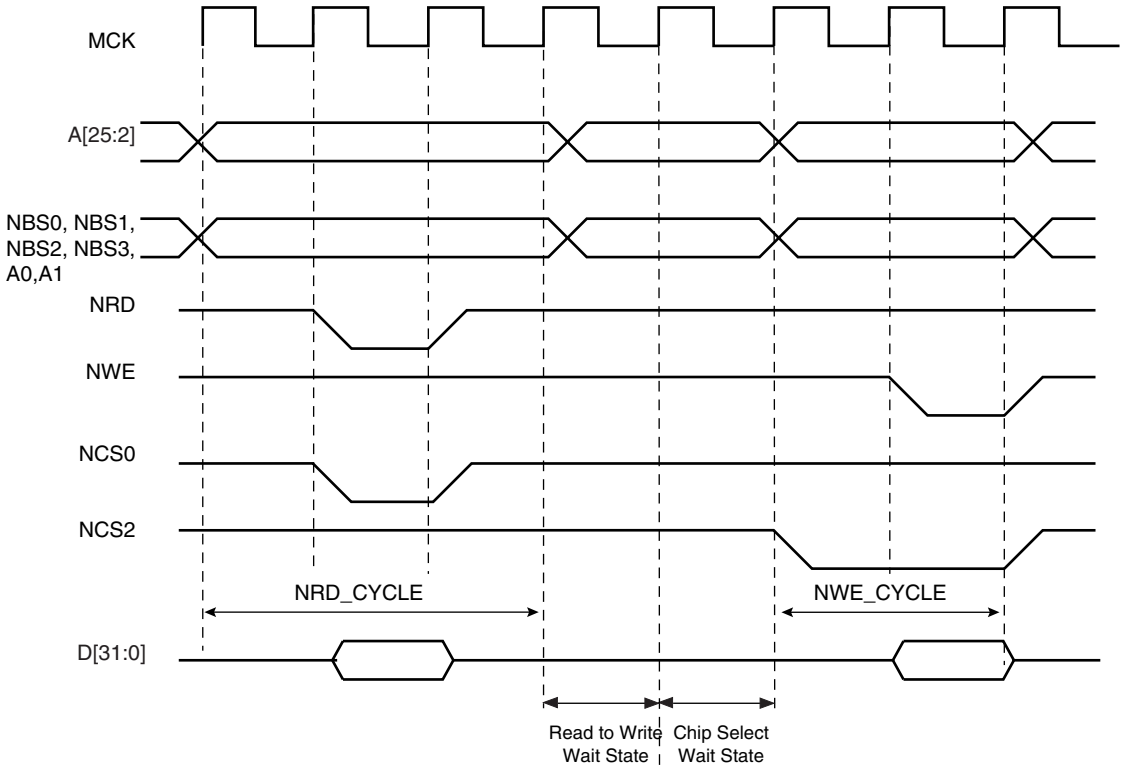
22.9.1 Chip Select Wait States

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

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

Figure 22-16 illustrates a chip select wait state between access on Chip Select 0 and Chip Select 2.

Figure 22-16. Chip Select Wait State between a Read Access on NCS0 and a Write Access on NCS2



22.9.2 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 22-17).
- 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 22-18). 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 22-19.

Figure 22-17. Early Read Wait State: Write with No Hold Followed by Read with No Setup

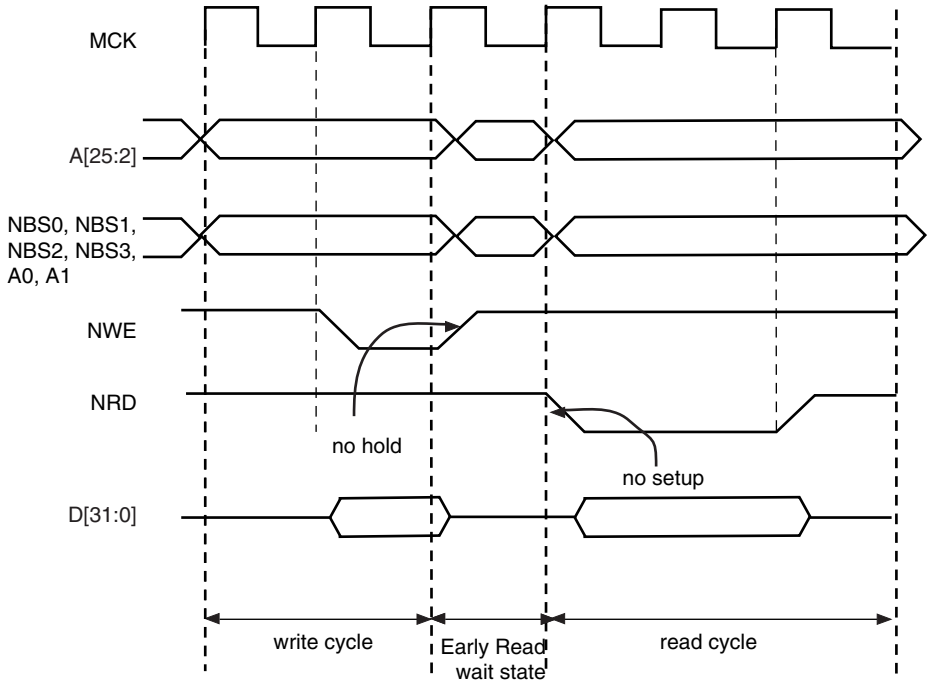


Figure 22-18. Early Read Wait State: NCS Controlled Write with No Hold Followed by a Read with No NCS Setup

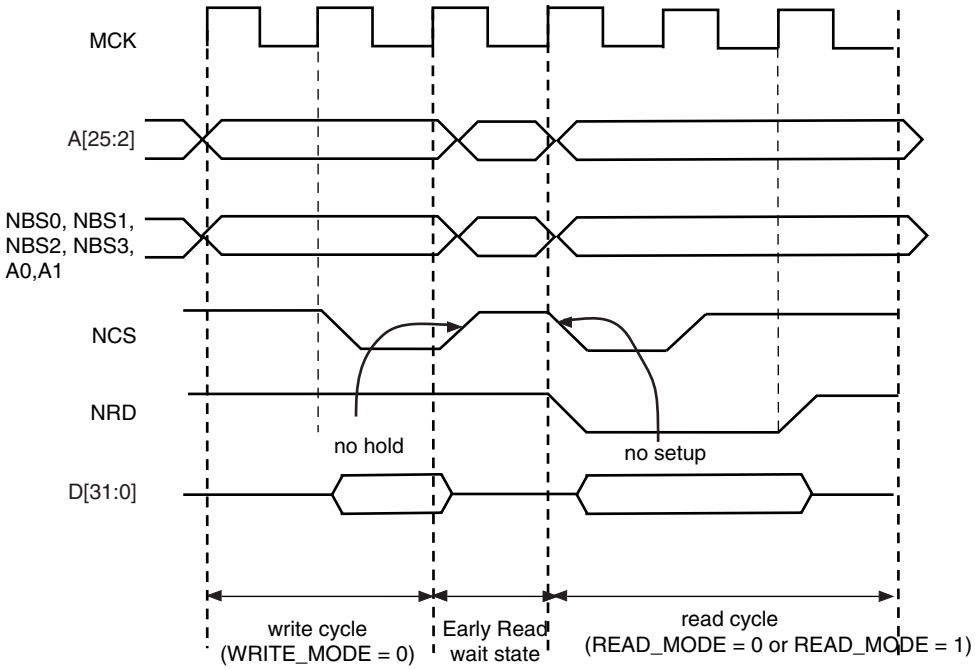
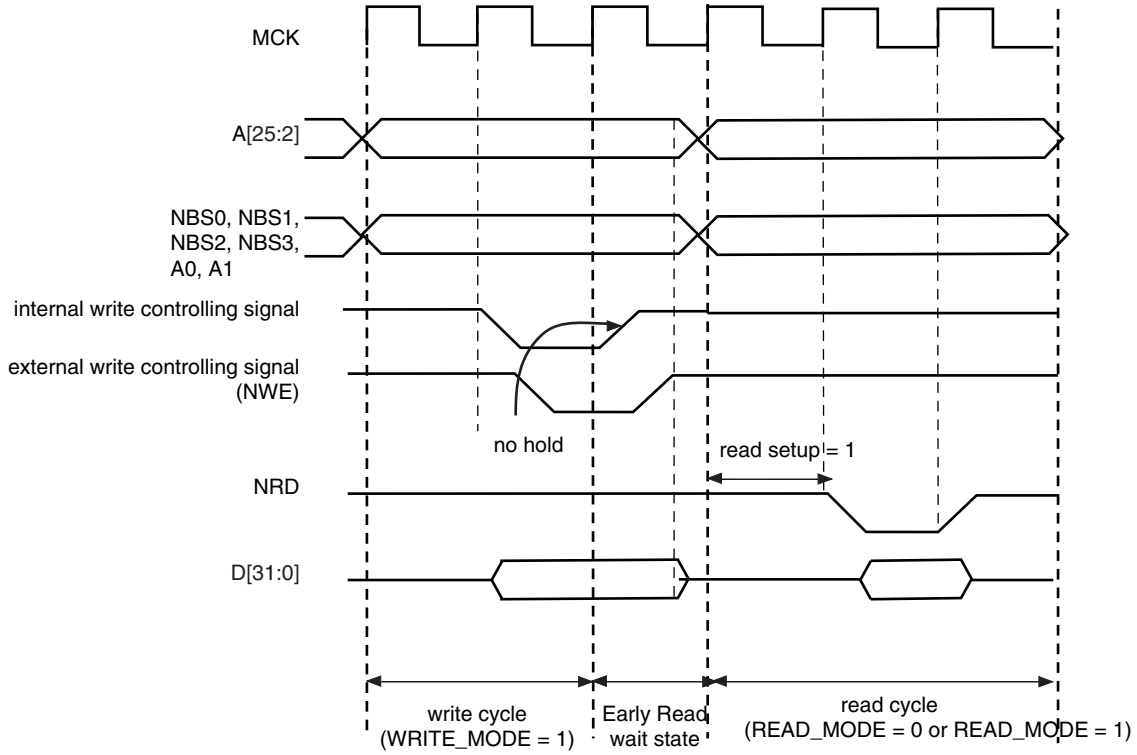


Figure 22-19. Early Read Wait State: NWE-controlled Write with No Hold Followed by a Read with one Set-up Cycle



22.9.3 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.

22.9.3.1 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.

22.9.3.2 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 221).

22.9.4 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 22-16 on page 207.

22.10 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.

22.10.1 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 22-20 illustrates the Data Float Period in NRD-controlled mode (READ_MODE =1), assuming a data float period of 2 cycles (TDF_CYCLES = 2). Figure 22-21 shows the read operation when controlled by NCS (READ_MODE = 0) and the TDF_CYCLES parameter equals 3.

Figure 22-20. TDF Period in NRD Controlled Read Access (TDF = 2)

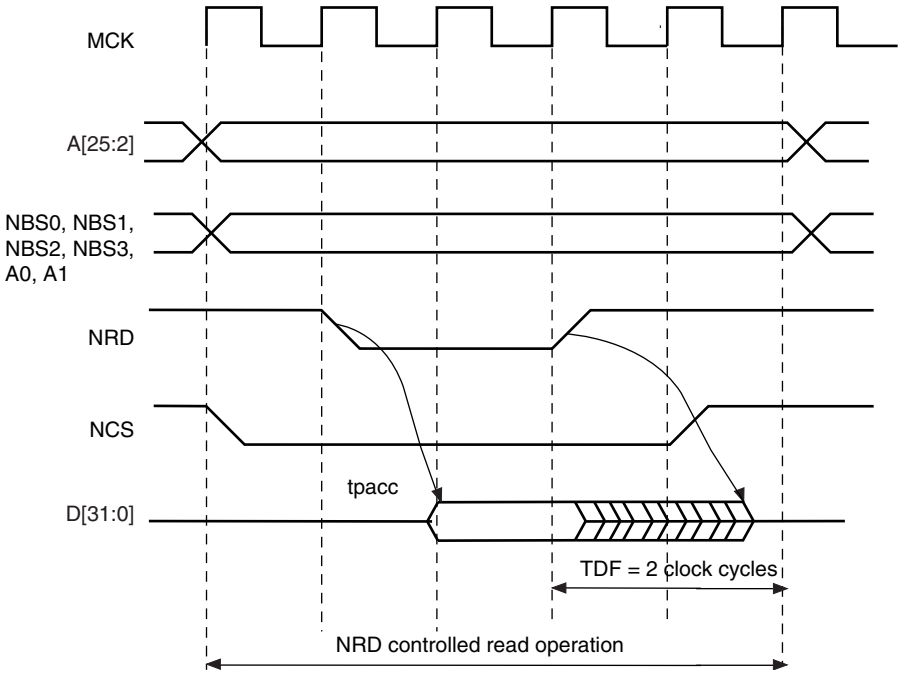
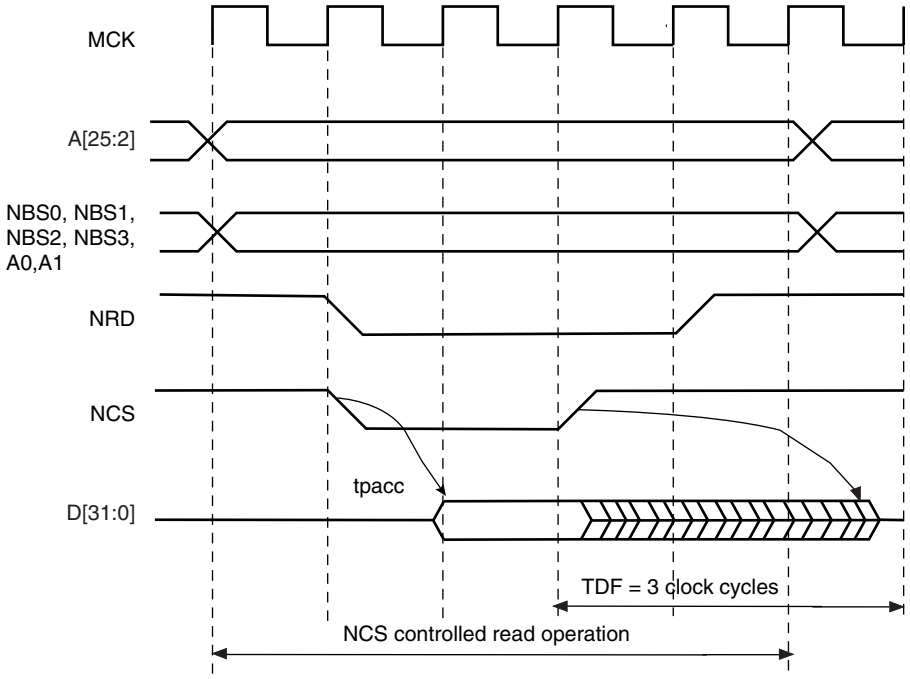


Figure 22-21. TDF Period in NCS Controlled Read Operation (TDF = 3)



22.10.2 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.

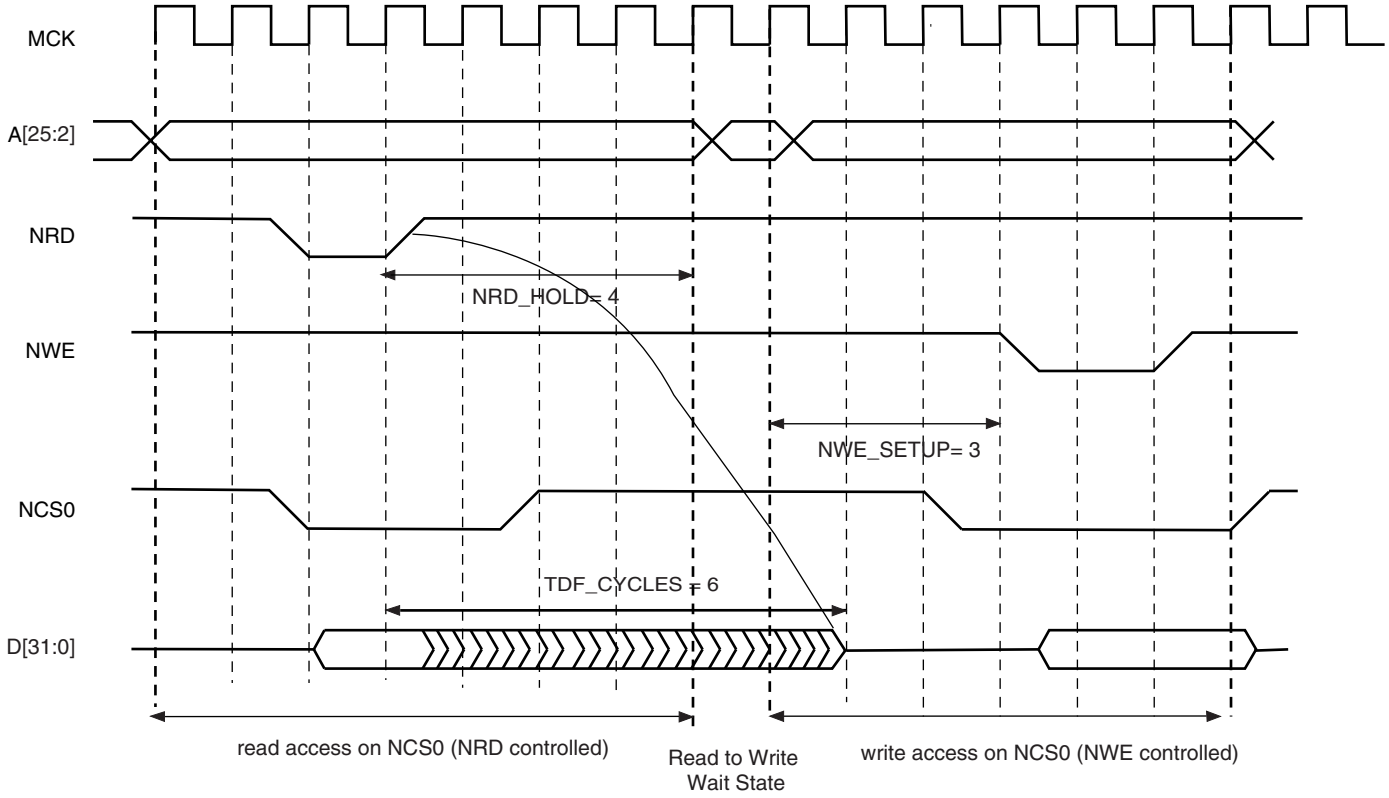
Figure 22-22 shows a read access controlled by NRD, followed by a write access controlled by NWE, on Chip Select 0. Chip Select 0 has been programmed with:

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



22.10.3 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 22-23, Figure 22-24 and Figure 22-25 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 22-23. TDF Optimization Disabled (TDF Mode = 0). TDF wait states between 2 read accesses on different chip selects

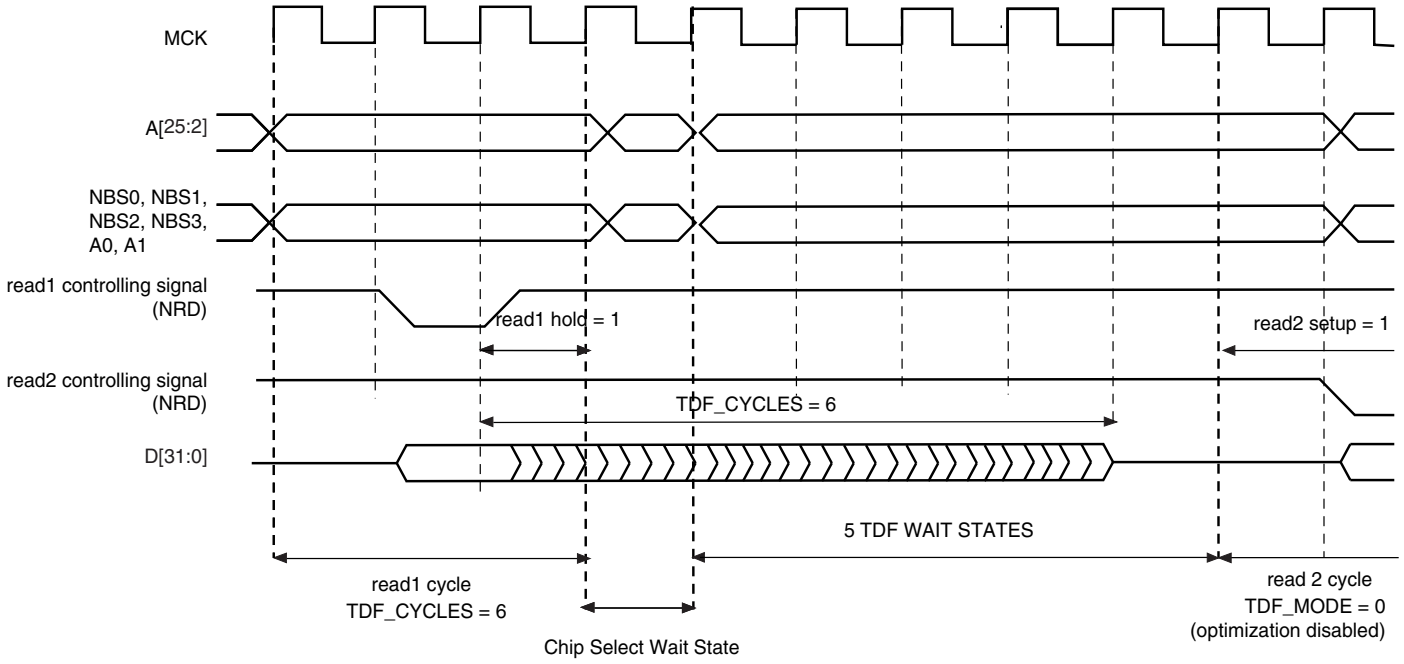


Figure 22-24. TDF Mode = 0: TDF wait states between a read and a write access on different chip selects

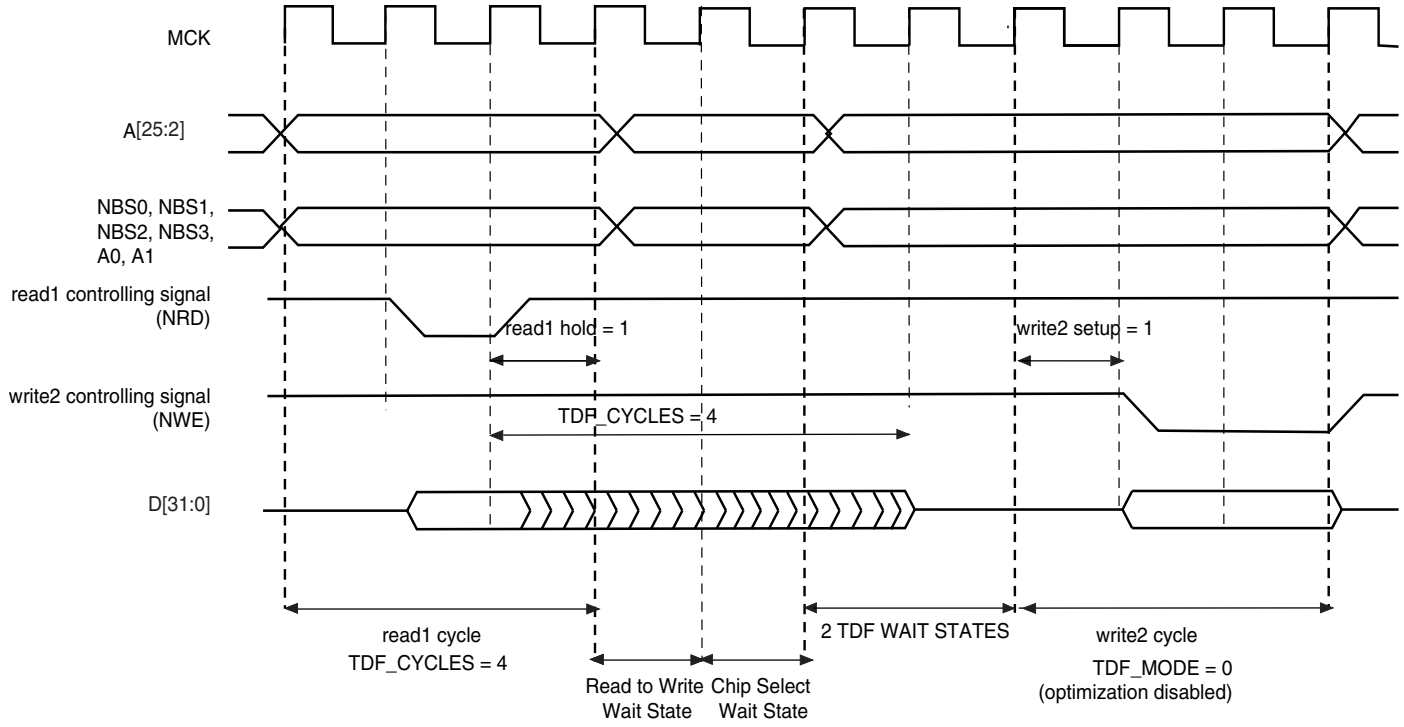
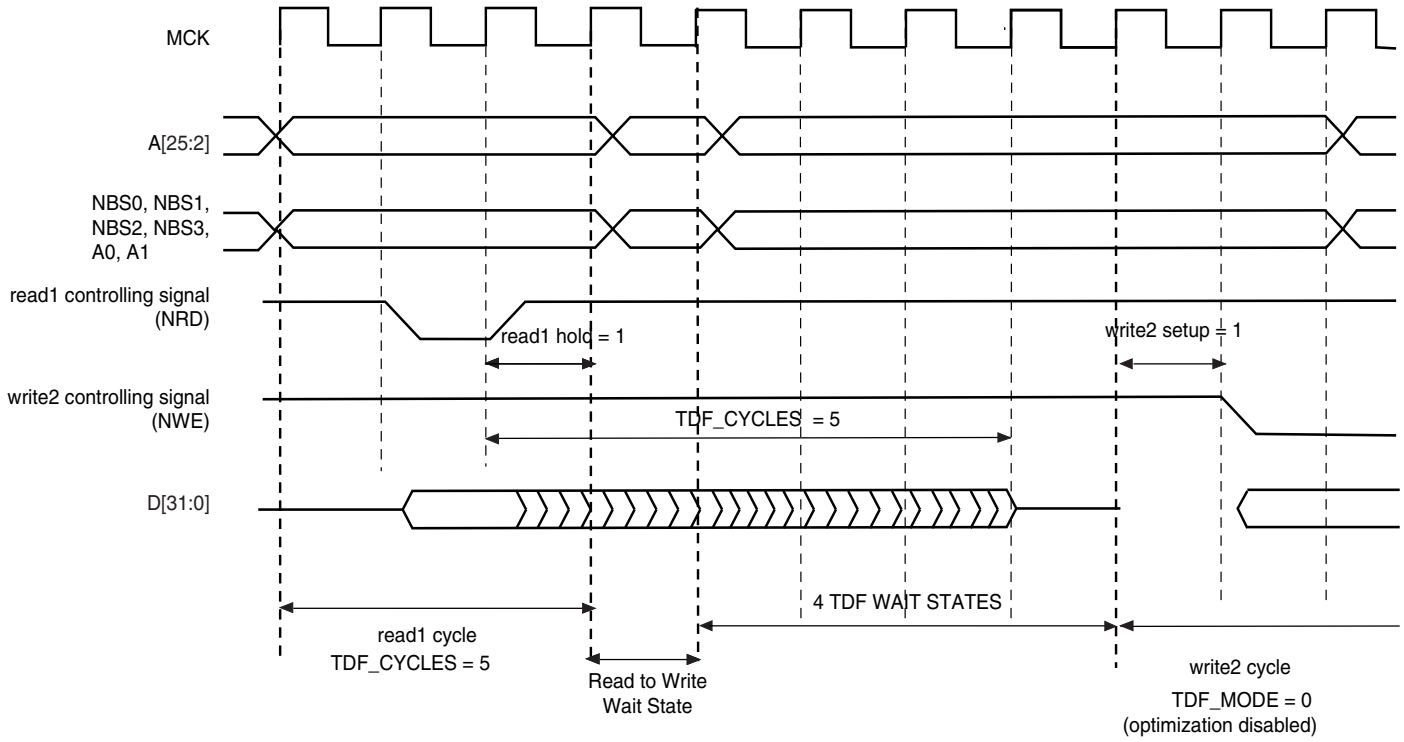


Figure 22-25. TDF Mode = 0: TDF wait states between read and write accesses on the same chip select



22.11 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.

22.11.1 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 224), or in Slow Clock Mode (“Slow Clock Mode” on page 221).**

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.

22.11.2 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 22-26. 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 22-27.

Figure 22-26. Write Access with NWAIT Assertion in Frozen Mode (EXNW_MODE = 10)

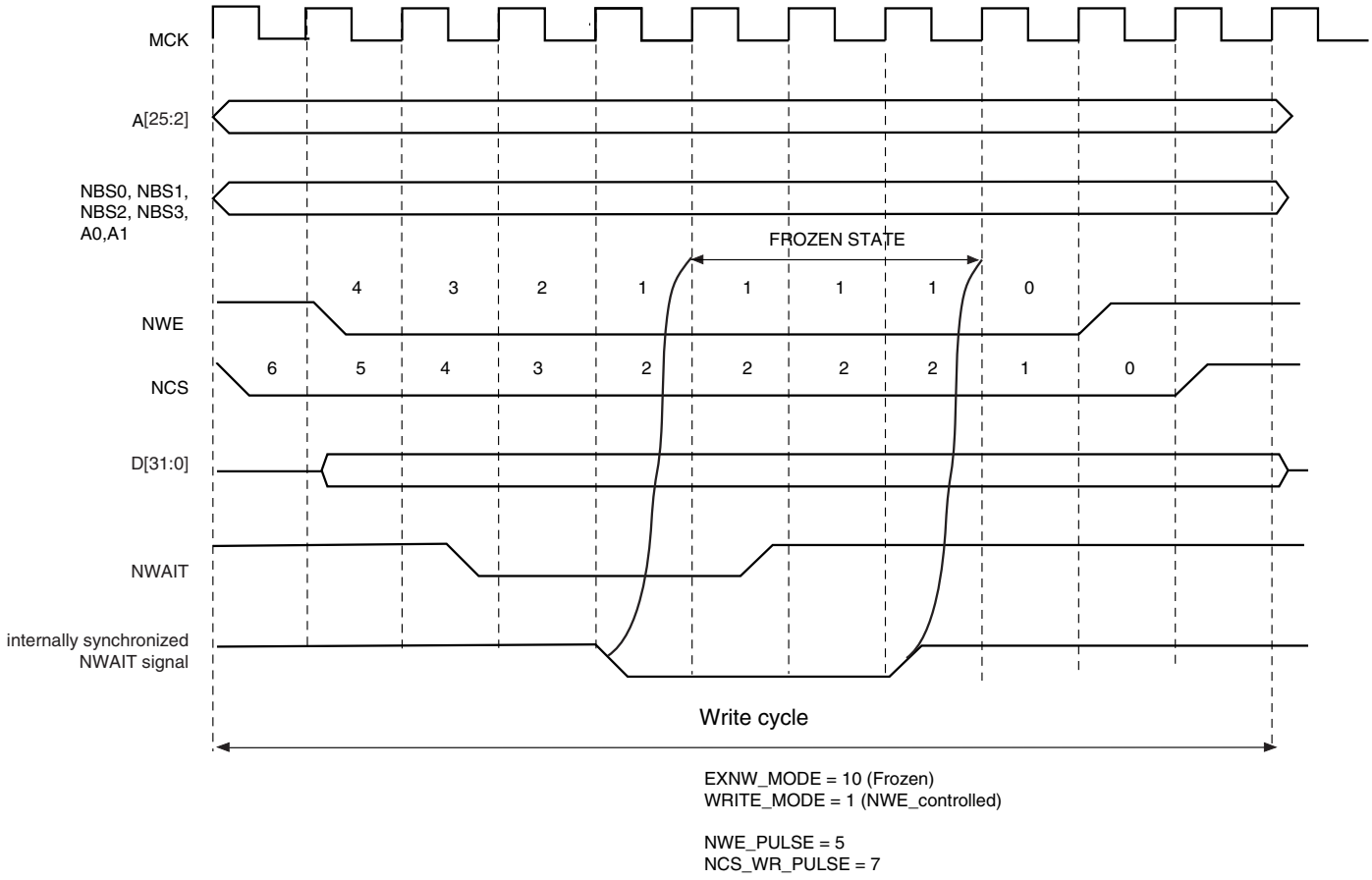
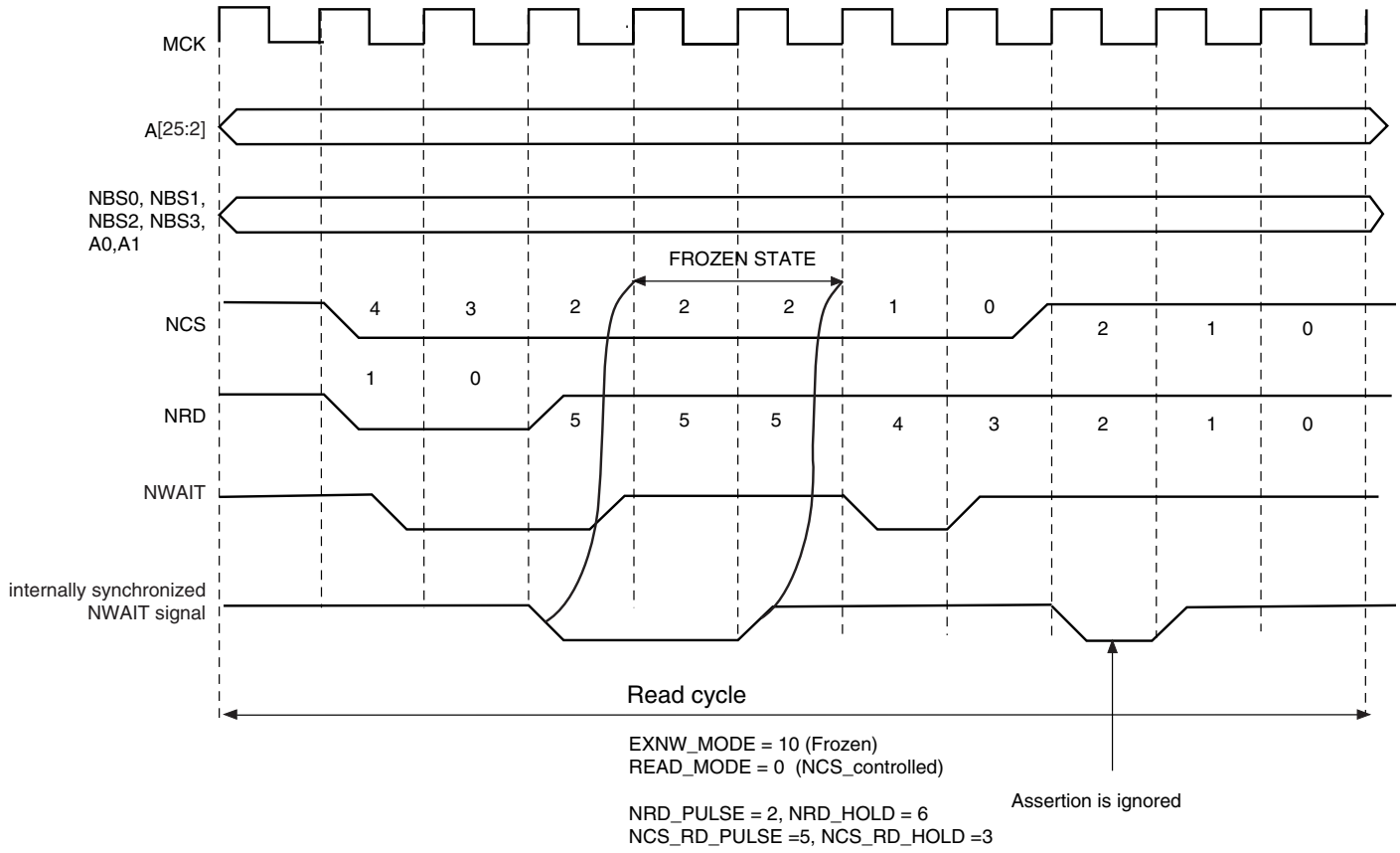


Figure 22-27. Read Access with NWAIT Assertion in Frozen Mode (EXNW_MODE = 10)



22.11.3 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 22-28 and Figure 22-29. 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 22-29.

Figure 22-28. NWAIT Assertion in Write Access: Ready Mode (EXNW_MODE = 11)

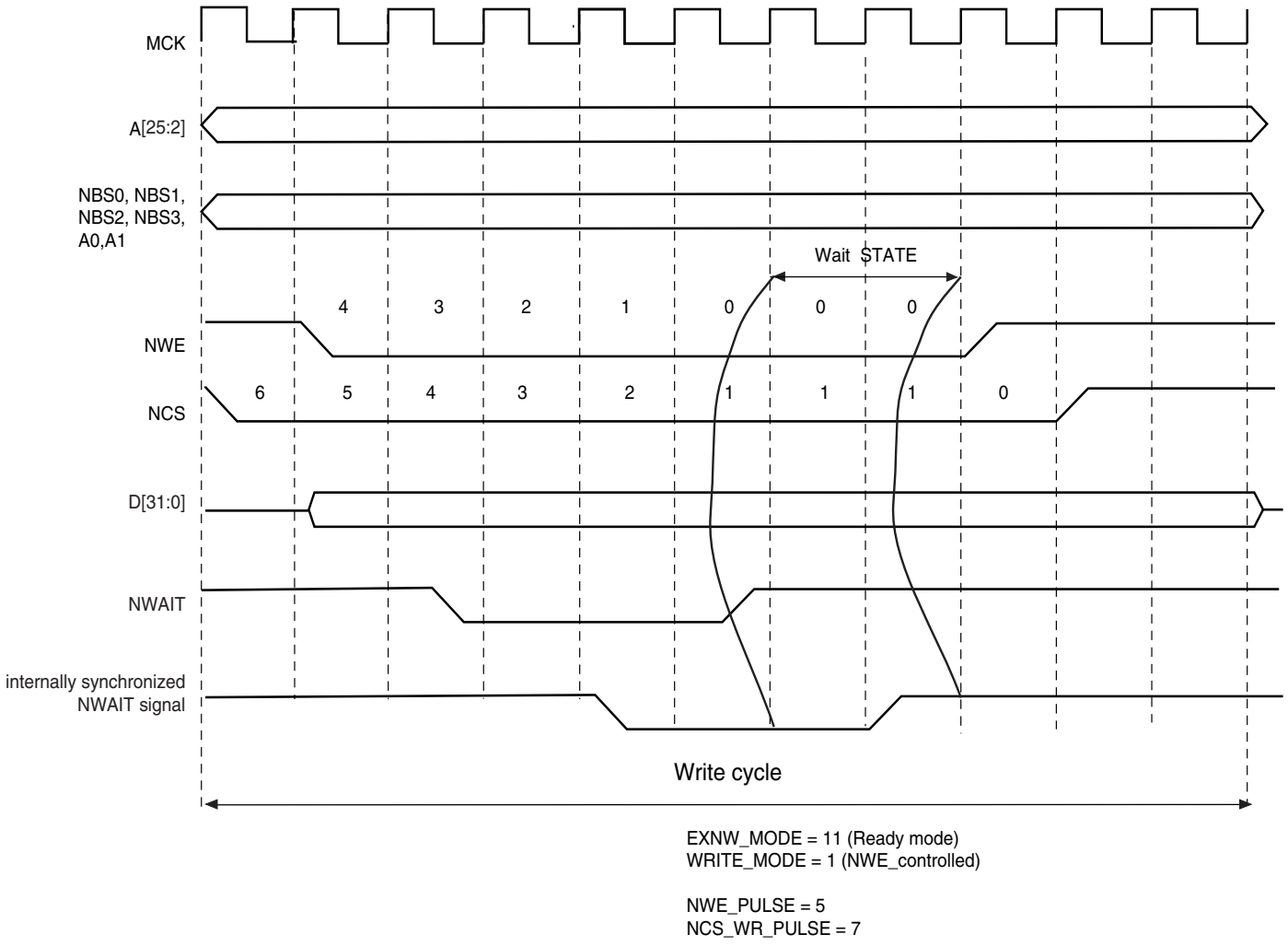
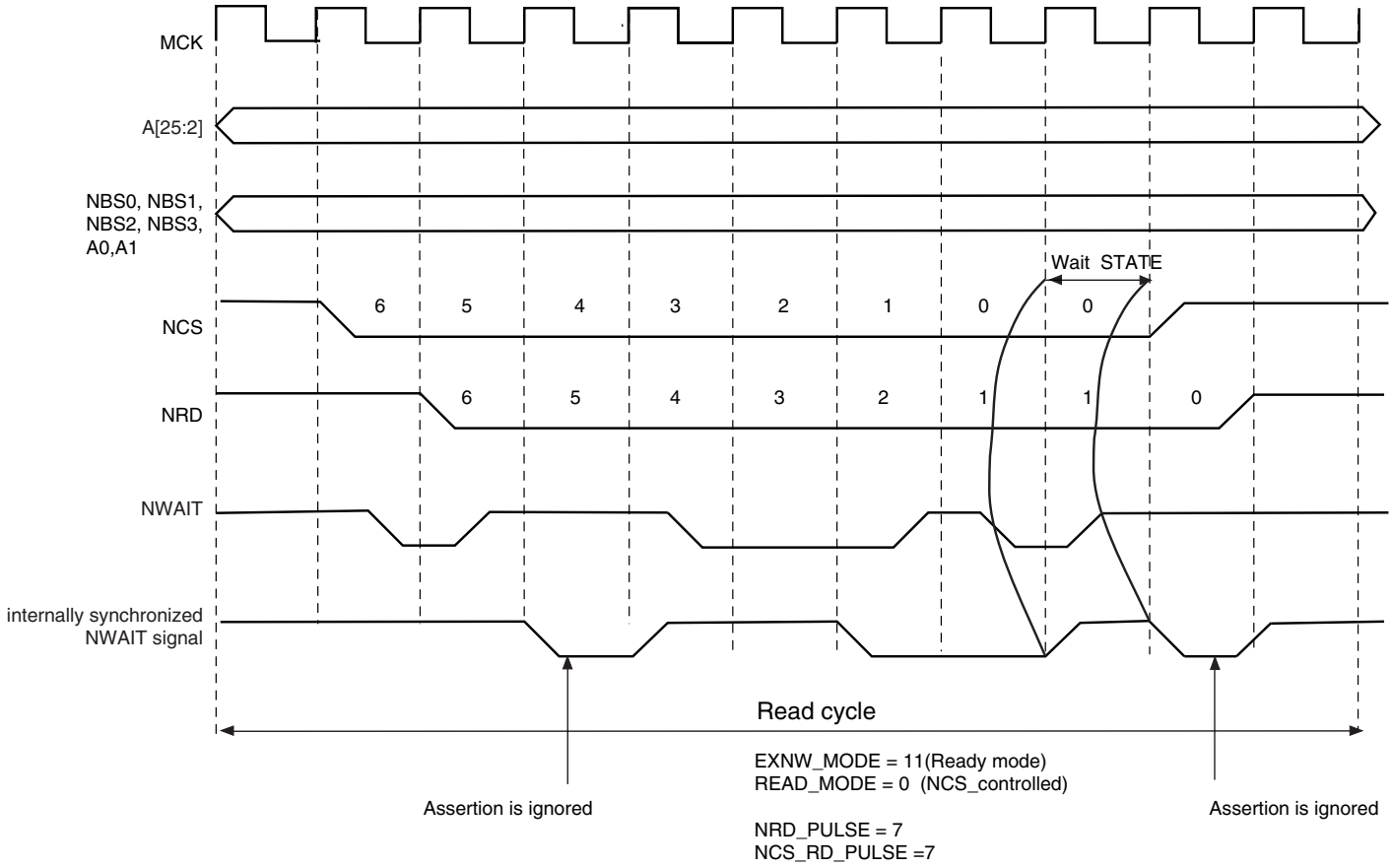


Figure 22-29. NWAIT Assertion in Read Access: Ready Mode (EXNW_MODE = 11)



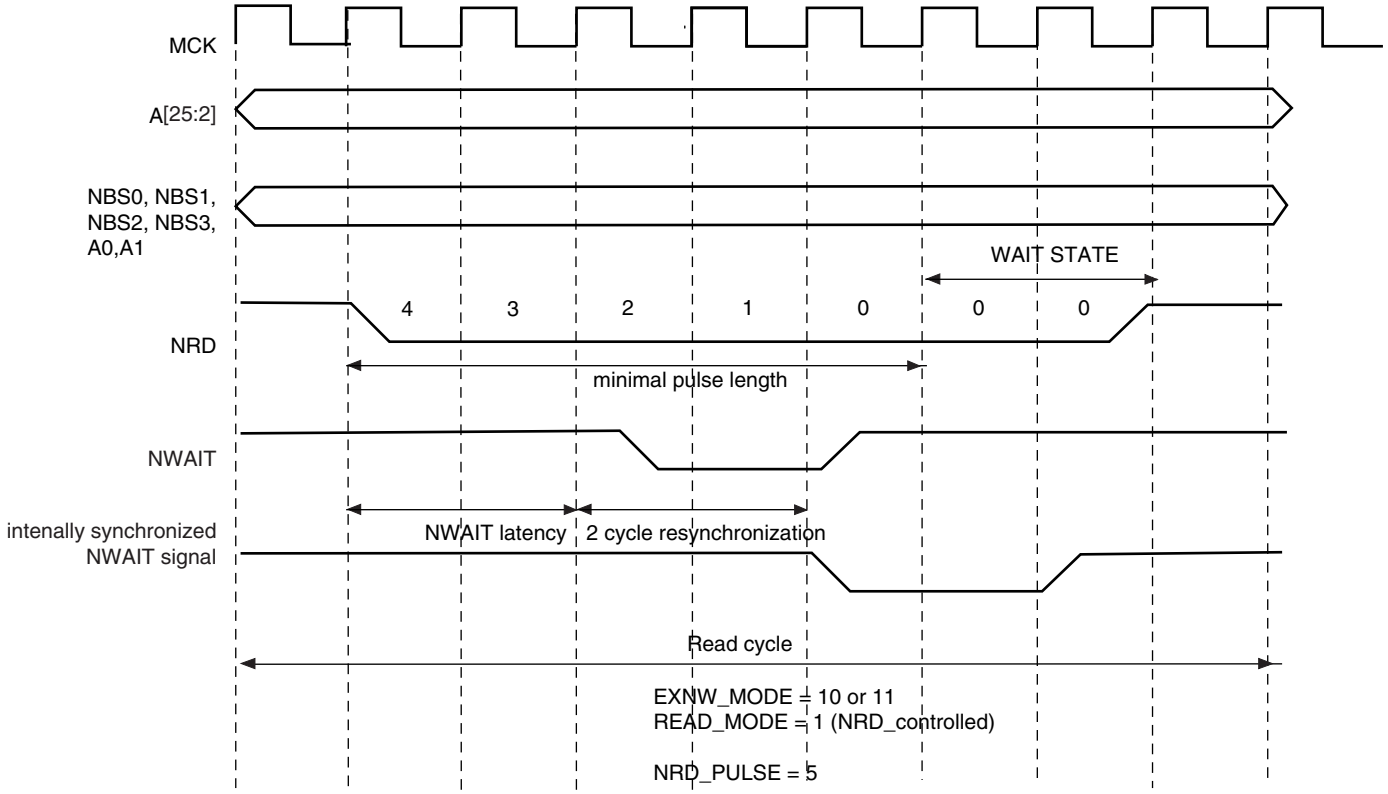
22.11.4 NWAIT Latency and Read/Write Timings

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 22-30.

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 22-30. NWAIT Latency



22.12 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.

22.12.1 Slow Clock Mode Waveforms

Figure 22-31 illustrates the read and write operations in slow clock mode. They are valid on all chip selects. Table 22-6 indicates the value of read and write parameters in slow clock mode.

Figure 22-31. Read/write Cycles in Slow Clock Mode

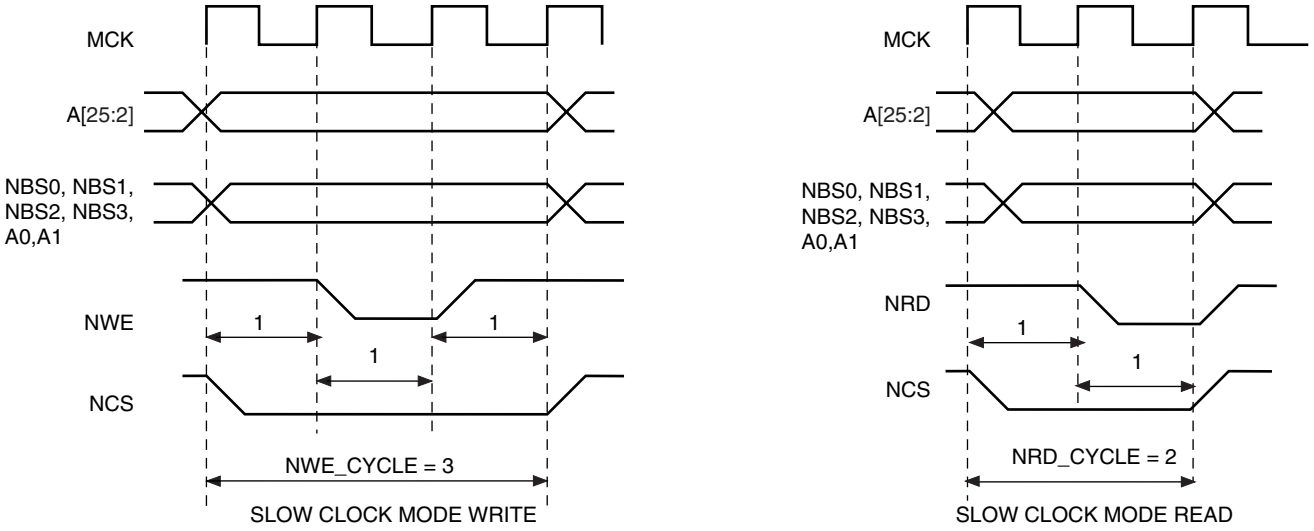


Table 22-6. Read and Write Timing Parameters in Slow Clock Mode

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

22.12.2 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 22-32 on page 222. The external device may not be fast enough to support such timings.

Figure 22-33 illustrates the recommended procedure to properly switch from one mode to the other.

Figure 22-32. Clock Rate Transition Occurs while the SMC is Performing a Write Operation

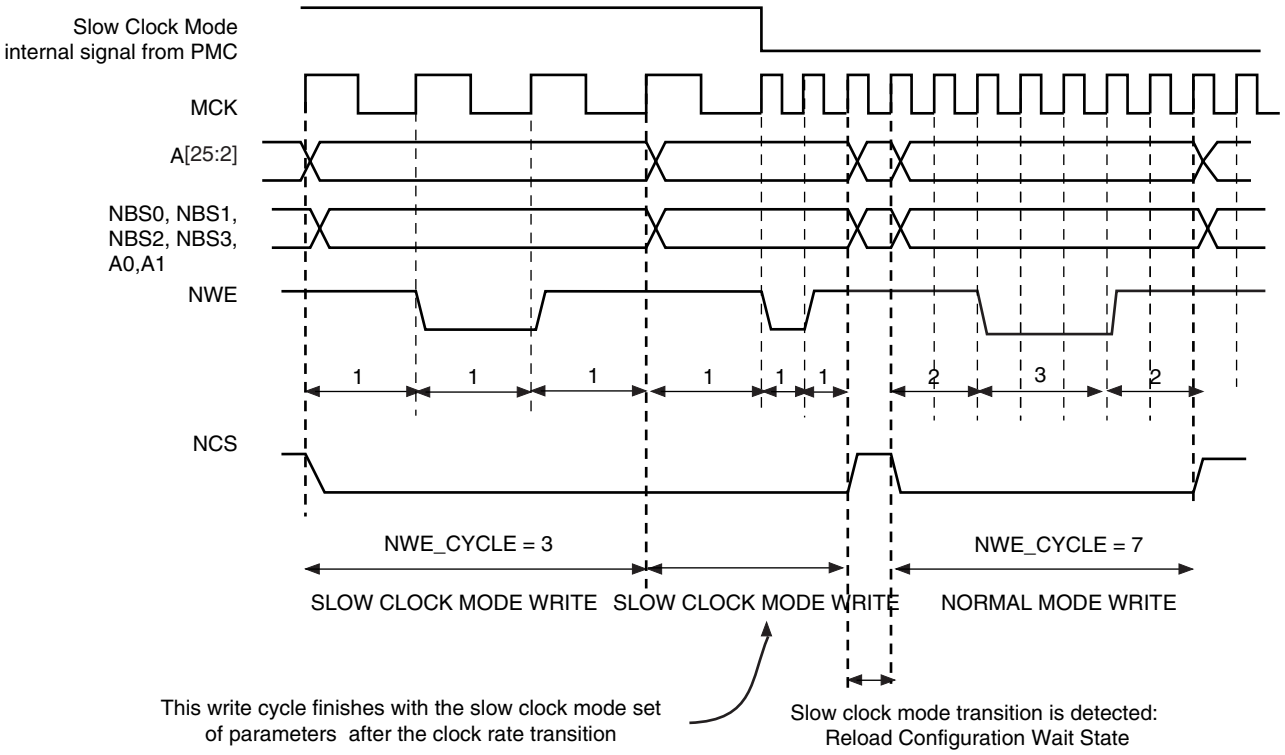
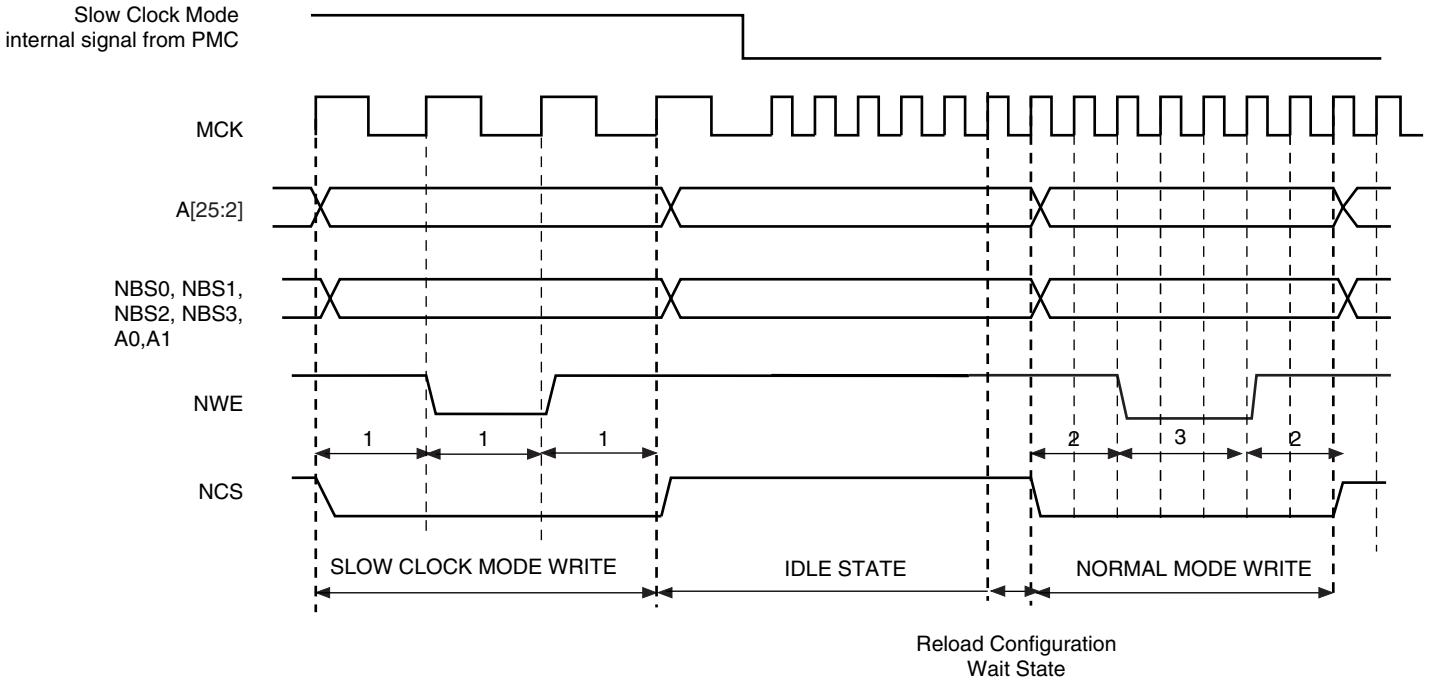


Figure 22-33. Recommended Procedure to Switch from Slow Clock Mode to Normal Mode or from Normal Mode to Slow Clock Mode



22.13 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 22-7.

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 22-34. 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 22-7. 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]

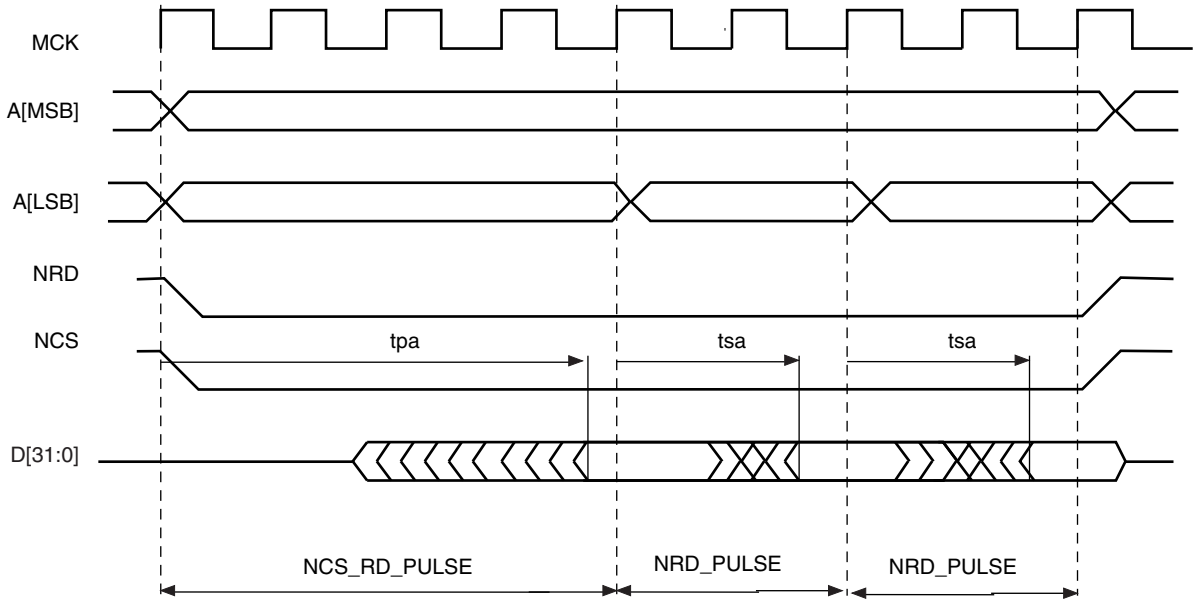
Notes: 1. A denotes the address bus of the memory device

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

22.13.1 Protocol and Timings in Page Mode

Figure 22-34 shows the NRD and NCS timings in page mode access.

Figure 22-34. Page Mode Read Protocol (Address MSB and LSB are defined in Table 22-7)



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 22-8:

Table 22-8. 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}.

22.13.2 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).

22.13.3 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.

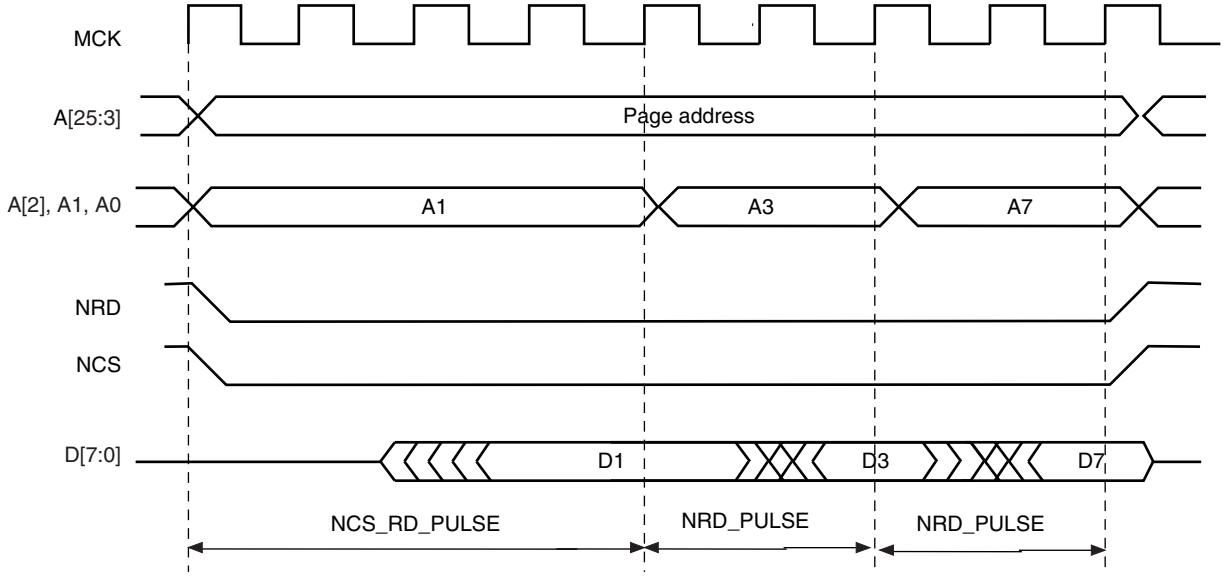
22.13.4 Sequential and Non-sequential Accesses

If the chip select and the MSB of addresses as defined in Table 22-7 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 22-35 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 22-35. Access to Non-sequential Data within the Same Page



22.14 Static Memory Controller (SMC) User Interface

The SMC is programmed using the registers listed in Table 22-9. For each chip select, a set of 4 registers is used to program the parameters of the external device connected on it. In Table 22-9, “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 22-9. 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	-	-	-

22.14.1 SMC Setup Register

Register Name:SMC_SETUP[0..7]

Addresses: 0xFFFFE400 (0)[0], 0xFFFFE410 (0)[1], 0xFFFFE420 (0)[2], 0xFFFFE430 (0)[3], 0xFFFFE440 (0)[4], 0xFFFFE450 (0)[5], 0xFFFFE460 (0)[6], 0xFFFFE470 (0)[7], 0xFFFFEA00 (1)[0], 0xFFFFEA10 (1)[1], 0xFFFFEA20 (1)[2], 0xFFFFEA30 (1)[3], 0xFFFFEA40 (1)[4], 0xFFFFEA50 (1)[5], 0xFFFFEA60 (1)[6], 0xFFFFEA70 (1)[7]

Access Type: Read-write

31	30	29	28	27	26	25	24
–	–	NCS_RD_SETUP					

23	22	21	20	19	18	17	16
–	–	NRD_SETUP					

15	14	13	12	11	10	9	8
–	–	NCS_WR_SETUP					

7	6	5	4	3	2	1	0
–	–	NWE_SETUP					

- **NWE_SETUP: NWE Setup Length**

The NWE signal setup length is defined as:

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

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

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

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

- **NRD_SETUP: NRD Setup Length**

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

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

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

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

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

22.14.2 SMC Pulse Register

Register Name:SMC_PULSE[0..7]

Addresses: 0xFFFFE404 (0)[0], 0xFFFFE414 (0)[1], 0xFFFFE424 (0)[2], 0xFFFFE434 (0)[3], 0xFFFFE444 (0)[4], 0xFFFFE454 (0)[5], 0xFFFFE464 (0)[6], 0xFFFFE474 (0)[7], 0xFFFFEA04 (1)[0], 0xFFFFEA14 (1)[1], 0xFFFFEA24 (1)[2], 0xFFFFEA34 (1)[3], 0xFFFFEA44 (1)[4], 0xFFFFEA54 (1)[5], 0xFFFFEA64 (1)[6], 0xFFFFEA74 (1)[7]

Access Type: Read-write

31	30	29	28	27	26	25	24
–	NCS_RD_PULSE						

23	22	21	20	19	18	17	16
–	NRD_PULSE						

15	14	13	12	11	10	9	8
–	NCS_WR_PULSE						

7	6	5	4	3	2	1	0
–	NWE_PULSE						

- **NWE_PULSE: NWE Pulse Length**

The NWE signal pulse length is defined as:

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

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

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

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

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

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

- **NRD_PULSE: NRD Pulse Length**

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

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

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

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

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

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

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

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

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

22.14.3 SMC Cycle Register

Register Name:SMC_CYCLE[0..7]

Addresses: 0xFFFFE408 (0)[0], 0xFFFFE418 (0)[1], 0xFFFFE428 (0)[2], 0xFFFFE438 (0)[3], 0xFFFFE448 (0)[4], 0xFFFFE458 (0)[5], 0xFFFFE468 (0)[6], 0xFFFFE478 (0)[7], 0xFFFFEA08 (1)[0], 0xFFFFEA18 (1)[1], 0xFFFFEA28 (1)[2], 0xFFFFEA38 (1)[3], 0xFFFFEA48 (1)[4], 0xFFFFEA58 (1)[5], 0xFFFFEA68 (1)[6], 0xFFFFEA78 (1)[7]

Access Type: Read-write

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

23	22	21	20	19	18	17	16
NRD_CYCLE							

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

7	6	5	4	3	2	1	0
NWE_CYCLE							

- **NWE_CYCLE: Total Write Cycle Length**

The total write cycle length is the total duration in clock cycles of the write cycle. It is equal to the sum of the setup, pulse and hold steps of the NWE and NCS signals. It is defined as:

Write cycle length = (NWE_CYCLE[8:7]*256 + NWE_CYCLE[6:0]) clock cycles

- **NRD_CYCLE: Total Read Cycle Length**

The total read cycle length is the total duration in clock cycles of the read cycle. It is equal to the sum of the setup, pulse and hold steps of the NRD and NCS signals. It is defined as:

Read cycle length = (NRD_CYCLE[8:7]*256 + NRD_CYCLE[6:0]) clock cycles

22.14.4 SMC MODE Register

Register Name:SMC_MODE[0..7]

Addresses: 0xFFFFE40C (0)[0], 0xFFFFE41C (0)[1], 0xFFFFE42C (0)[2], 0xFFFFE43C (0)[3], 0xFFFFE44C (0)[4], 0xFFFFE45C (0)[5], 0xFFFFE46C (0)[6], 0xFFFFE47C (0)[7], 0xFFFFEA0C (1)[0], 0xFFFFEA1C (1)[1], 0xFFFFEA2C (1)[2], 0xFFFFEA3C (1)[3], 0xFFFFEA4C (1)[4], 0xFFFFEA5C (1)[5], 0xFFFFEA6C (1)[6], 0xFFFFEA7C (1)[7]

Access Type: Read-write

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

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

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

7	6	5	4	3	2	1	0
–	–	EXNW_MODE		–	–	WRITE_MODE	READ_MODE

- **READ_MODE:**

1: The read operation is controlled by the NRD signal.

  – If TDF cycles are programmed, the external bus is marked busy after the rising edge of NRD.

  – If TDF optimization is enabled (TDF_MODE =1), TDF wait states are inserted after the setup of NRD.

0: The read operation is controlled by the NCS signal.

  – If TDF cycles are programmed, the external bus is marked busy after the rising edge of NCS.

  – If TDF optimization is enabled (TDF_MODE =1), TDF wait states are inserted after the setup of NCS.

- **WRITE_MODE**

1: The write operation is controlled by the NWE signal.

  – If TDF optimization is enabled (TDF_MODE =1), TDF wait states will be inserted after the setup of NWE.

0: The write operation is controlled by the NCS signal.

  – If TDF optimization is enabled (TDF_MODE =1), TDF wait states will be inserted after the setup of NCS.

- **EXNW_MODE: NWAIT Mode**

The NWAIT signal is used to extend the current read or write signal. It is only taken into account during the pulse phase of the read and write controlling signal. When the use of NWAIT is enabled, at least one cycle hold duration must be programmed for the read and write controlling signal.

EXNW_MODE		NWAIT Mode
0	0	Disabled
0	1	Reserved
1	0	Frozen Mode
1	1	Ready Mode

- Disabled Mode: The NWAIT input signal is ignored on the corresponding Chip Select.
- Frozen Mode: If asserted, the NWAIT signal freezes the current read or write cycle. After deassertion, the read/write cycle is resumed from the point where it was stopped.

- Ready Mode: The NWAIT signal indicates the availability of the external device at the end of the pulse of the controlling read or write signal, to complete the access. If high, the access normally completes. If low, the access is extended until NWAIT returns high.

- **BAT: Byte Access Type**

This field is used only if DBW defines a 16- or 32-bit data bus.

- 1: Byte write access type:
  - Write operation is controlled using NCS, NWR0, NWR1, NWR2, NWR3.
  - Read operation is controlled using NCS and NRD.
- 0: Byte select access type:
  - Write operation is controlled using NCS, NWE, NBS0, NBS1, NBS2 and NBS3
  - Read operation is controlled using NCS, NRD, NBS0, NBS1, NBS2 and NBS3

- **DBW: Data Bus Width**

DBW		Data Bus Width
0	0	8-bit bus
0	1	16-bit bus
1	0	32-bit bus
1	1	Reserved

- **TDF_CYCLES: Data Float Time**

This field gives the integer number of clock cycles required by the external device to release the data after the rising edge of the read controlling signal. The SMC always provide one full cycle of bus turnaround after the TDF_CYCLES period. The external bus cannot be used by another chip select during TDF_CYCLES + 1 cycles. From 0 up to 15 TDF_CYCLES can be set.

- **TDF_MODE: TDF Optimization**

1: TDF optimization is enabled.

  - The number of TDF wait states is optimized using the setup period of the next read/write access.

0: TDF optimization is disabled.

  - The number of TDF wait states is inserted before the next access begins.

- **PMEN: Page Mode Enabled**

1: Asynchronous burst read in page mode is applied on the corresponding chip select.

0: Standard read is applied.

- **PS: Page Size**

If page mode is enabled, this field indicates the size of the page in bytes.

PS		Page Size
0	0	4-byte page
0	1	8-byte page
1	0	16-byte page
1	1	32-byte page

23. SDRAM Controller (SDRAMC)

23.1 Overview

The SDRAM Controller (SDRAMC) extends the memory capabilities of a chip by providing the interface to an external 16-bit or 32-bit SDRAM device. The page size supports ranges from 2048 to 8192 and the number of columns from 256 to 2048. It supports byte (8-bit), half-word (16-bit) and word (32-bit) accesses.

The SDRAM Controller supports a read or write burst length of one location. It keeps track of the active row in each bank, thus maximizing SDRAM performance, e.g., the application may be placed in one bank and data in the other banks. So as to optimize performance, it is advisable to avoid accessing different rows in the same bank.

The SDRAM controller supports a CAS latency of 1, 2 or 3 and optimizes the read access depending on the frequency.

The different modes available - self-refresh, power-down and deep power-down modes - minimize power consumption on the SDRAM device.

23.2 I/O Lines Description

Table 23-1. I/O Line Description

Name	Description	Type	Active Level
SDCK	SDRAM Clock	Output	
SDCKE	SDRAM Clock Enable	Output	High
SDCS	SDRAM Controller Chip Select	Output	Low
BA[1:0]	Bank Select Signals	Output	
RAS	Row Signal	Output	Low
CAS	Column Signal	Output	Low
SDWE	SDRAM Write Enable	Output	Low
NBS[3:0]	Data Mask Enable Signals	Output	Low
SDRAMC_A[12:0]	Address Bus	Output	
D[31:0]	Data Bus	I/O	

23.3 Application Example

23.3.1 Software Interface

The SDRAM address space is organized into banks, rows, and columns. The SDRAM controller allows mapping different memory types according to the values set in the SDRAMC configuration register.

The SDRAM Controller's function is to make the SDRAM device access protocol transparent to the user. Table 23-2 to Table 23-7 illustrate the SDRAM device memory mapping seen by the user in correlation with the device structure. Various configurations are illustrated.

23.3.1.1 32-bit Memory Data Bus Width

Table 23-2. SDRAM Configuration Mapping: 2K Rows, 256/512/1024/2048 Columns

CPU Address Line																										
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
					Bk[1:0]		Row[10:0]											Column[7:0]								M[1:0]	
				Bk[1:0]			Row[10:0]											Column[8:0]								M[1:0]	
			Bk[1:0]				Row[10:0]											Column[9:0]								M[1:0]	
		Bk[1:0]					Row[10:0]											Column[10:0]								M[1:0]	

Table 23-3. SDRAM Configuration Mapping: 4K Rows, 256/512/1024/2048 Columns

CPU Address Line																										
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
				Bk[1:0]			Row[11:0]											Column[7:0]								M[1:0]	
			Bk[1:0]				Row[11:0]											Column[8:0]								M[1:0]	
		Bk[1:0]					Row[11:0]											Column[9:0]								M[1:0]	
	Bk[1:0]						Row[11:0]											Column[10:0]								M[1:0]	

Table 23-4. SDRAM Configuration Mapping: 8K Rows, 256/512/1024/2048 Columns

CPU Address Line																										
27	**26**	**25**	**24**	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
			Bk[1:0]				Row[12:0]											Column[7:0]								M[1:0]	
		Bk[1:0]					Row[12:0]											Column[8:0]								M[1:0]	
	Bk[1:0]						Row[12:0]											Column[9:0]								M[1:0]	
Bk[1:0]							Row[12:0]											Column[10:0]								M[1:0]	

Notes: 1. M[1:0] is the byte address inside a 32-bit word.

3. Bk[1] = BA1, Bk[0] = BA0.

23.3.1.2 16-bit Memory Data Bus Width

Table 23-5. SDRAM Configuration Mapping: 2K Rows, 256/512/1024/2048 Columns

CPU Address Line																										
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						Bk[1:0]	Row[10:0]												Column[7:0]								M0
					Bk[1:0]	Row[10:0]													Column[8:0]								M0
				Bk[1:0]	Row[10:0]														Column[9:0]								M0
			Bk[1:0]	Row[10:0]															Column[10:0]								M0

Table 23-6. SDRAM Configuration Mapping: 4K Rows, 256/512/1024/2048 Columns

CPU Address Line																										
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Bk[1:0]	Row[11:0]													Column[7:0]								M0
				Bk[1:0]	Row[11:0]														Column[8:0]								M0
			Bk[1:0]	Row[11:0]															Column[9:0]								M0
		Bk[1:0]	Row[11:0]																Column[10:0]								M0

Table 23-7. SDRAM Configuration Mapping: 8K Rows, 256/512/1024/2048 Columns

CPU Address Line																										
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				Bk[1:0]	Row[12:0]														Column[7:0]								M0
			Bk[1:0]	Row[12:0]															Column[8:0]								M0
		Bk[1:0]	Row[12:0]																Column[9:0]								M0
	Bk[1:0]	Row[12:0]																	Column[10:0]								M0

Notes: 1. M0 is the byte address inside a 16-bit half-word.

4. Bk[1] = BA1, Bk[0] = BA0.

23.4 Product Dependencies

23.4.1 SDRAM Device Initialization

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

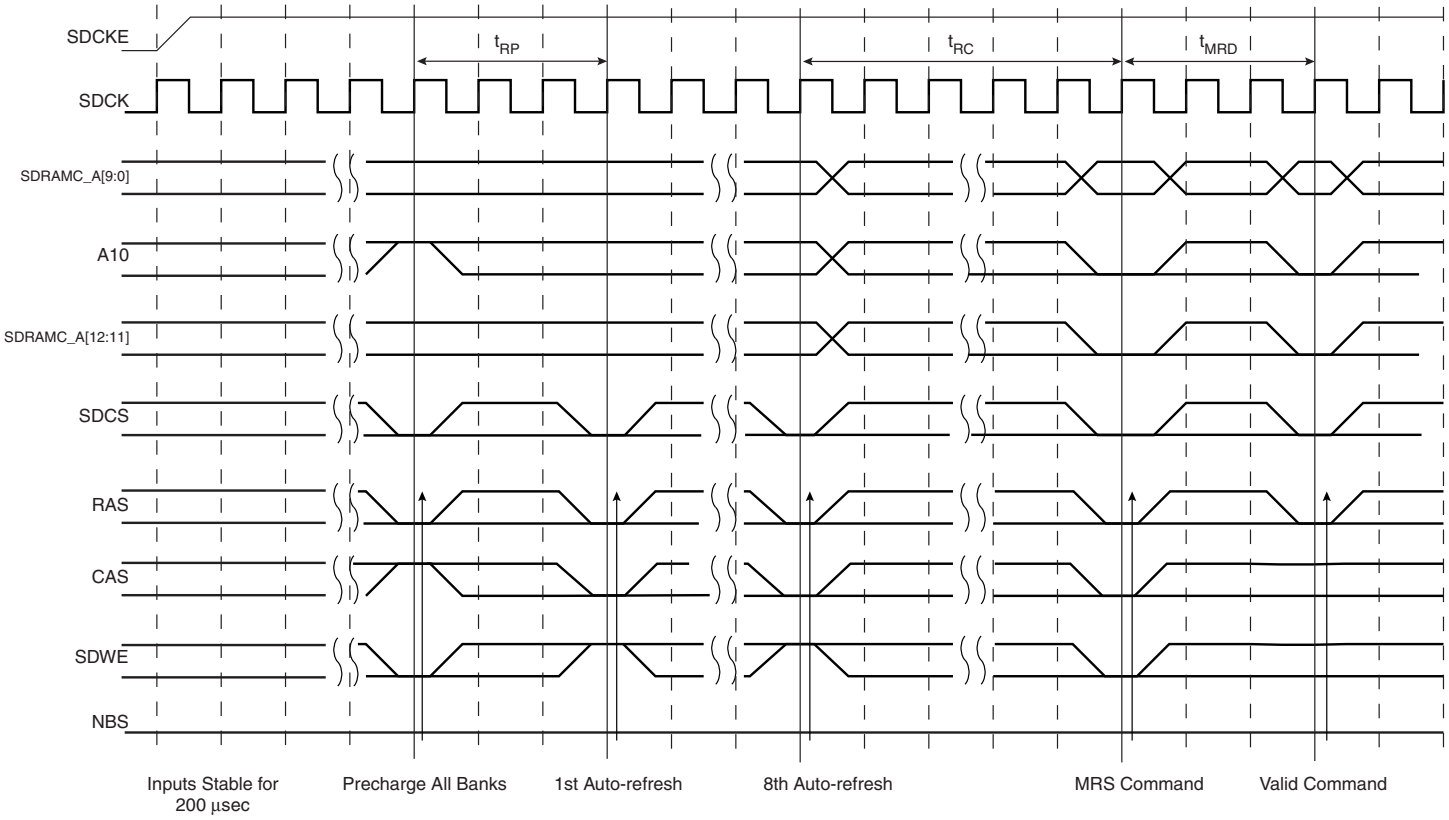
1. SDRAM features must be set in the configuration register: asynchronous timings (TRC, TRAS, etc.), number of columns, rows, CAS latency, and the data bus width.
2. For mobile SDRAM, temperature-compensated self refresh (TCSR), drive strength (DS) and partial array self refresh (PASR) must be set in the Low Power Register.
3. The SDRAM memory type must be set in the Memory Device Register.
4. A minimum pause of 200 µs is provided to precede any signal toggle.
5. [(1)]A NOP command is issued to the SDRAM devices. The application must set Mode to 1 in the Mode Register and perform a write access to any SDRAM address.

6. An All Banks Precharge command is issued to the SDRAM devices. The application must set Mode to 2 in the Mode Register and perform a write access to any SDRAM address.
7. Eight auto-refresh (CBR) cycles are provided. The application must set the Mode to 4 in the Mode Register and perform a write access to any SDRAM location eight times.
8. A Mode Register set (MRS) cycle is issued to program the parameters of the SDRAM devices, in particular CAS latency and burst length. The application must set Mode to 3 in the Mode Register and perform a write access to the SDRAM. The write address must be chosen so that BA[1:0] are set to 0. For example, with a 16-bit 128 MB SDRAM (12 rows, 9 columns, 4 banks) bank address, the SDRAM write access should be done at the address 0x20000000.
9. For mobile SDRAM initialization, an Extended Mode Register set (EMRS) cycle is issued to program the SDRAM parameters (TCSR, PASR, DS). The application must set Mode to 5 in the Mode Register and perform a write access to the SDRAM. The write address must be chosen so that BA[1] or BA[0] are set to 1. For example, with a 16-bit 128 MB SDRAM, (12 rows, 9 columns, 4 banks) bank address the SDRAM write access should be done at the address 0x20800000 or 0x20400000.
10. The application must go into Normal Mode, setting Mode to 0 in the Mode Register and performing a write access at any location in the SDRAM.
11. Write the refresh rate into the count field in the SDRAMC Refresh Timer register. (Refresh rate = delay between refresh cycles). The SDRAM device requires a refresh every 15.625 µs or 7.81 µs. With a 100 MHz frequency, the Refresh Timer Counter Register must be set with the value 1562(15.652 µs x 100 MHz) or 781(7.81 µs x 100 MHz).

After initialization, the SDRAM devices are fully functional.

Note: 1. It is strongly recommended to respect the instructions stated in Step 5 of the initialization process in order to be certain that the subsequent commands issued by the SDRAMC will be taken into account.

Figure 23-1. SDRAM Device Initialization Sequence



23.4.2 I/O Lines

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

23.4.3 Interrupt

The SDRAM Controller interrupt (Refresh Error notification) is connected to the Memory Controller. This interrupt may be ORed with other System Peripheral interrupt lines and is finally provided as the System Interrupt Source (Source 1) to the AIC (Advanced Interrupt Controller).

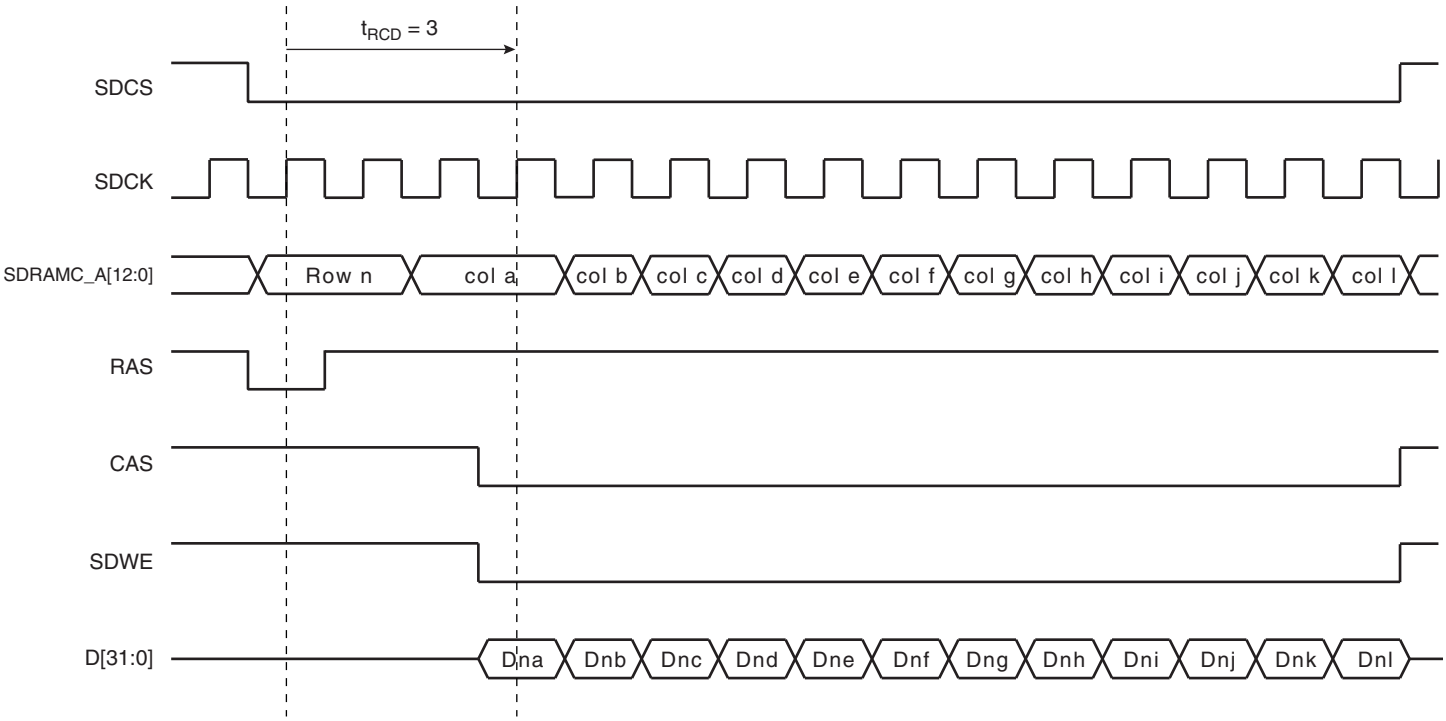
Using the SDRAM Controller interrupt requires the AIC to be programmed first.

23.5 Functional Description

23.5.1 SDRAM Controller Write Cycle

The SDRAM Controller allows burst access or single access. In both cases, the SDRAM controller keeps track of the active row in each bank, thus maximizing performance. To initiate a burst access, the SDRAM Controller uses the transfer type signal provided by the master requesting the access. If the next access is a sequential write access, writing to the SDRAM device is carried out. If the next access is a write-sequential access, but the current access is to a boundary page, or if the next access is in another row, then the SDRAM Controller generates a precharge command, activates the new row and initiates a write command. To comply with SDRAM timing parameters, additional clock cycles are inserted between precharge/active (t_{RP}) commands and active/write (t_{RCD}) commands. For definition of these timing parameters, refer to the "SDRAMC Configuration Register" on page 249. This is described in Figure 23-2 below.

Figure 23-2. Write Burst, 32-bit SDRAM Access



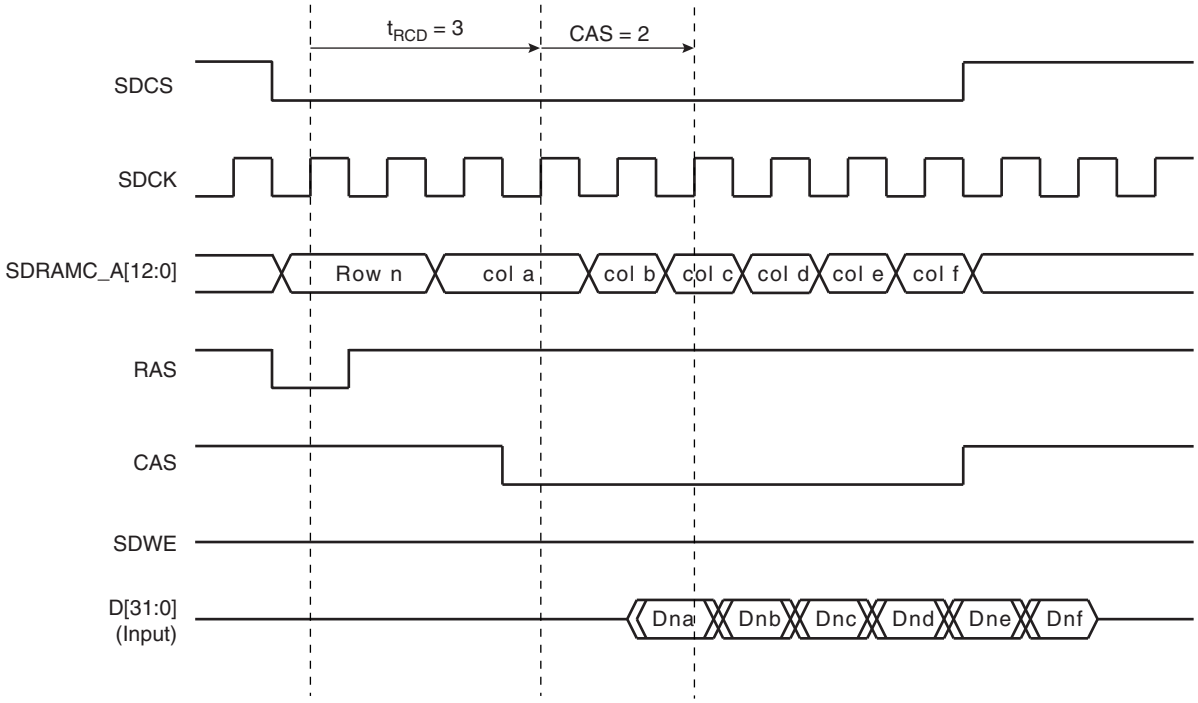
23.5.2 SDRAM Controller Read Cycle

The SDRAM Controller allows burst access, incremental burst of unspecified length or single access. In all cases, the SDRAM Controller keeps track of the active row in each bank, thus maximizing performance of the SDRAM. If row and bank addresses do not match the previous row/bank address, then the SDRAM controller automatically generates a precharge command, activates the new row and starts the read command. To comply with the SDRAM timing parameters, additional clock cycles on SDCK are inserted between precharge and active commands (t_{RP}) and between active and read command (t_{RCD}). These two parameters are set in the configuration register of the SDRAM Controller. After a read command, additional wait states are generated to comply with the CAS latency (1, 2 or 3 clock delays specified in the configuration register).

For a single access or an incremented burst of unspecified length, the SDRAM Controller anticipates the next access. While the last value of the column is returned by the SDRAM Controller on the bus, the SDRAM Controller anticipates the read to the next column and thus anticipates the CAS latency. This reduces the effect of the CAS latency on the internal bus.

For burst access of specified length (4, 8, 16 words), access is not anticipated. This case leads to the best performance. If the burst is broken (border, busy mode, etc.), the next access is handled as an incrementing burst of unspecified length.

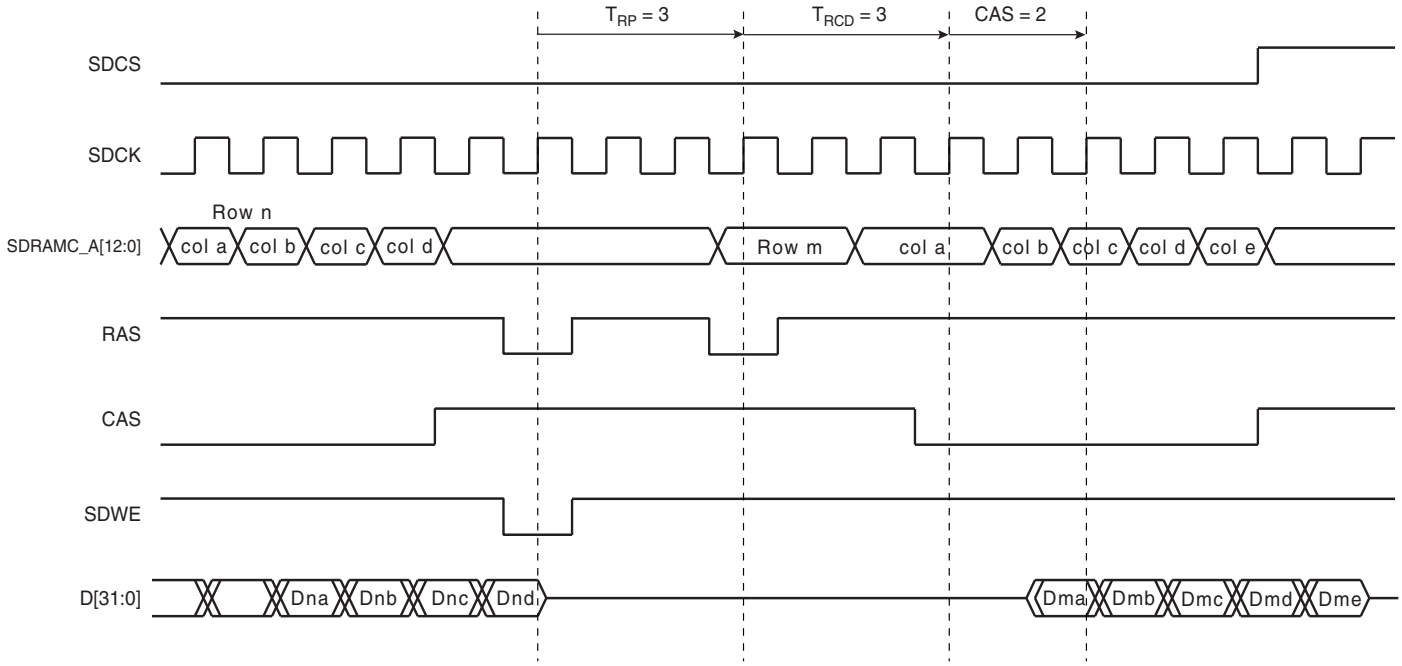
Figure 23-3. Read Burst, 32-bit SDRAM Access



23.5.3 Border Management

When the memory row boundary has been reached, an automatic page break is inserted. In this case, the SDRAM controller generates a precharge command, activates the new row and initiates a read or write command. To comply with SDRAM timing parameters, an additional clock cycle is inserted between the precharge/active (t_{RP}) command and the active/read (t_{RCD}) command. This is described in Figure 23-4 below.

Figure 23-4. Read Burst with Boundary Row Access



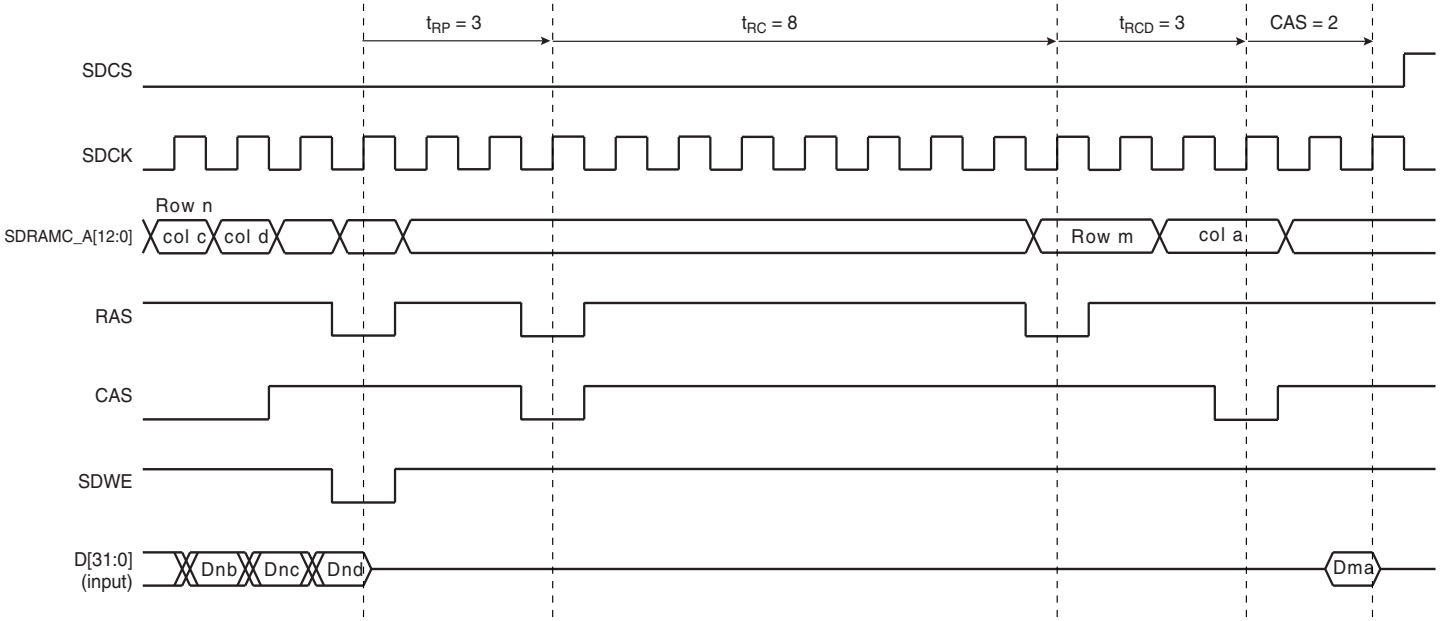
23.5.4 SDRAM Controller Refresh Cycles

An auto-refresh command is used to refresh the SDRAM device. Refresh addresses are generated internally by the SDRAM device and incremented after each auto-refresh automatically. The SDRAM Controller generates these auto-refresh commands periodically. An internal timer is loaded with the value in the register SDRAMC_TR that indicates the number of clock cycles between refresh cycles.

A refresh error interrupt is generated when the previous auto-refresh command did not perform. It is acknowledged by reading the Interrupt Status Register (SDRAMC_ISR).

When the SDRAM Controller initiates a refresh of the SDRAM device, internal memory accesses are not delayed. However, if the CPU tries to access the SDRAM, the slave indicates that the device is busy and the master is held by a wait signal. See Figure 23-5.

Figure 23-5. Refresh Cycle Followed by a Read Access



23.5.5 Power Management

Three low-power modes are available:

- Self-refresh Mode: The SDRAM executes its own Auto-refresh cycle without control of the SDRAM Controller. Current drained by the SDRAM is very low.
- Power-down Mode: Auto-refresh cycles are controlled by the SDRAM Controller. Between auto-refresh cycles, the SDRAM is in power-down. Current drained in Power-down mode is higher than in Self-refresh Mode.
- Deep Power-down Mode: (Only available with Mobile SDRAM) The SDRAM contents are lost, but the SDRAM does not drain any current.

The SDRAM Controller activates one low-power mode as soon as the SDRAM device is not selected. It is possible to delay the entry in self-refresh and power-down mode after the last access by programming a timeout value in the Low Power Register.

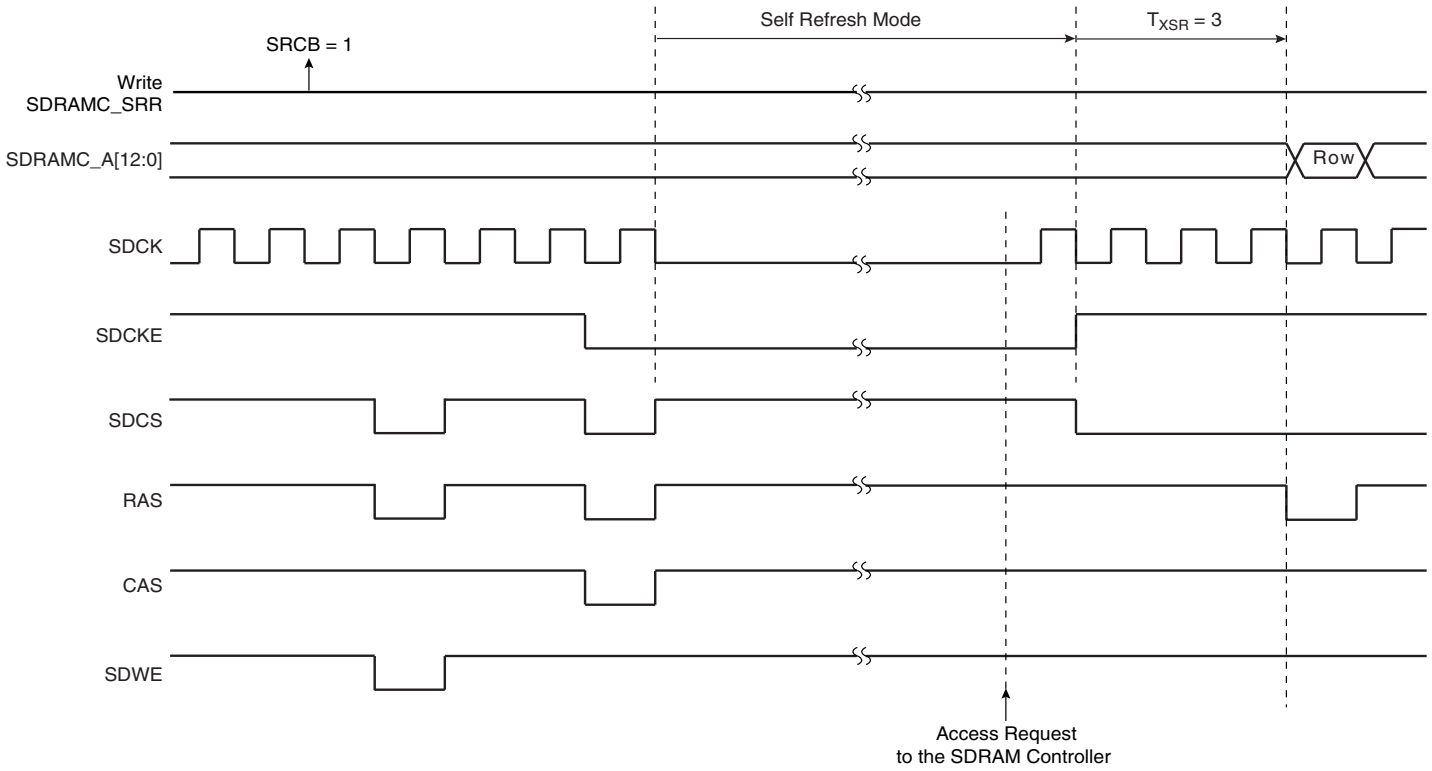
23.5.5.1 Self-refresh Mode

This mode is selected by programming the LPCB field to 1 in the SDRAMC Low Power Register. In self-refresh mode, the SDRAM device retains data without external clocking and provides its own internal clocking, thus performing its own auto-refresh cycles. All the inputs to the SDRAM device become “don’t care” except SDCKE, which remains low. As soon as the SDRAM device is selected, the SDRAM Controller provides a sequence of commands and exits self-refresh mode.

Some low-power SDRAMs (e.g., mobile SDRAM) can refresh only one quarter or a half quarter or all banks of the SDRAM array. This feature reduces the self-refresh current. To configure this feature, Temperature Compensated Self Refresh (TCSR), Partial Array Self Refresh (PASR) and Drive Strength (DS) parameters must be set in the Low Power Register and transmitted to the low-power SDRAM during initialization.

The SDRAM device must remain in self-refresh mode for a minimum period of t_{RAS} and may remain in self-refresh mode for an indefinite period. This is described in Figure 23-6.

Figure 23-6. Self-refresh Mode Behavior

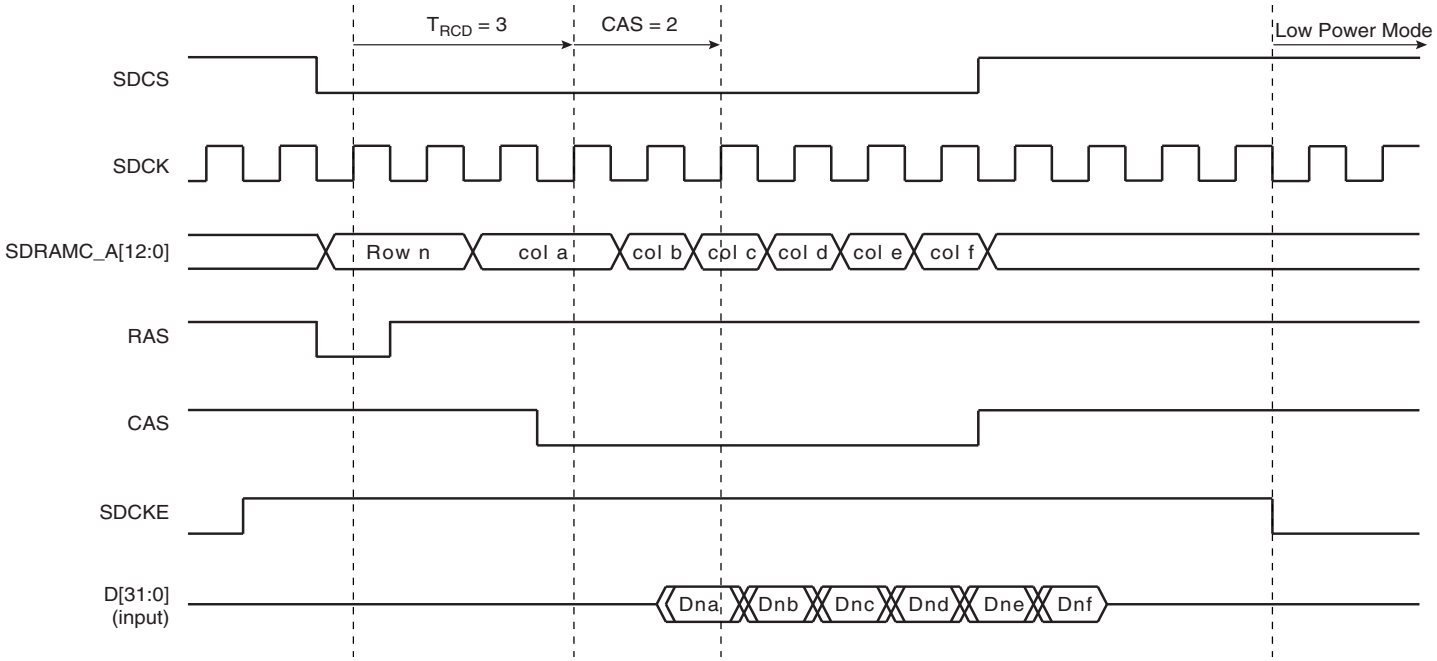


23.5.5.2 Low-power Mode

This mode is selected by programming the LPCB field to 2 in the SDRAMC Low Power Register. Power consumption is greater than in self-refresh mode. All the input and output buffers of the SDRAM device are deactivated except SDCKE, which remains low. In contrast to self-refresh mode, the SDRAM device cannot remain in low-power mode longer than the refresh period (64 ms for a whole device refresh operation). As no auto-refresh operations are performed by the SDRAM itself, the SDRAM Controller carries out the refresh operation. The exit procedure is faster than in self-refresh mode.

This is described in Figure 23-7.

Figure 23-7. Low-power Mode Behavior



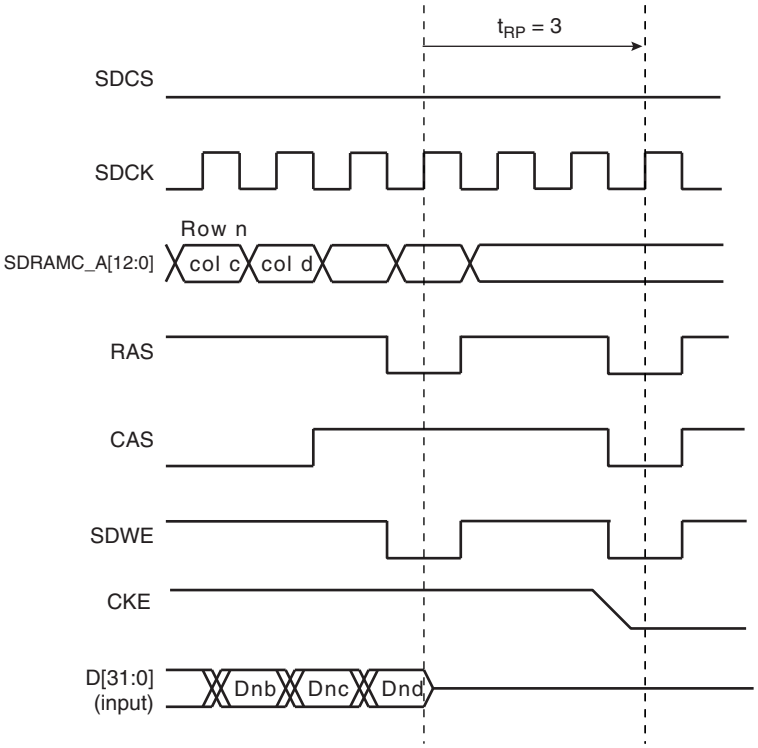
23.5.5.3 Deep Power-down Mode

This mode is selected by programming the LPCB field to 3 in the SDRAMC Low Power Register. When this mode is activated, all internal voltage generators inside the SDRAM are stopped and all data is lost.

When this mode is enabled, the application must not access to the SDRAM until a new initialization sequence is done (See “SDRAM Device Initialization” on page 235).

This is described in Figure 23-8.

Figure 23-8. Deep Power-down Mode Behavior



23.6 SDRAM Controller (SDRAMC) User Interface

Table 23-8. Register Mapping

Offset	Register	Name	Access	Reset
0x00	SDRAMC Mode Register	SDRAMC_MR	Read-write	0x00000000
0x04	SDRAMC Refresh Timer Register	SDRAMC_TR	Read-write	0x00000000
0x08	SDRAMC Configuration Register	SDRAMC_CR	Read-write	0x852372C0
0x10	SDRAMC Low Power Register	SDRAMC_LPR	Read-write	0x0
0x14	SDRAMC Interrupt Enable Register	SDRAMC_IER	Write-only	–
0x18	SDRAMC Interrupt Disable Register	SDRAMC_IDR	Write-only	–
0x1C	SDRAMC Interrupt Mask Register	SDRAMC_IMR	Read-only	0x0
0x20	SDRAMC Interrupt Status Register	SDRAMC_ISR	Read-only	0x0
0x24	SDRAMC Memory Device Register	SDRAMC_MDR	Read	0x0
0x28 - 0xFC	Reserved	–	–	–

23.6.1 SDRAMC Mode Register

Register Name:SDRAMC_MR

Addresses: 0xFFFFE200 (0), 0xFFFFE800 (1)

Access Type:Read-write

Reset Value: 0x00000000

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

- **MODE: SDRAMC Command Mode**

This field defines the command issued by the SDRAM Controller when the SDRAM device is accessed.

MODE			Description
0	0	0	Normal mode. Any access to the SDRAM is decoded normally. To activate this mode, command must be followed by a write to the SDRAM.
0	0	1	The SDRAM Controller 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.
0	1	0	The SDRAM Controller 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.
0	1	1	The SDRAM Controller 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.
1	0	0	The SDRAM Controller issues an “Auto-Refresh” Command when the SDRAM device is accessed regardless of the cycle. Previously, an “All Banks Precharge” command must be issued. To activate this mode, command must be followed by a write to the SDRAM.
1	0	1	The SDRAM Controller 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; most low-power SDRAM devices use the bank 1.
1	1	0	Deep power-down mode. Enters deep power-down mode.

23.6.2 SDRAMC Refresh Timer Register

Register Name:SDRAMC_TR

Addresses: 0xFFFFE204 (0), 0xFFFFE804 (1)

Access Type:Read-write

Reset Value: 0x00000000

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

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

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

7	6	5	4	3	2	1	0
COUNT							

- **COUNT: SDRAMC Refresh Timer Count**

This 12-bit field is loaded into a timer that generates the refresh pulse. Each time the refresh pulse is generated, a refresh burst is initiated. The value to be loaded depends on the SDRAMC clock frequency (MCK: Master Clock), the refresh rate of the SDRAM device and the refresh burst length where 15.6 µs per row is a typical value for a burst of length one.

To refresh the SDRAM device, this 12-bit field must be written. If this condition is not satisfied, no refresh command is issued and no refresh of the SDRAM device is carried out.

23.6.3 SDRAMC Configuration Register

Register Name:SDRAMC_CR

Addresses: 0xFFFFE208 (0), 0xFFFFE808 (1)

Access Type:Read-write

Reset Value: 0x852372C0

31	30	29	28	27	26	25	24
TXSR				TRAS			
23	22	21	20	19	18	17	16
TRCD				TRP			
15	14	13	12	11	10	9	8
TRC				TWR			
7	6	5	4	3	2	1	0
DBW	CAS		NB	NR		NC	

- **NC: Number of Column Bits**

Reset value is 8 column bits.

NC		Column Bits
0	0	8
0	1	9
1	0	10
1	1	11

- **NR: Number of Row Bits**

Reset value is 11 row bits.

NR		Row Bits
0	0	11
0	1	12
1	0	13
1	1	Reserved

- **NB: Number of Banks**

Reset value is two banks.

NB	Number of Banks
0	2
1	4

- **CAS: CAS Latency**

Reset value is two cycles.

In the SDRAMC, only a CAS latency of one, two and three cycles are managed.

CAS		CAS Latency (Cycles)
0	0	Reserved
0	1	1
1	0	2
1	1	3

- **DBW: Data Bus Width**

Reset value is 16 bits

0: Data bus width is 32 bits.

1: Data bus width is 16 bits.

- **TWR: Write Recovery Delay**

Reset value is two cycles.

This field defines the Write Recovery Time in number of cycles. Number of cycles is between 0 and 15.

- **TRC: Row Cycle Delay**

Reset value is seven cycles.

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

- **TRP: Row Precharge Delay**

Reset value is three cycles.

This field defines the delay between a Precharge Command and another Command in number of cycles. Number of cycles is between 0 and 15.

- **TRCD: Row to Column Delay**

Reset value is two cycles.

This field defines the delay between an Activate Command and a Read/Write Command in number of cycles. Number of cycles is between 0 and 15.

- **TRAS: Active to Precharge Delay**

Reset value is five cycles.

This field defines the delay between an Activate Command and a Precharge Command in number of cycles. Number of cycles is between 0 and 15.

- **TXSR: Exit Self Refresh to Active Delay**

Reset value is eight cycles.

This field defines the delay between SCKE set high and an Activate Command in number of cycles. Number of cycles is between 0 and 15.

23.6.4 SDRAMC Low Power Register

Register Name:SDRAMC_LPR

Addresses: 0xFFFFE210 (0), 0xFFFFE810 (1)

Access Type:Read-write

Reset Value: 0x0

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

- **LPCB: Low-power Configuration Bits**

00	Low Power Feature is inhibited: no Power-down, Self-refresh or Deep Power-down command is issued to the SDRAM device.
01	The SDRAM Controller issues a Self-refresh command to the SDRAM device, the SDCLK clock is deactivated and the SDCKE signal is set low. The SDRAM device leaves the Self Refresh Mode when accessed and enters it after the access.
10	The SDRAM Controller issues a Power-down Command to the SDRAM device after each access, the SDCKE signal is set to low. The SDRAM device leaves the Power-down Mode when accessed and enters it after the access.
11	The SDRAM Controller issues a Deep Power-down command to the SDRAM device. This mode is unique to low-power SDRAM.

- **PASR: Partial Array Self-refresh (only for low-power SDRAM)**

PASR parameter is transmitted to the SDRAM during initialization to specify whether only one quarter, one half or all banks of the SDRAM array are enabled. Disabled banks are not refreshed in self-refresh mode. This parameter must be set according to the SDRAM device specification.

- **TCSR: Temperature Compensated Self-Refresh (only for low-power SDRAM)**

TCSR parameter is transmitted to the SDRAM during initialization to set the refresh interval during self-refresh mode depending on the temperature of the low-power SDRAM. This parameter must be set according to the SDRAM device specification.

- **DS: Drive Strength (only for low-power SDRAM)**

DS parameter is transmitted to the SDRAM during initialization to select the SDRAM strength of data output. This parameter must be set according to the SDRAM device specification.

- **TIMEOUT: Time to define when low-power mode is enabled**

00	The SDRAM controller activates the SDRAM low-power mode immediately after the end of the last transfer.
01	The SDRAM controller activates the SDRAM low-power mode 64 clock cycles after the end of the last transfer.
10	The SDRAM controller activates the SDRAM low-power mode 128 clock cycles after the end of the last transfer.
11	Reserved.

23.6.5 SDRAMC Interrupt Enable Register

Register Name:SDRAMC_IER

Addresses: 0xFFFFE214 (0), 0xFFFFE814 (1)

Access Type:Write-only

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

- **RES: Refresh Error Status**

0: No effect.

1: Enables the refresh error interrupt.

23.6.6 SDRAMC Interrupt Disable Register

Register Name:SDRAMC_IDR

Addresses: 0xFFFFE218 (0), 0xFFFFE818 (1)

Access Type:Write-only

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

- **RES: Refresh Error Status**

0: No effect.

1: Disables the refresh error interrupt.

23.6.7 SDRAMC Interrupt Mask Register

Register Name:SDRAMC_IMR

Addresses: 0xFFFFE21C (0), 0xFFFFE81C (1)

Access Type:Read-only

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

- **RES: Refresh Error Status**

0: The refresh error interrupt is disabled.

1: The refresh error interrupt is enabled.

23.6.8 SDRAMC Interrupt Status Register

Register Name:SDRAMC_ISR

Addresses: 0xFFFFE220 (0), 0xFFFFE820 (1)

Access Type:Read-only

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

- **RES: Refresh Error Status**

0: No refresh error has been detected since the register was last read.

1: A refresh error has been detected since the register was last read.

23.6.9 SDRAMC Memory Device Register

Register Name:SDRAMC_MDR

Addresses: 0xFFFFE224 (0), 0xFFFFE824 (1)

Access Type:Read-write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	MD	

- **MD: Memory Device Type**

00	SDRAM
01	Low-power SDRAM
10	Reserved
11	Reserved.

24. Error Corrected Code (ECC) Controller

24.1 Overview

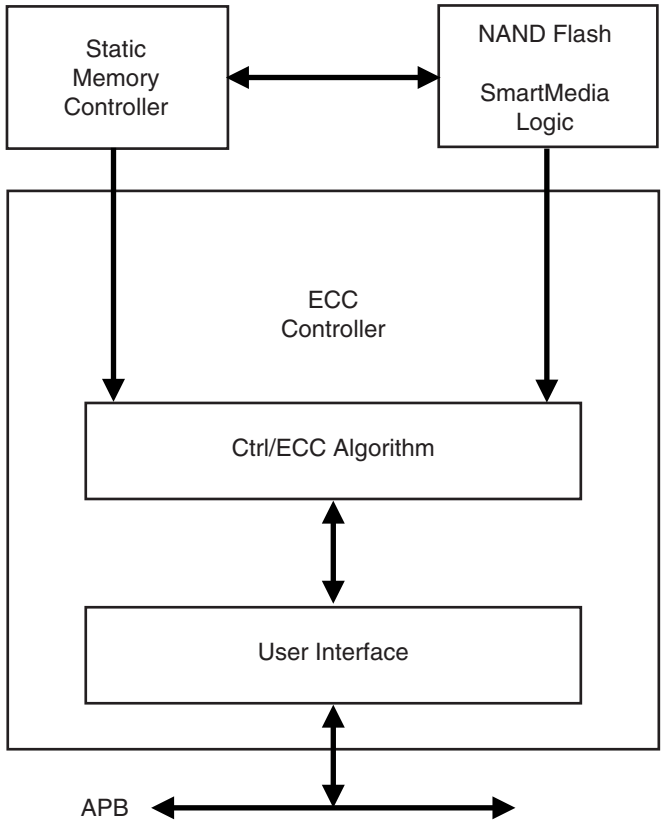
NAND Flash/SmartMedia devices contain by default invalid blocks which have one or more invalid bits. Over the NAND Flash/SmartMedia lifetime, additional invalid blocks may occur which can be detected/corrected by ECC code.

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).

24.2 Block Diagram

Figure 24-1. Block Diagram



24.3 Functional Description

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

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

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

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

24.3.1 Write Access

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

24.3.2 Read Access

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

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

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

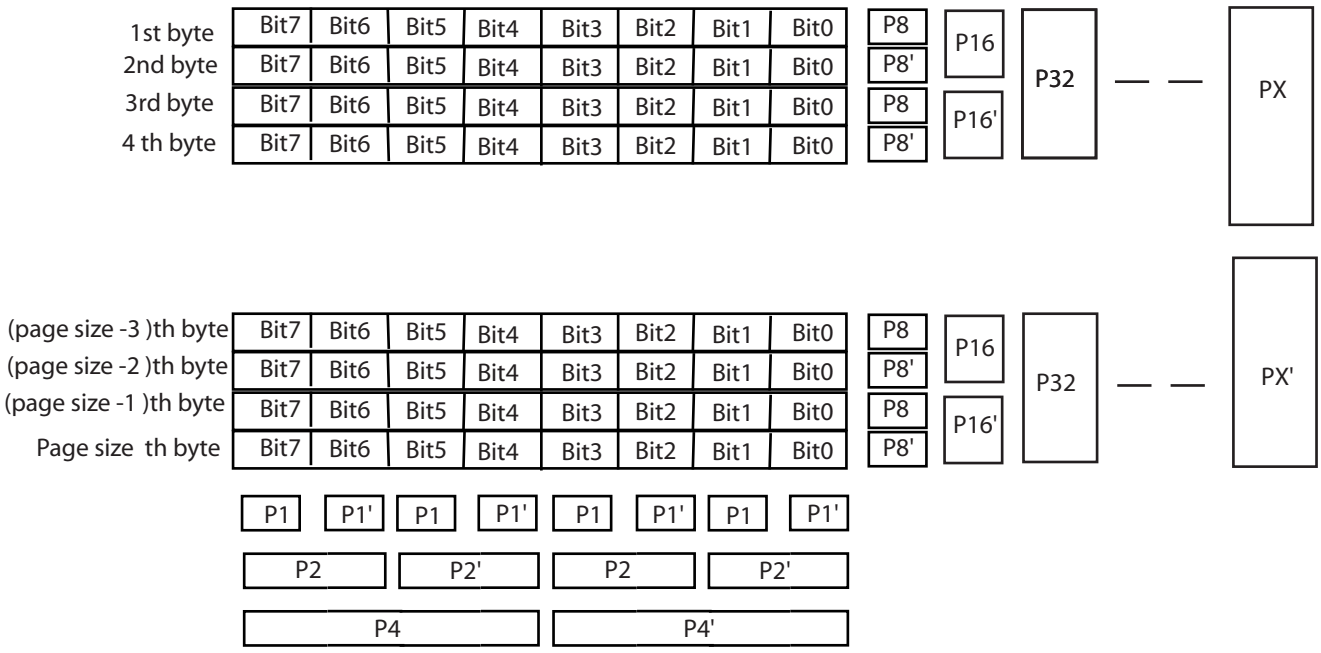
- No error: XOR between the ECC computation and the ECC code stored at the end of the NAND Flash or SmartMedia page is equal to 0. No error flags in the ECC Status Register (ECC_SR).
- Recoverable error: Only the RECERR flag in the ECC Status register (ECC_SR) is set. The corrupted word offset in the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

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

For Single-bit Error Correction and Double-bit Error Detection (SEC-DED) hsiao code is used. 32-bit ECC is generated in order to perform one bit correction per 512/1024/2048/4096 8- or 16-

bit words. Of the 32 ECC bits, 26 bits are for line parity and 6 bits are for column parity. They are generated according to the schemes shown in Figure 24-2 and Figure 24-3.

Figure 24-2. Parity Generation for 512/1024/2048/4096 8-bit Words1



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

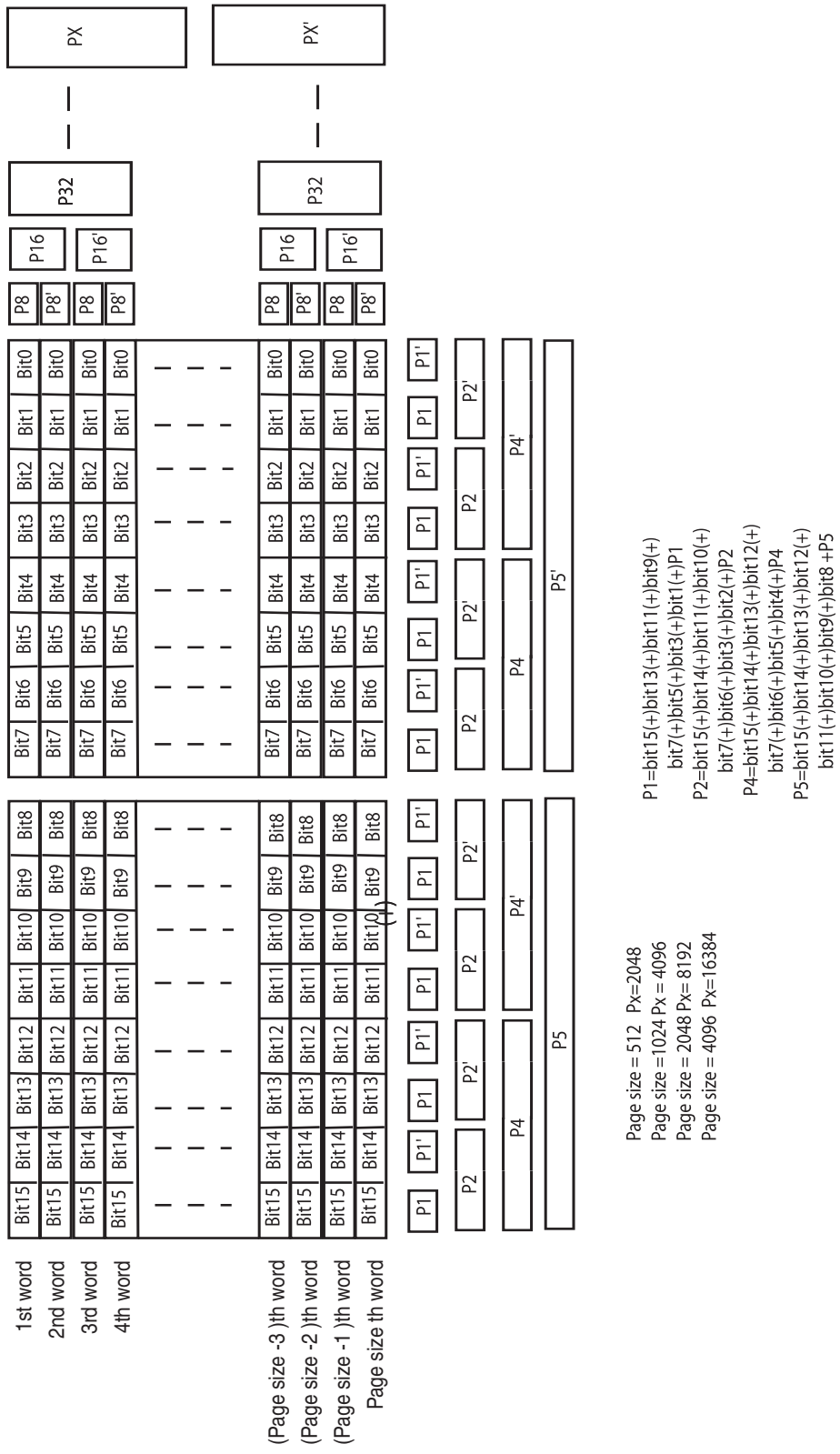
P1=bit7(+)bit5(+)bit3(+)bit1(+)P1
P2=bit7(+)bit6(+)bit3(+)bit2(+)P2
P4=bit7(+)bit6(+)bit5(+)bit4(+)P4
P1'=bit6(+)bit4(+)bit2(+)bit0(+)P1'
P2'=bit5(+)bit4(+)bit1(+)bit0(+)P2'
P4'=bit7(+)bit6(+)bit5(+)bit4(+)P4'

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

```
Page size = 2^n

  for i =0 to n
  begin
   for (j = 0 to page_size_byte)
   begin
    if(j[i] ==1)
     P[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 24-3. 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
```

24.4 Error Corrected Code Controller (ECC) User Interface

Table 24-1. Register Mapping

Offset	Register	Register Name	Access	Reset
0x00	ECC Control Register	ECC_CR	Write-only	0x0
0x04	ECC Mode Register	ECC_MR	Read-write	0x0
0x08	ECC Status Register	ECC_SR	Read-only	0x0
0x0C	ECC Parity Register	ECC_PR	Read-only	0x0
0x10	ECC NParity Register	ECC_NPR	Read-only	0x0
0x14 - 0xFC	Reserved	–	–	–

24.4.1 ECC Control Register

Name: ECC_CR

Addresses: 0xFFFFE000 (0), 0xFFFFE600 (1)

Access Type: Write-only

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

- **RST: RESET Parity**

Provides reset to current ECC by software.

1 = Resets ECC Parity and ECC NParity register.

0 = No effect.

24.4.2 ECC Mode Register

Register Name:ECC_MR

Addresses: 0xFFFFE004 (0), 0xFFFFE604 (1)

Access Type:Read-write

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

- **PAGESIZE: Page Size**

This field defines the page size of the NAND Flash device.

Page Size	Description
00	528 words
01	1056 words
10	2112 words
11	4224 words

A word has a value of 8 bits or 16 bits, depending on the NAND Flash or SmartMedia memory organization.

24.4.3 ECC Status Register

Register Name:ECC_SR

Addresses: 0xFFFFE008 (0), 0xFFFFE608 (1)

Access Type:Read-only

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

- **RECERR: Recoverable Error**

0 = No Errors Detected.

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

- **ECCERR: ECC Error**

0 = No Errors Detected.

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

Read both ECC Parity and ECC NParity register, the error occurred at the location which contains a 1 in the least significant 16 bits.

- **MULERR: Multiple Error**

0 = No Multiple Errors Detected.

1 = Multiple Errors Detected.

24.4.4 ECC Parity Register

Register Name:ECC_PR

Addresses: 0xFFFFE00C (0), 0xFFFFE60C (1)

Access Type:Read-only

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

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

- **BITADDR**

During a page read, this value contains the corrupted bit offset where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

- **WORDADDR**

During a page read, this value contains the word address (8-bit or 16-bit word depending on the memory plane organization) where an error occurred, if a single error was detected. If multiple errors were detected, this value is meaningless.

24.4.5 ECC NParity Register

Register Name:ECC_NPR

Addresses: 0xFFFFE010 (0), 0xFFFFE610 (1)

Access Type:Read-only

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

- **NPARITY:**

Once the entire main area of a page is written with data, the register content must be stored at any free location of the spare area.

25. DMA Controller (DMAC)

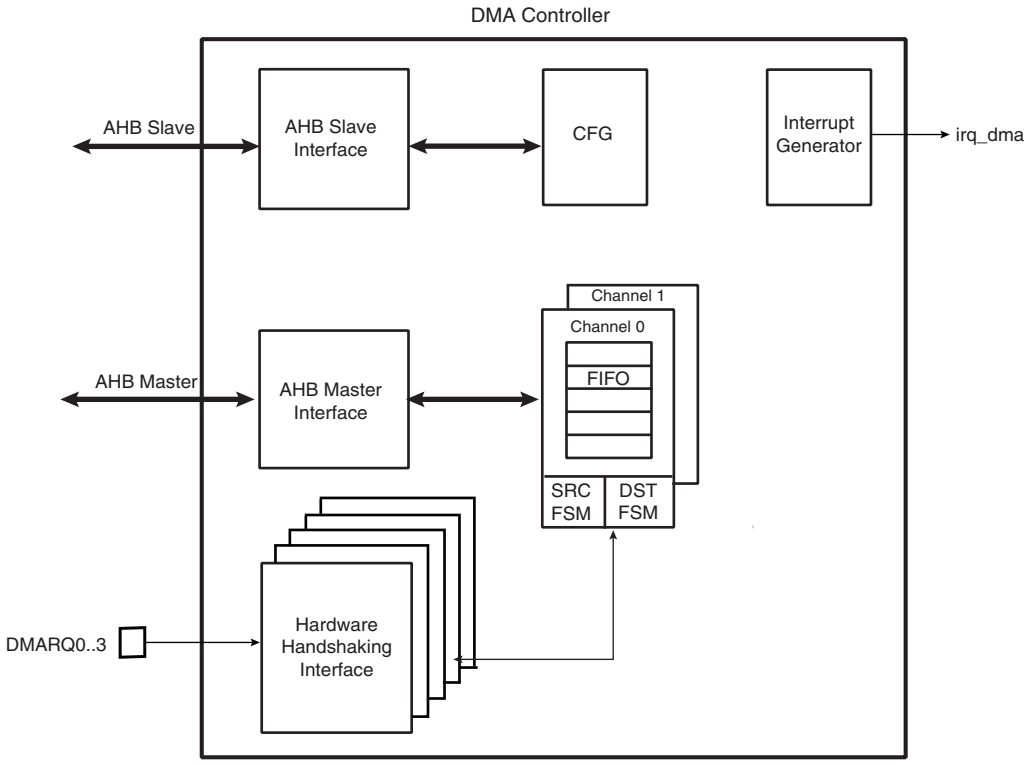
25.1 Overview

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 DMA data transfer. This is also known as a dual-access transfer.

The DMAC is programmed via the AHB slave interface.

25.2 Block Diagram

Figure 25-1. DMA Controller (DMAC) Block Diagram



25.3 Functional Description

25.3.1 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 DMA transfer and does not require a handshaking interface to interact with the DMAC. A peripheral should be assigned as memory

only if it does not insert more than 16 wait states. If more than 16 wait states are required, then the peripheral should use a handshaking interface (the default if the peripheral is not programmed to be memory) in order to signal when it is ready to accept or supply data.

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 AHB 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 burst transaction between them. This interface is used to request, acknowledge, and control a DMAC transaction. A channel can receive a request through one of three types of handshaking interface: hardware, software, or peripheral interrupt.

Hardware handshaking interface: Uses hardware signals to control the transfer of a single or burst transaction between the DMAC and the source or destination peripheral.

Software handshaking interface: Uses software registers to control the transfer of a single or burst transaction 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.

Peripheral interrupt handshaking interface: A simple use of the hardware handshaking interface. In this mode, the interrupt line from the peripheral is tied to the dma_req input of the hardware handshaking interface. Other interface signals are ignored.

Flow controller: The device (either the DMAC or source/destination peripheral) that determines the length of and terminates a DMA block transfer. If the length of a block is known before enabling the channel, then the DMAC should be programmed as the flow controller. If the length of a block is not known prior to enabling the channel, the source or destination peripheral needs to terminate a block transfer. In this mode, the peripheral is the flow controller.

Flow control mode (DMAC_CFGx.FCMODE): Special mode that only applies when the destination peripheral is the flow controller. It controls the pre-fetching of data from the source peripheral.

Transfer hierarchy: Figure 25-2 on page 271 illustrates the hierarchy between DMAC transfers, block transfers, transactions (single or burst), and AMBA transfers (single or burst) for non-memory peripherals. Figure 25-3 on page 271 shows the transfer hierarchy for memory.

Figure 25-2. DMAC Transfer Hierarchy for Non-Memory Peripheral

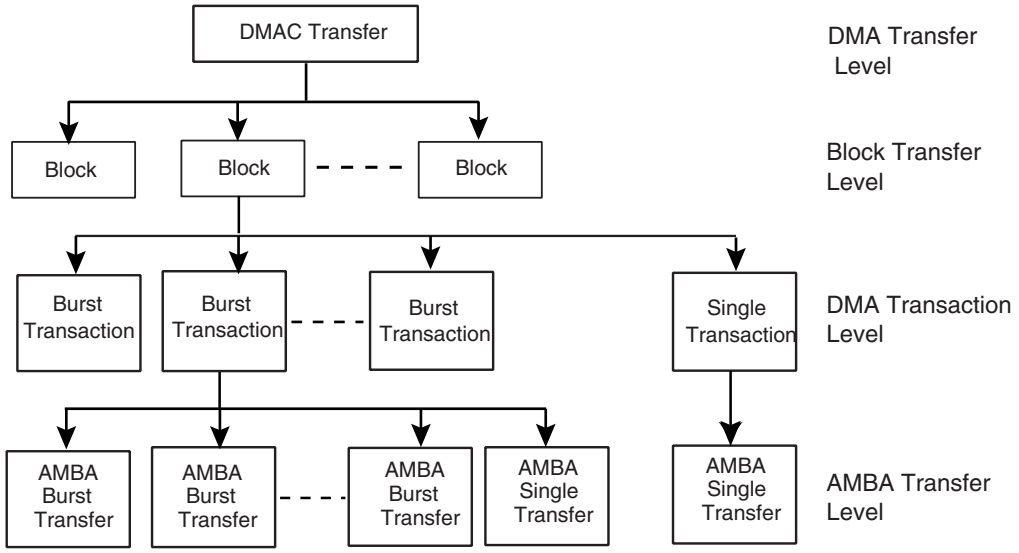
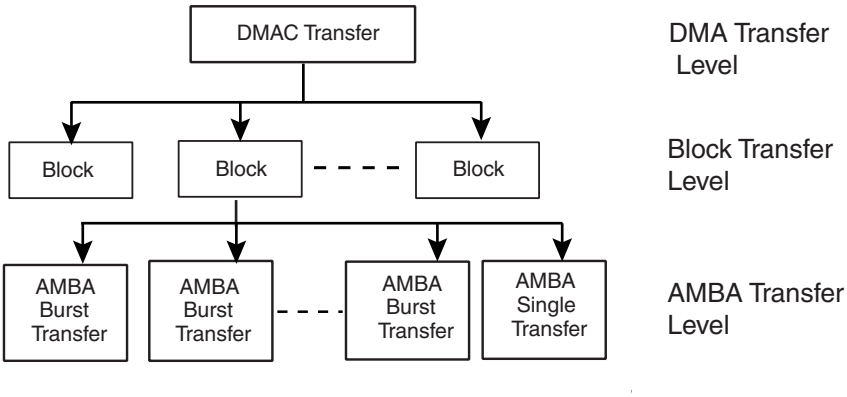


Figure 25-3. DMAC Transfer Hierarchy for Memory



Block: A block of DMAC data. The amount of data (block length) is determined by the flow controller. For transfers between the DMAC and memory, a block is broken directly into a sequence of AMBA bursts and AMBA single transfers. For transfers between the DMAC and a non-memory peripheral, a block is broken into a sequence of DMAC transactions (single and bursts). 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 and burst.

- **Single transaction:** The length of a single transaction is always 1 and is converted to a single AMBA transfer.
- **Burst transaction:** The length of a burst transaction is programmed into the DMAC. The burst transaction is converted into a sequence of AMBA bursts and AMBA single transfers. DMAC executes each AMBA burst transfer by performing incremental bursts that are no longer than the maximum AMBA burst size set. The

burst transaction length is under program control and normally bears some relationship to the FIFO sizes in the DMAC and in the source and destination peripherals.

DMA transfer: Software controls the number of blocks in a DMAC transfer. Once the DMA transfer has completed, then hardware within the DMAC disables the channel and can generate an interrupt to signal the completion of the DMA transfer. You can then re-program the channel for a new DMA transfer.

Single-block DMA transfer: Consists of a single block.

Multi-block DMA transfer: A DMA transfer may consist of multiple DMAC blocks. Multi-block DMA transfers are supported through block chaining (linked list pointers), auto-reloading of channel registers, and contiguous blocks. The source and destination can independently select which method to use.

- **Linked lists (block chaining) –** A linked list pointer (LLP) 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 block (block descriptor) and an LLP register. The DMAC fetches the LLI at the beginning of every block when block chaining is enabled.
- **Auto-reloading –** The DMAC automatically reloads the channel registers at the end of each block to the value when the channel was first enabled.
- **Contiguous blocks –** Where the address between successive blocks is selected to be a continuation from the end of the previous block.

Scatter: Relevant to destination transfers within a block. The destination AMBA address is incremented/decremented by a programmed amount when a scatter boundary is reached. The number of AMBA transfers between successive scatter boundaries is under software control.

Gather: Relevant to source transfers within a block. The source AMBA address is incremented/decremented by a programmed amount when a gather boundary is reached. The number of AMBA transfers between successive gather boundaries is under software control.

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 DMA transfer, block, or transaction (single or burst).

Bus locking: Software can program a channel to maintain control of the AMBA bus by asserting hlock for the duration of a DMA transfer, block, or transaction (single or burst). Channel locking is asserted for the duration of bus locking at a minimum.

FIFO mode: Special mode to improve bandwidth. When enabled, the channel waits until the FIFO is less than half full to fetch the data from the source peripheral and waits until the FIFO is greater than or equal to half full to send data to the destination peripheral. Thus, the channel can transfer the data using AMBA bursts, eliminating the need to arbitrate for the AHB master interface for each single AMBA transfer. When this mode is not enabled, the channel only waits until the FIFO can transmit/accept a single AMBA transfer before requesting the master bus interface.

Pseudo fly-by operation: Typically, it takes two AMBA bus cycles to complete a transfer, one for reading the source and one for writing to the destination. However, when the source and destination peripherals of a DMA transfer are on different AMBA layers, it is possible for the DMAC to fetch data from the source and store it in the channel FIFO at the same time as the DMAC extracts data from the channel FIFO and writes it to the destination peripheral. This activity is

known as *pseudo fly-by operation*. For this to occur, the master interface for both source and destination layers must win arbitration of their AHB layer. Similarly, the source and destination peripherals must win ownership of their respective master interfaces.

25.3.2 Memory Peripherals

Figure 25-3 on page 271 shows the DMA 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.

25.3.3 Handshaking Interface

Handshaking interfaces are used at the transaction level to control the flow of single or burst transactions. 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 DMA 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.

25.3.3.1 Software Handshaking

When the slave peripheral requires the DMAC to perform a DMA 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 DMA transaction. These software registers are used to implement the software handshaking interface.

The HS_SEL_SRC/HS_SEL_DST bit in the DMAC_CFGx channel configuration register must be set to enable software handshaking.

When the peripheral is not the flow controller, then the last transaction registers DMAC_LstSrcReg and DMAC_LstDstReg are not used, and the values in these registers are ignored.

25.3.3.2 Burst Transactions

Writing a 1 to the DMAC_ReqSrcReg[x]/DMAC_ReqDstReg[x] register is always interpreted as a burst transaction request, where *x* is the channel number. However, in order for a burst transaction request to start, software must write a 1 to the DMAC_SglReqSrcReg[x]/DMAC_SglReqDstReg[x] register.

You can write a 1 to the DMAC_SglReqSrcReg[*x*]/DMAC_SglReqDstReg[*x*] and DMAC_ReqSrcReg[*x*]/DMAC_ReqDstReg[*x*] registers in any order, but both registers must be asserted in order to initiate a burst transaction. Upon completion of the burst transaction, the hardware clears the DMAC_SglReqSrcReg[*x*]/DMAC_SglReqDstReg[*x*] and DMAC_ReqSrcReg[*x*]/DMAC_ReqDstReg[*x*] registers.

25.3.3.3 *Single Transactions*

Writing a 1 to the DMAC_SglReqSrcReg/DMAC_SglReqDstReg initiates a single transaction. Upon completion of the single transaction, both the DMAC_SglReqSrcReg/DMAC_SglReqDstReg and DMAC_ReqSrcReg/DMAC_ReqDstReg bits are cleared by hardware. Therefore, writing a 1 to the DMAC_ReqSrcReg/DMAC_ReqDstReg is ignored while a single transaction has been initiated, and the requested burst transaction is not serviced.

Again, writing a 1 to the DMAC_ReqSrcReg/DMAC_ReqDstReg register is always a burst transaction request. However, in order for a burst transaction request to start, the corresponding channel bit in the DMAC_SglReqSrcReg/DMAC_SglReqDstReg must be asserted. Therefore, to ensure that a burst transaction is serviced, you must write a 1 to the DMAC_ReqSrcReg/DMAC_ReqDstReg before writing a 1 to the DMAC_SglReqSrcReg/DMAC_SglReqDstReg register.

Software can poll the relevant channel bit in the DMAC_SglReqSrcReg/ DMAC_SglReqDstReg and DMAC_ReqSrcReg/DMAC_ReqDstReg registers. When both are 0, then either the requested burst or single transaction has completed. Alternatively, the IntSrcTran or IntDstTran interrupts can be enabled and unmasked in order to generate an interrupt when the requested source or destination transaction has completed.

Note: The transaction-complete interrupts are triggered when both single and burst transactions are complete. The same transaction-complete interrupt is used for both single and burst transactions.

25.3.3.4 *Hardware Handshaking*

There are 5 hardware handshaking interfaces connected to four external DMA requests (see Table 25-1).

Table 25-1. Hardware Handshaking Connection

Request	Definition	Hardware Handshaking Interface
DMAREQ0	External DMA Request 0	1
DMAREQ1	External DMA Request 1	2
DMAREQ2	External DMA Request 2	3
DMAREQ3	External DMA Request 3	4

25.3.3.5 *External DMA Request Definition*

When an external slave peripheral requires the DMAC to perform DMA transactions, it communicates its request by asserting the external nDMAREQx signal. This signal is resynchronized to ensure a proper functionality (see Figure 25-4 on page 275).

The external nDMAREQx is asserted when the source threshold level is reached. After resynchronization, the rising edge of dma_req starts the transfer. dma_req is de-asserted when dma_ack is asserted.

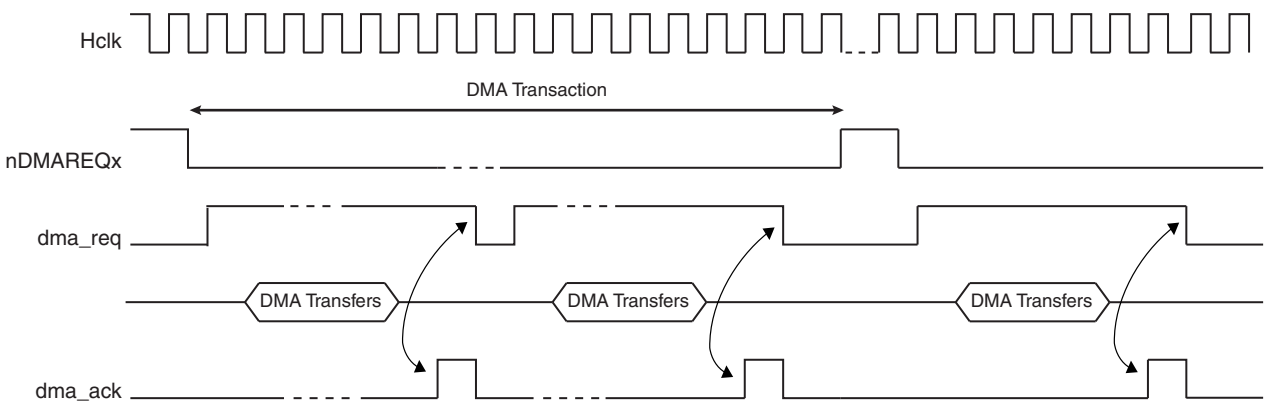
Each DMAREQx assertion leads to a transfer. Its size (given by CTLxL.SRC_MSIZE and CTLxL.DEST_MSIZE) is decremented from CTLxH.BLOCK_TS.

The external nDMAREQx signal must be de-asserted after the last transfer and re-asserted again before a new transaction starts. The DMA ends the current transfer.

For a source FIFO, an active edge is triggered on nDMAREQx when the source FIFO exceeds a watermark level. For a destination FIFO, an active edge is triggered on nDMAREQx when the destination FIFO drops below the watermark level.

The source transaction length, CTLxL.SRC_MSIZE, and destination transaction length, CTLxL.DEST_MSIZE, must be set according to watermark levels on the source/destination peripherals.

Figure 25-4. External DMA Request Timing



25.3.4 DMAC Transfer Types

A DMA transfer may consist of single or multi-block transfers. On successive blocks of a multi-block transfer, the DMAC_SARx/DMAC_DARx register in the DMAC is reprogrammed using either of the following methods:

- Block chaining using linked lists
- Auto-reloading
- Contiguous address between blocks

On successive blocks of a multi-block transfer, the DMAC_CTLx register in the DMAC is re-programmed using either of the following methods:

- Block chaining using linked lists
- Auto-reloading

When block chaining, using linked lists is the multi-block method of choice, and on successive blocks, the DMAC_LLPx register in the DMAC is re-programmed using the following method:

- Block chaining using linked lists

A block descriptor (LLI) consists of following registers, DMAC_SARx, DMAC_DARx, DMAC_LLPx, DMAC_CTLx. These registers, along with the DMAC_CFGx register, are used by the DMAC to set up and describe the block transfer.

25.3.4.1 Multi-block Transfers

25.3.4.2 Block Chaining Using Linked Lists

In this case, the DMAC re-programs the channel registers prior to the start of each block by fetching the block descriptor for that block from system memory. This is known as an LLI update.

DMAC block chaining is supported by using a Linked List Pointer register (DMAC_LLPx) that stores the address in memory of the next linked list item. Each LLI (block descriptor) contains the corresponding block descriptor (DMAC_SARx, DMAC_DARx, DMAC_LLPx, DMAC_CTLx).

To set up block chaining, a sequence of linked lists must be programmed in memory.

The DMAC_SARx, DMAC_DARx, DMAC_LLPx and DMAC_CTLx registers are fetched from system memory on an LLI update. Figure 25-5 shows how to use chained linked lists in memory to define multi-block transfers using block chaining.

The Linked List multi-block transfers is initiated by programming DMAC_LLPx with LLPx(0) (LLI(0) base address) and DMAC_CTLx with DMAC_CTLx.LLP_S_EN and DMAC_CTLx.LLP_D_EN.

Figure 25-5. Multi-block Transfer Using Linked Lists

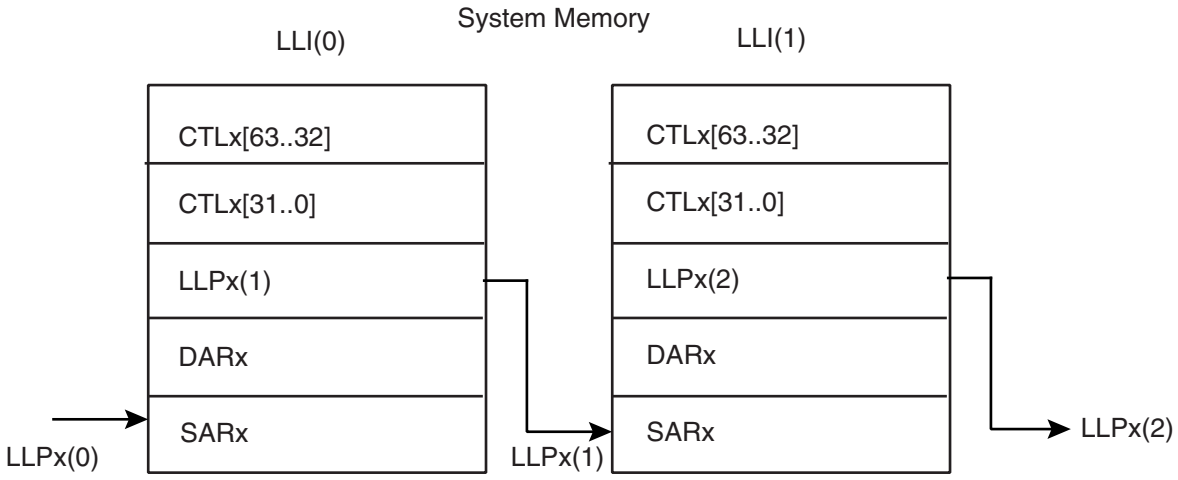


Table 25-2. Programming of Transfer Types and Channel Register Update Method (DMAC State Machine Table)

Transfer Type	LLP.LOC = 0	LLP_S_EN (DMAC_CTLx)	RELOAD_SR (DMAC_CFGx)	LLP_D_EN (DMAC_CTLx)	RELOAD_DS (DMAC_CFGx)	DMAC_CTLx, DMAC_LLPx Update Method	DMAC_SARx Update Method	DMAC_DARx Update Method	–
1) Single Block or last transfer of multi-Block	Yes	0	0	0	0	None, user reprograms	None (single)	None (single)	–
2) AutoReload multi-block transfer with contiguous SAR	Yes	0	0	0	1	DMAC_CTLx,DMAC_LLPx are reloaded from initial values.	Contiguous	Auto-Reload	–
3) AutoReload multi-block transfer with contiguous DAR	Yes	0	1	0	0	DMAC_CTLx,DMAC_LLPx are reloaded from initial values.	Auto-Reload	Con-tiguous	–
4) AutoReload multi-block transfer	Yes	0	1	0	1	DMAC_CTLx,DMAC_LLPx are reloaded from initial values.	Auto-Reload	Auto-Reload	–
6) Linked List multi-block transfer with contiguous SAR	No	0	0	1	0	DMAC_CTLx,DMAC_LLPx loaded from next Linked List item	Contiguous	Linked List	–
7) Linked List multi-block transfer with auto-reload SAR	No	0	1	1	0	DMAC_CTLx,DMAC_LLPx loaded from next Linked List item	Auto-Reload	Linked List	–
8) Linked List multi-block transfer with contiguous DAR	No	1	0	0	0	DMAC_CTLx,DMAC_LLPx loaded from next Linked List item	Linked List	Con-tiguous	–
9) Linked List multi-block transfer with auto-reload DAR	No	1	0	0	1	DMAC_CTLx,DMAC_LLPx loaded from next Linked List item	Linked List	Auto-Reload	–
10) Linked List multi-block transfer	No	1	0	1	0	DMAC_CTLx,DMAC_LLPx loaded from next Linked List item	Linked List	Linked List	–

25.3.4.3 Auto-reloading of Channel Registers

During auto-reloading, the channel registers are reloaded with their initial values at the completion of each block and the new values used for the new block. Depending on the row number in Table 25-2 on page 277, some or all of the DMAC_SARx, DMAC_DARx and DMAC_CTLx channel registers are reloaded from their initial value at the start of a block transfer.

25.3.4.4 Contiguous Address Between Blocks

In this case, the address between successive blocks is selected to be a continuation from the end of the previous block. Enabling the source or destination address to be contiguous between blocks is a function of DMAC_CTLx.LLP_S_EN, DMAC_CFGx.RELOAD_SR, DMAC_CTLx.LLP_D_EN, and DMAC_CFGx.RELOAD_DS registers (see Figure 25-2 on page 277).

Note: Both DMAC_SARx and DMAC_DARx updates cannot be selected to be contiguous. If this functionality is required, the size of the Block Transfer (DMAC_CTLx.BLOCK_TS) must be increased. If this is at the maximum value, use Row 10 of Table 25-2 on page 277 and setup the LLI.DMAC_SARx address of the block descriptor to be equal to the end DMAC_SARx address of the previous block. Similarly, setup the LLI.DMAC_DARx address of the block descriptor to be equal to the end DMAC_DARx address of the previous block.

25.3.4.5 Suspension of Transfers Between Blocks

At the end of every block transfer, an end of block interrupt is asserted if:

- interrupts are enabled, DMAC_CTLx.INT_EN = 1
- the channel block interrupt is unmasked, DMAC_MaskBlock[n] = 0, where n is the channel number.

Note: The block complete interrupt is generated at the completion of the block transfer to the destination.

For rows 6, 8, and 10 of Table 25-2 on page 277, the DMA transfer does not stall between block transfers. For example, at the end of block N, the DMAC automatically proceeds to block N + 1.

For rows 2, 3, 4, 7, and 9 of Table 25-2 on page 277 (DMAC_SARx and/or DMAC_DARx auto-reloaded between block transfers), the DMA transfer automatically stalls after the end of block. Interrupt is asserted if the end of block interrupt is enabled and unmasked.

The DMAC does not proceed to the next block transfer until a write to the block interrupt clear register, DMAC_ClearBlock[n], is performed by software. This clears the channel block complete interrupt.

For rows 2, 3, 4, 7, and 9 of Table 25-2 on page 277 (DMAC_SARx and/or DMAC_DARx auto-reloaded between block transfers), the DMA transfer does not stall if either:

- interrupts are disabled, DMAC_CTLx.INT_EN = 0, or
- the channel block interrupt is masked, DMAC_MaskBlock[n] = 1, where n is the channel number.

Channel suspension between blocks is used to ensure that the end of block ISR (interrupt service routine) of the next-to-last block is serviced before the start of the final block commences. This ensures that the ISR has cleared the DMAC_CFGx.RELOAD_SR and/or DMAC_CFGx.RELOAD_DS bits before completion of the final block. The reload bits DMAC_CFGx.RELOAD_SR and/or DMAC_CFGx.RELOAD_DS should be cleared in the ‘end of block ISR’ for the next-to-last block transfer.

25.3.4.6 Ending Multi-block Transfers

All multi-block transfers must end as shown in Row 1 of Table 25-2 on page 277. At the end of every block transfer, the DMAC samples the row number, and if the DMAC is in Row 1 state, then the previous block transferred was the last block and the DMA transfer is terminated.

For rows 2,3 and 4 of Table 25-2 on page 277, (DMAC_LLPx = 0 and DMAC_CFGx.RELOAD_SR and/or DMAC_CFGx.RELOAD_DS is set), multi-block DMA transfers continue until both the DMAC_CFGx.RELOAD_SR and DMAC_CFGx.RELOAD_DS registers are cleared by software. They should be programmed to zero in the end of block interrupt service routine that services the next-to-last block transfer. This puts the DMAC into Row 1 state.

Note: For rows 6, 8, and 10 (both DMAC_CFGx.RELOAD_SR and DMAC_CFGx.RELOAD_DS cleared) the user must setup the last block descriptor in memory such that both LLI.DMAC_CTLx.LLP_S_EN and LLI.DMAC_CTLx.LLP_D_EN are zero.For rows 7 and 9, the end-of-block interrupt service routine that services the next-to-last block transfer should clear the DMAC_CFGx.RELOAD_SR and DMAC_CFGx.RELOAD_DS reload bits. The last block descriptor in memory should be set up so that both the LLI.DMAC_CTLx.LLP_S_EN and LLI.DMAC_CTLx.LLP_D_EN are zero.

25.3.5 Programming a Channel

Three registers, the DMAC_LLPx, the DMAC_CTLx and DMAC_CFGx, need to be programmed to set up whether single or multi-block transfers take place, and which type of multi-block transfer is used. The different transfer types are shown in Table 25-2 on page 277.

The “Update Method” column indicates where the values of DMAC_SARx, DMAC_DARx, DMAC_CTLx, and DMAC_LLPx are obtained for the next block transfer when multi-block DMAC transfers are enabled.

Note: In Table 25-2 on page 277, all other combinations of DMAC_LLPx.LOC = 0, DMAC_CTLx.LLP_S_EN, DMAC_CFGx.RELOAD_SR, DMAC_CTLx.LLP_D_EN, and DMAC_CFGx.RELOAD_DS are illegal, and causes indeterminate or erroneous behavior.

25.3.5.1 Programming Examples

25.3.5.2 Single-block Transfer (Row 1)

1. Read the Channel Enable register to choose a free (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: DMAC_ClearTfr, DMAC_ClearBlock, DMAC_ClearSrcTran, DMAC_ClearDstTran, DMAC_ClearErr. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
3. Program the following channel registers:
   a. Write the starting source address in the DMAC_SARx register for channel x.
   b. Write the starting destination address in the DMAC_DARx register for channel x.
   c. Program DMAC_CTLx and DMAC_CFGx according to Row 1 as shown in Table 25-2 on page 277. Program the DMAC_LLPx register with ‘0’.
   d. Write the control information for the DMA transfer in the DMAC_CTLx 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 TT_FC of the DMAC_CTLx register.

– ii. Set up the transfer characteristics, such as:
  – Transfer width for the source in the SRC_TR_WIDTH field.
  – Transfer width for the destination in the DST_TR_WIDTH field.
  – Source master layer in the SMS field where source resides.
  – Destination master layer in the DMS field where destination resides.
  – Incrementing/decrementing or fixed address for source in SINC field.
  – Incrementing/decrementing or fixed address for destination in DINC 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 HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a '0' activates the hardware handshaking interface to handle source/destination requests. Writing a '1' 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 DEST_PER bits, respectively.

f. If gather is enabled (DMAC_CTLx.S_GATH_EN is enabled), program the DMAC_SGRx register for channel x.

g. If scatter is enabled (DMAC_CTLx.D_SCAT_EN, program the DMAC_DSRx register for channel x.

4. After the DMAC selected channel has been programmed, enable the channel by writing a '1' to the DMAC_ChEnReg.CH_EN bit. Make sure that bit 0 of the DMAC_DmaCfgReg register is enabled.
5. Source and destination request single and burst DMA transactions to transfer the block of data (assuming non-memory peripherals). The DMAC acknowledges at the completion of every transaction (burst and single) in the block and carry out the block transfer.
6. Once the transfer completes, hardware sets the interrupts and disables the channel. At this time you can either respond to the Block Complete or Transfer Complete interrupts, or poll for the Channel Enable (DMAC_ChEnReg.CH_EN) bit until it is cleared by hardware, to detect when the transfer is complete.

25.3.5.3 Multi-block Transfer with Linked List for Source and Linked List for Destination (Row 10)

1. Read the Channel Enable register to choose a free (disabled) channel.
2. Set up the chain of Linked List Items (otherwise known as block descriptors) in memory. Write the control information in the LLI.DMAC_CTLx register location of the block descriptor for each LLI in memory (see Figure 25-8 on page 284) 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 TT_FC of the DMAC_CTLx register.

   b. Set up the transfer characteristics, such as:

   – i. Transfer width for the source in the SRC_TR_WIDTH field.

   – ii. Transfer width for the destination in the DST_TR_WIDTH field.

   – iii. Source master layer in the SMS field where source resides.

   – iv. Destination master layer in the DMS field where destination resides.

– v. Incrementing/decrementing or fixed address for source in SINC field.

– vi. Incrementing/decrementing or fixed address for destination DINC field.

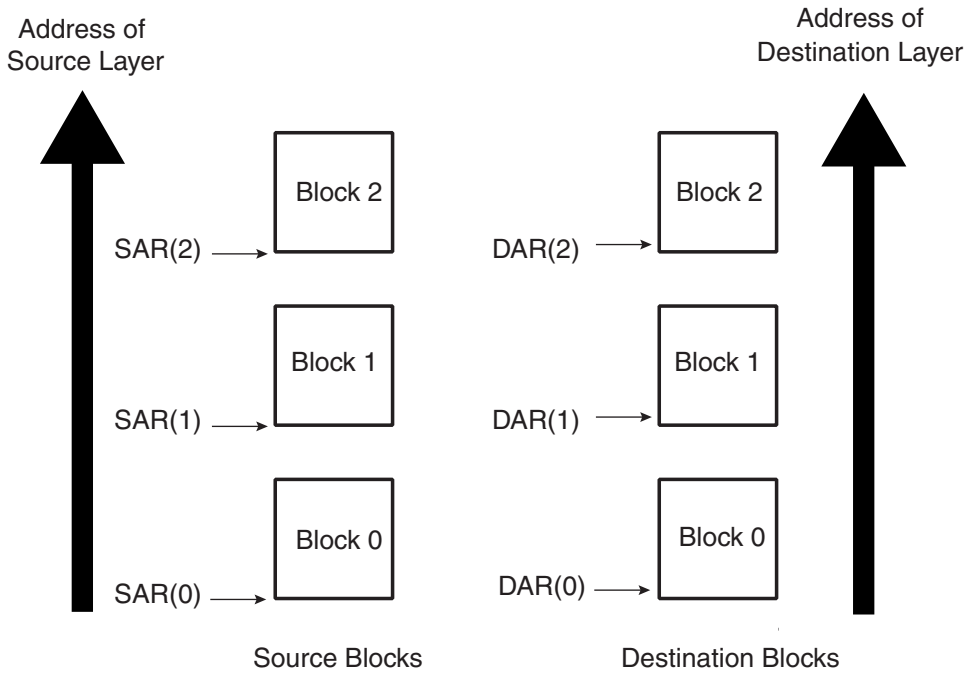
3. Write the channel configuration information into the DMAC_CFGx register for channel x.
   a. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a '0' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '1' 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 DEST_PER bits, respectively.
4. Make sure that the LLI.DMAC_CTLx register locations of all LLI entries in memory (except the last) are set as shown in Row 10 of Table 25-2 on page 277. The LLI.DMAC_CTLx register of the last Linked List Item must be set as described in Row 1 of Table 25-2. Figure 25-7 on page 283 shows a Linked List example with two list items.
5. Make sure that the LLI.DMAC_LLPx 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_SARx/LLI.DMAC_DARx register locations of all LLI entries in memory point to the start source/destination block address preceding that LLI fetch.
7. Make sure that the LLI.DMAC_CTLx.DONE field of the LLI.DMAC_CTLx register locations of all LLI entries in memory are cleared.
8. If gather is enabled (DMAC_CTLx.S_GATH_EN is enabled), program the DMAC_SGRx register for channel x.
9. If scatter is enabled (DMAC_CTLx.D_SCAT_EN is enabled), program the DMAC_DSRx register for channel x.
10. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: DMAC_ClearTfr, DMAC_ClearBlock, DMAC_ClearSrcTran, DMAC_ClearDstTran, DMAC_ClearErr. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
11. Program the DMAC_CTLx, DMAC_CFGx registers according to Row 10 as shown in Table 25-2 on page 277.
12. Program the DMAC_LLPx register with DMAC_LLPx(0), the pointer to the first Linked List item.
13. Finally, enable the channel by writing a '1' to the DMAC_ChEnReg.CH_EN bit. The transfer is performed.
14. The DMAC fetches the first LLI from the location pointed to by DMAC_LLPx(0).

Note: The LLI.DMAC_SARx, LLI. DMAC_DARx, LLI.DMAC_LLPx and LLI.DMAC_CTLx registers are fetched. The DMAC automatically reprograms the DMAC_SARx, DMAC_DARx, DMAC_LLPx and DMAC_CTLx channel registers from the DMAC_LLPx(0).

15. Source and destination request single and burst DMA transactions to transfer the block of data (assuming non-memory peripheral). The DMAC acknowledges at the completion of every transaction (burst and single) in the block and carry out the block transfer.
16. The DMAC does not wait for the block interrupt to be cleared, but continues fetching the next LLI from the memory location pointed to by current DMAC_LLPx register and auto-

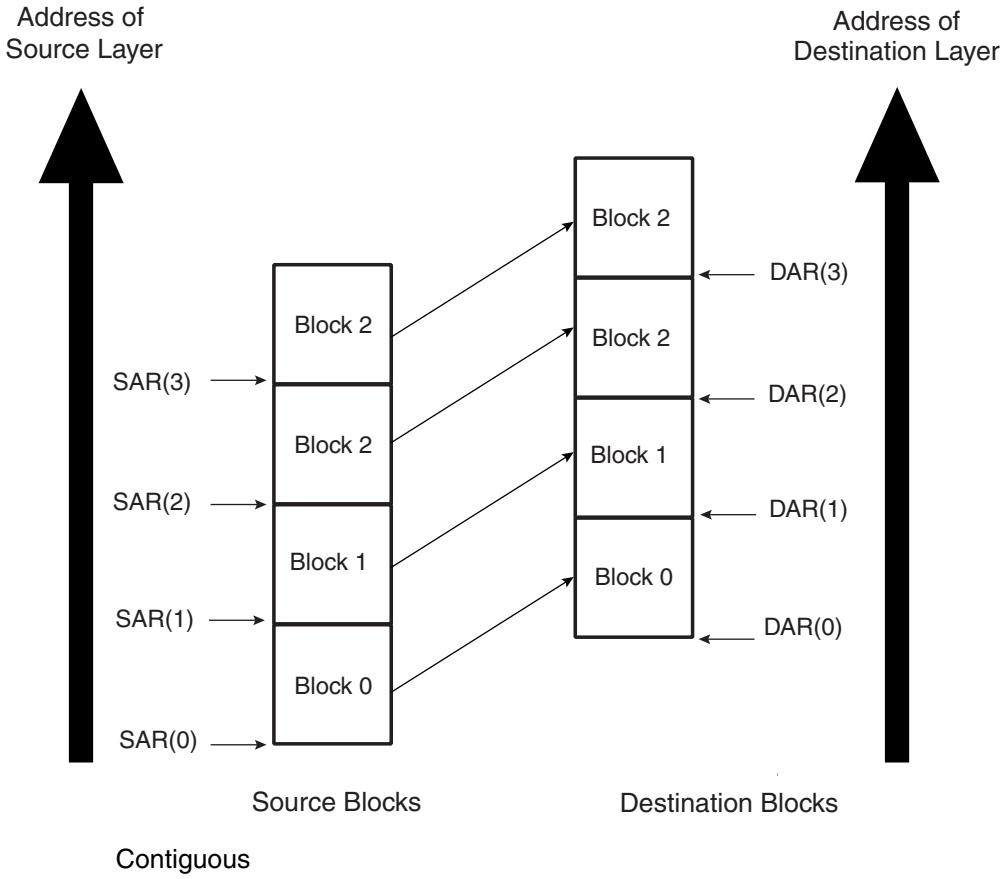
matically reprograms the DMAC_SARx, DMAC_DARx, DMAC_LLPx and DMAC_CTLx channel registers. The DMA transfer continues until the DMAC determines that the DMAC_CTLx and DMAC_LLPx registers at the end of a block transfer match that described in Row 1 of Table 25-2 on page 277. The DMAC then knows that the previous block transferred was the last block in the DMA transfer. The DMA transfer might look like that shown in Figure 25-6 on page 282.

Figure 25-6. Multi-Block with Linked List Address for Source and Destination



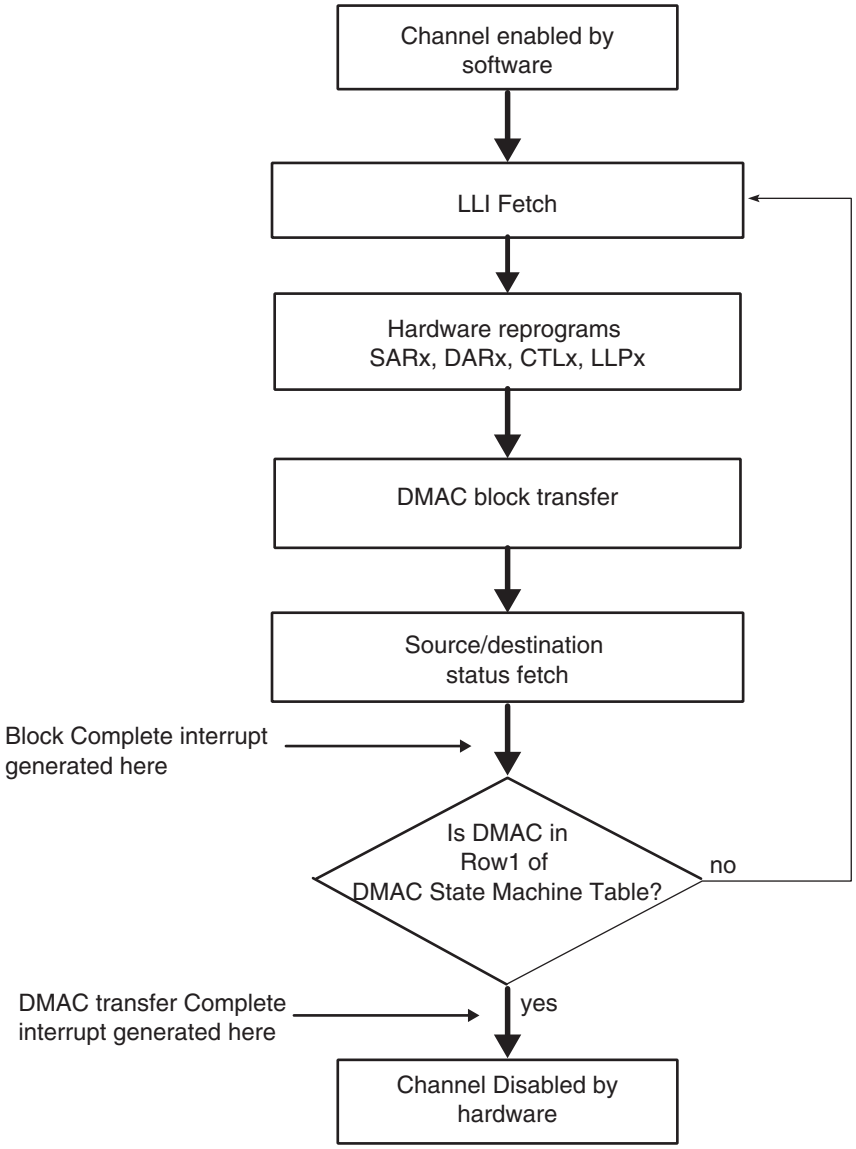
If the user needs to execute a DMA transfer where the source and destination address are contiguous but the amount of data to be transferred is greater than the maximum block size DMAC_CTLx.BLOCK_TS, then this can be achieved using the type of multi-block transfer as shown in Figure 25-7 on page 283.

Figure 25-7. Multi-Block with Linked Address for Source and Destination Blocks are



The DMA transfer flow is shown in Figure 25-8 on page 284.

Figure 25-8. DMA Transfer Flow for Source and Destination Linked List Address



25.3.5.4 Multi-block Transfer with Source Address Auto-reloaded and Destination Address Auto-reloaded (Row 4)

1. Read the Channel Enable register to choose an available (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: DMAC_ClearTfr, DMAC_ClearBlock, DMAC_ClearSrcTran, DMAC_ClearDstTran, DMAC_ClearErr. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
3. Program the following channel registers:

a. Write the starting source address in the DMAC_SARx register for channel x.
b. Write the starting destination address in the DMAC_DARx register for channel x.
c. Program DMAC_CTLx and DMAC_CFGx according to Row 4 as shown in Table 25-2 on page 277. Program the DMAC_LLPx register with '0'.
d. Write the control information for the DMA transfer in the DMAC_CTLx 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 TT_FC of the DMAC_CTLx register.
– ii. Set up the transfer characteristics, such as:
  – Transfer width for the source in the SRC_TR_WIDTH field.
  – Transfer width for the destination in the DST_TR_WIDTH field.
  – Source master layer in the SMS field where source resides.
  – Destination master layer in the DMS field where destination resides.
  – Incrementing/decrementing or fixed address for source in SINC field.
  – Incrementing/decrementing or fixed address for destination in DINC field.

e. If gather is enabled (DMAC_CTLx.S_GATH_EN is enabled), program the DMAC_SGRx register for channel x.
f. If scatter is enabled (DMAC_CTLx.D_SCAT_EN), program the DMAC_DSRx 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. RELOAD_SR and DMAC_CFGx.RELOAD_DS 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 HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a '0' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '1' 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 DEST_PER bits, respectively.

4. After the DMAC selected channel has been programmed, enable the channel by writing a '1' to the DMAC_ChEnReg.CH_EN bit. Make sure that bit 0 of the DMAC_DmaCfgReg register is enabled.
5. Source and destination request single and burst DMAC transactions to transfer the block of data (assuming non-memory peripherals). The DMAC acknowledges on completion of each burst/single transaction and carry out the block transfer.
6. When the block transfer has completed, the DMAC reloads the DMAC_SARx, DMAC_DARx and DMAC_CTLx registers. Hardware sets the Block Complete interrupt. The DMAC then samples the row number as shown in Table 25-2 on page 277. If the DMAC is in Row 1, then the DMA transfer has completed. Hardware sets the transfer complete interrupt and disables the channel. So you can either respond to the Block Complete or Transfer Complete interrupts, or poll for the Channel Enable (DMAC_ChEnReg.CH_EN) bit until it is disabled, to detect when the transfer is complete. If the DMAC is not in Row 1, the next step is performed.
7. The DMA transfer proceeds as follows:

a. If interrupts are enabled (DMAC_CTLx.INT_EN = 1) and the block complete interrupt is un-masked (DMAC_MaskBlock[x] = 1'b1, where x is the channel number) hardware sets the block complete interrupt when the block transfer has completed. It then stalls until the block complete interrupt is cleared by software. If the next block is to be the last block in the DMA transfer, then the block complete ISR (interrupt service routine) should clear the reload bits in the DMAC_CFGx.RELOAD_SR and DMAC_CFGx.RELOAD_DS registers. This put the DMAC into Row 1 as shown in Table 25-2 on page 277. If the next block is not the last block in the DMA transfer, then the reload bits should remain enabled to keep the DMAC in Row 4.

b. If interrupts are disabled (DMAC_CTLx.INT_EN = 0) or the block complete interrupt is masked (DMAC_MaskBlock[x] = 1'b0, where x is the channel number), then hardware does not stall until it detects a write to the block complete interrupt clear register but starts the next block transfer immediately. In this case software must clear the reload bits in the DMAC_CFGx.RELOAD_SR and DMAC_CFGx.RELOAD_DS registers to put the DMAC into ROW 1 of Table 25-2 on page 277 before the last block of the DMA transfer has completed. The transfer is similar to that shown in Figure 25-9 on page 286. The DMA transfer flow is shown in Figure 25-10 on page 287.

Figure 25-9. Multi-Block DMA Transfer with Source and Destination Address Auto-reloaded

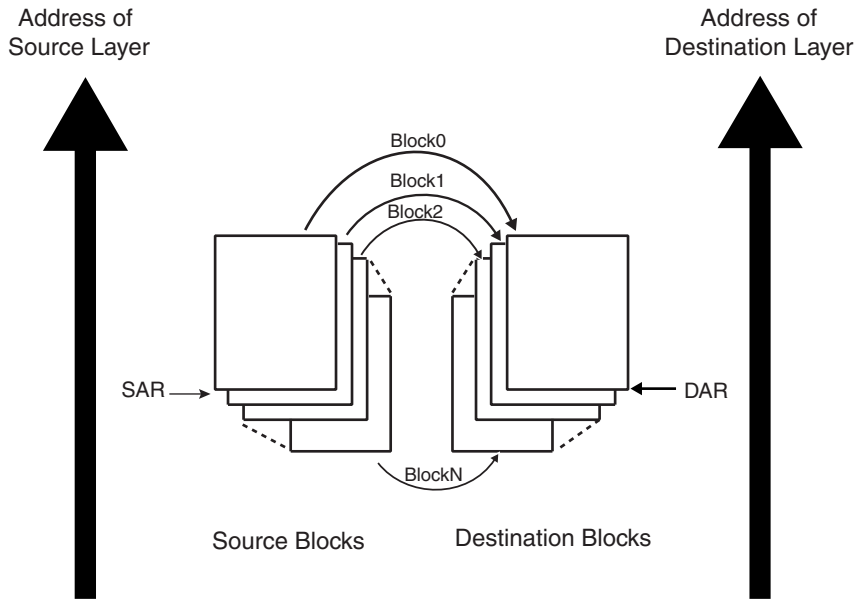
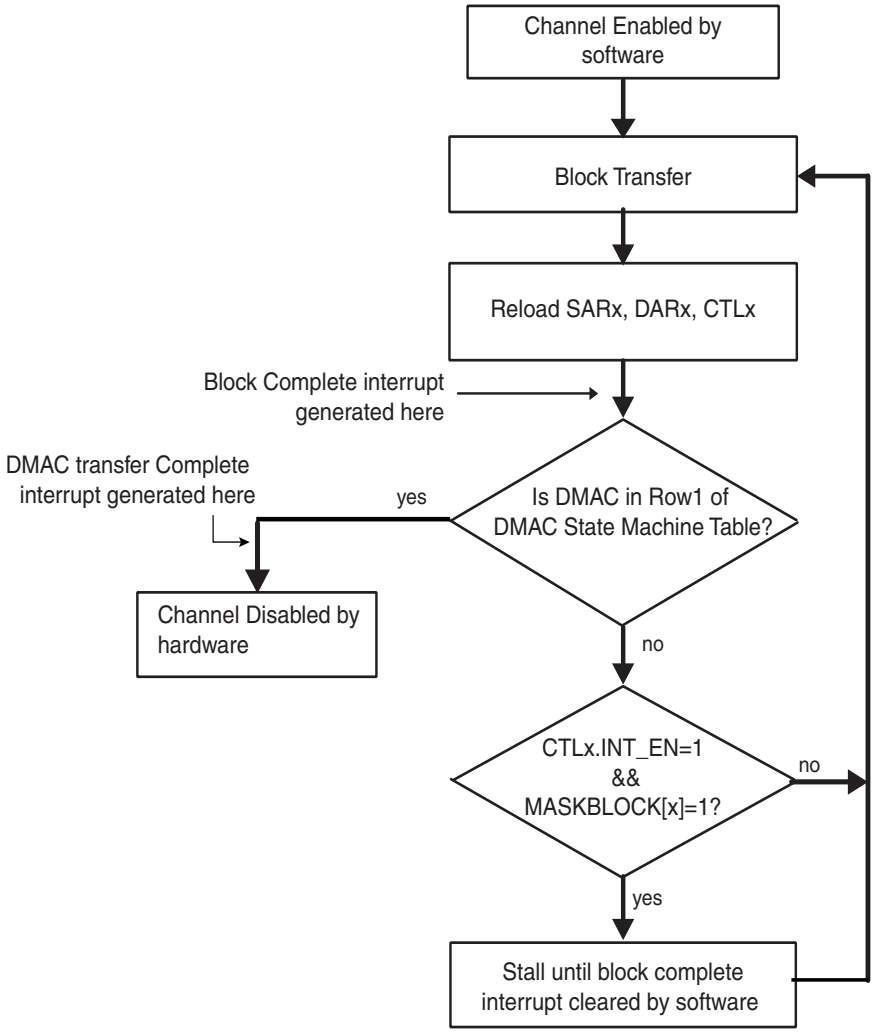


Figure 25-10. DMA Transfer Flow for Source and Destination Address Auto-reloaded



25.3.5.5 Multi-block Transfer with Source Address Auto-reloaded and Linked List Destination Address (Row7)

1. Read the Channel Enable register to choose a free (disabled) channel.
2. Set up the chain of linked list items (otherwise known as block descriptors) in memory. Write the control information in the LLI.DMAC_CTLx register location of the block 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 TT_FC of the DMAC_CTLx register.
   b. Set up the transfer characteristics, such as:
   – i. Transfer width for the source in the SRC_TR_WIDTH field.
   – ii. Transfer width for the destination in the DST_TR_WIDTH field.
   – iii. Source master layer in the SMS field where source resides.
   – iv. Destination master layer in the DMS field where destination resides.
   – v. Incrementing/decrementing or fixed address for source in SINC field.

– vi. Incrementing/decrementing or fixed address for destination DINC field.

3. Write the starting source address in the DMAC_SARx register for channel x.

Note: The values in the LLI.DMAC_SARx register locations of each of the Linked List Items (LLIs) setup up in memory, although fetched during a LLI fetch, are not used.

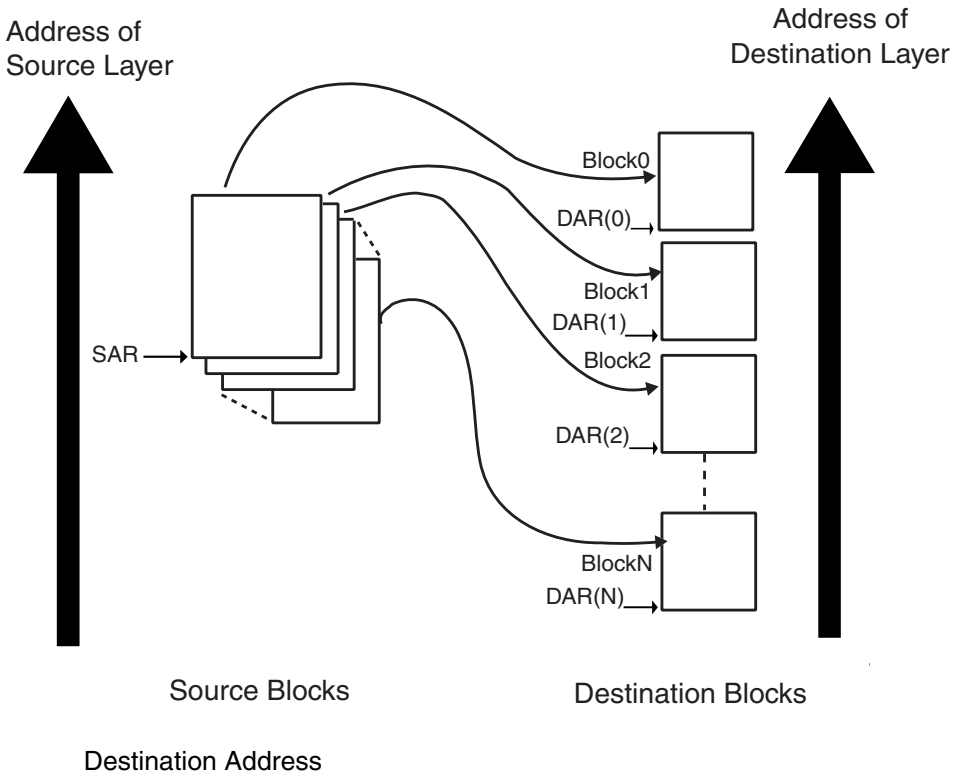
4. Write the channel configuration information into the DMAC_CFGx register for channel x.
   a. Designate the handshaking interface type (hardware or software) for the source and destination peripherals. This is not required for memory. This step requires programming the HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a '0' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '1' 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 DEST_PER bits, respectively.
5. Make sure that the LLI.DMAC_CTLx register locations of all LLIs in memory (except the last) are set as shown in Row 7 of Table 25-2 on page 277 while the LLI.DMAC_CTLx register of the last Linked List item must be set as described in Row 1 of Table 25-2. Figure 25-8 on page 284 shows a Linked List example with two list items.
6. Make sure that the LLI.DMAC_LLPx 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_DARx register location of all LLIs in memory point to the start destination block address proceeding that LLI fetch.
8. Make sure that the LLI.DMAC_CTLx.DONE field of the LLI.DMAC_CTLx register locations of all LLIs in memory is cleared.
9. If gather is enabled (DMAC_CTLx.S_GATH_EN is enabled), program the DMAC_SGRx register for channel x.
10. If scatter is enabled (DMAC_CTLx.D_SCAT_EN is enabled), program the DMAC_DSRx register for channel x.
11. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: DMAC_ClearTfr, DMAC_ClearBlock, DMAC_ClearSrcTran, DMAC_ClearDstTran, DMAC_ClearErr. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
12. Program the DMAC_CTLx, DMAC_CFGx registers according to Row 7 as shown in Table 25-2 on page 277.
13. Program the DMAC_LLPx register with DMAC_LLPx(0), the pointer to the first Linked List item.
14. Finally, enable the channel by writing a '1' to the DMAC_ChEnReg.CH_EN bit. The transfer is performed. Make sure that bit 0 of the DMAC_DmaCfgReg register is enabled.
15. The DMAC fetches the first LLI from the location pointed to by DMAC_LLPx(0).

Note: The LLI.DMAC_SARx, LLI.DMAC_DARx, LLI. DMAC_LLPx and LLI.DMAC_CTLx registers are fetched. The LLI.DMAC_SARx register although fetched is not used.

16. Source and destination request single and burst DMAC transactions to transfer the block of data (assuming non-memory peripherals). DMAC acknowledges at the completion of every transaction (burst and single) in the block and carry out the block transfer.

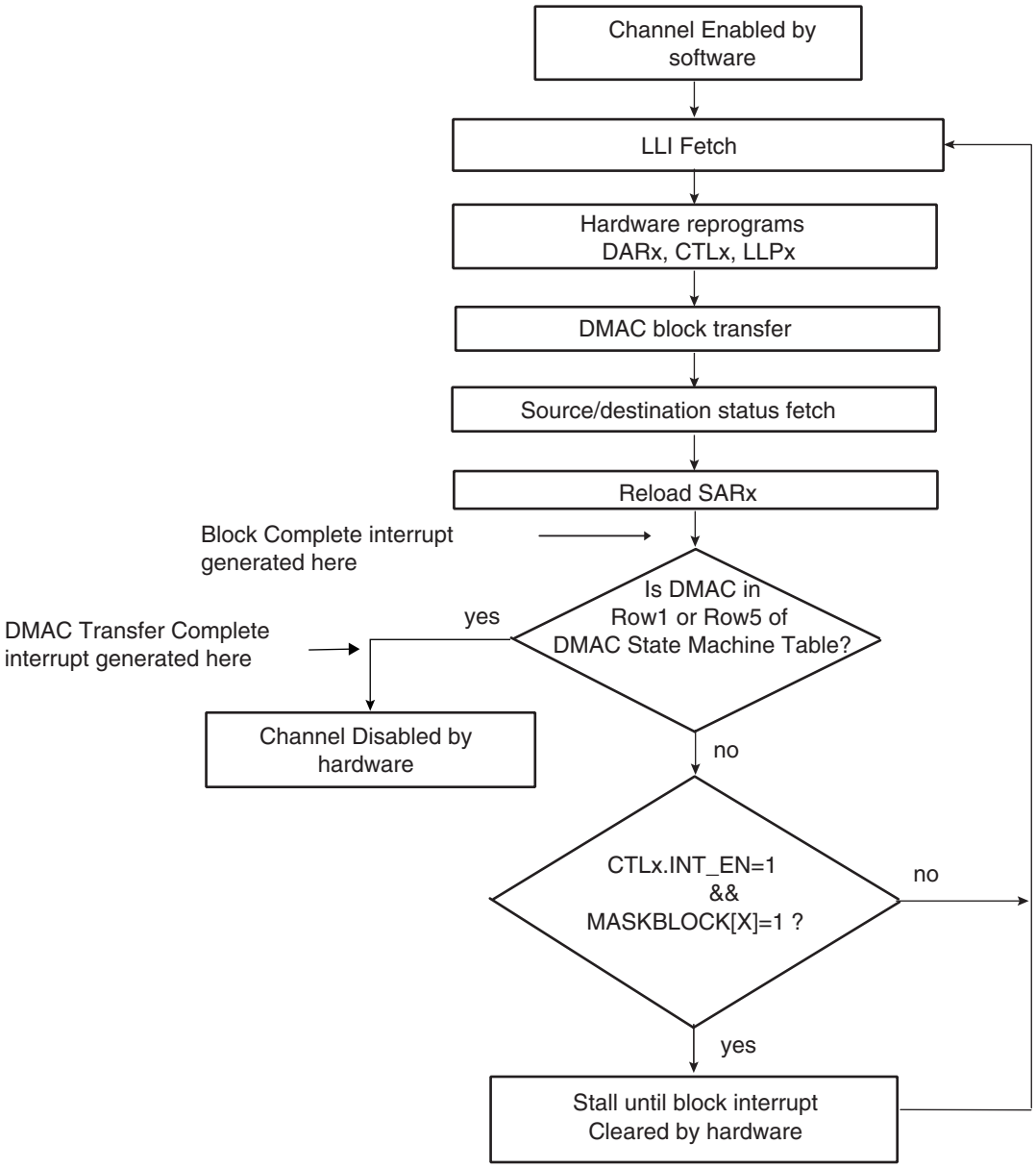
17. The DMAC reloads the DMAC_SARx register from the initial value. Hardware sets the block complete interrupt. The DMAC samples the row number as shown in Table 25-2 on page 277. If the DMAC is in Row 1 or 5, then the DMA transfer has completed. Hardware sets the transfer complete interrupt and disables the channel. You can either respond to the Block Complete or Transfer Complete interrupts, or poll for the Channel Enable (DMAC_ChEnReg.CH_EN) bit until it is cleared by hardware, to detect when the transfer is complete. If the DMAC is not in Row 1 or 5 as shown in Table 25-2 on page 277 the following steps are performed.
18. The DMA transfer proceeds as follows:
    a. If interrupts are enabled (DMAC_CTLx.INT_EN = 1) and the block complete interrupt is un-masked (DMAC_MaskBlock[x] = 1'b1, where x is the channel number) hardware sets the block complete interrupt when the block transfer has completed. It then stalls until the block complete interrupt is cleared by software. If the next block is to be the last block in the DMA transfer, then the block complete ISR (interrupt service routine) should clear the DMAC_CFGx.RELOAD_SR source reload bit. This puts the DMAC into Row1 as shown in Table 25-2 on page 277. If the next block is not the last block in the DMA transfer, then the source reload bit should remain enabled to keep the DMAC in Row 7 as shown in Table 25-2 on page 277.
    b. If interrupts are disabled (DMAC_CTLx.INT_EN = 0) or the block complete interrupt is masked (DMAC_MaskBlock[x] = 1'b0, where x is the channel number) then hardware does not stall until it detects a write to the block complete interrupt clear register but starts the next block transfer immediately. In this case, software must clear the source reload bit, DMAC_CFGx.RELOAD_SR, to put the device into Row 1 of Table 25-2 on page 277 before the last block of the DMA transfer has completed.
19. The DMAC fetches the next LLI from memory location pointed to by the current DMAC_LLPx register, and automatically reprograms the DMAC_DARx, DMAC_CTLx and DMAC_LLPx channel registers. Note that the DMAC_SARx is not re-programmed as the reloaded value is used for the next DMA block transfer. If the next block is the last block of the DMA transfer then the DMAC_CTLx and DMAC_LLPx registers just fetched from the LLI should match Row 1 of Table 25-2 on page 277. The DMA transfer might look like that shown in Figure 25-11 on page 290.

Figure 25-11. Multi-Block DMA Transfer with Source Address Auto-reloaded and Linked List



The DMA Transfer flow is shown in Figure 25-12 on page 291

Figure 25-12. DMA Transfer Flow for Source Address Auto-reloaded and Linked List Destination Address



25.3.5.6 Multi-block Transfer with Source Address Auto-reloaded and Contiguous Destination Address (Row 3)

1. Read the Channel Enable register to choose a free (disabled) channel.
2. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: DMAC_ClearTfr, DMAC_ClearBlock, DMAC_ClearSrcTran, DMAC_ClearDstTran, DMAC_ClearErr. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
3. Program the following channel registers:

a. Write the starting source address in the DMAC_SARx register for channel x.

b. Write the starting destination address in the DMAC_DARx register for channel x.

c. Program DMAC_CTLx and DMAC_CFGx according to Row 3 as shown in Table 25-2 on page 277. Program the DMAC_LLPx register with '0'.

d. Write the control information for the DMA transfer in the DMAC_CTLx 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 TT_FC of the DMAC_CTLx register.

– ii. Set up the transfer characteristics, such as:

  – Transfer width for the source in the SRC_TR_WIDTH field.

  – Transfer width for the destination in the DST_TR_WIDTH field.

  – Source master layer in the SMS field where source resides.

  – Destination master layer in the DMS field where destination resides.

  – Incrementing/decrementing or fixed address for source in SINC field.

  – Incrementing/decrementing or fixed address for destination in DINC field.

e. If gather is enabled (DMAC_CTLx.S_GATH_EN is enabled), program the DMAC_SGRx register for channel x.

f. If scatter is enabled (DMAC_CTLx.D_SCAT_EN), program the DMAC_DSRx 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 HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a '0' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '1' 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 DEST_PER bits, respectively.

4. After the DMAC channel has been programmed, enable the channel by writing a '1' to the DMAC_ChEnReg.CH_EN bit. Make sure that bit 0 of the DMAC_DmaCfgReg register is enabled.

5. Source and destination request single and burst DMAC transactions to transfer the block of data (assuming non-memory peripherals). The DMAC acknowledges at the completion of every transaction (burst and single) in the block and carries out the block transfer.

6. When the block transfer has completed, the DMAC reloads the DMAC_SARx register. The DMAC_DARx register remains unchanged. Hardware sets the block complete interrupt. The DMAC then samples the row number as shown in Table 25-2 on page 277. If the DMAC is in Row 1, then the DMA transfer has completed. Hardware sets the transfer complete interrupt and disables the channel. So you can either respond to the Block Complete or Transfer Complete interrupts, or poll for the Channel Enable (DMAC_ChEnReg.CH_EN) 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 DMA transfer proceeds as follows:

a. If interrupts are enabled (DMAC_CTLx.INT_EN = 1) and the block complete interrupt is un-masked (DMAC_MaskBlock[x] = 1'b1, where x is the channel number) hardware sets the block complete interrupt when the block transfer has completed. It then stalls until the block complete interrupt is cleared by software. If the next block is to be the last block in the DMA transfer, then the block complete ISR (interrupt service routine) should clear the source reload bit, DMAC_CFGx.RELOAD_SR. This puts the DMAC into Row1 as shown in Table 25-2 on page 277. If the next block is not the last block in the DMA transfer then the source reload bit should remain enabled to keep the DMAC in Row3 as shown in Table 25-2 on page 277.

b. If interrupts are disabled (DMAC_CTLx.INT_EN = 0) or the block complete interrupt is masked (DMAC_MaskBlock[x] = 1'b0, where x is the channel number) then hardware does not stall until it detects a write to the block complete interrupt clear register but starts the next block transfer immediately. In this case software must clear the source reload bit, DMAC_CFGx.RELOAD_SR, to put the device into ROW 1 of Table 25-2 on page 277 before the last block of the DMA transfer has completed.

The transfer is similar to that shown in Figure 25-13 on page 293.

The DMA Transfer flow is shown in Figure 25-14 on page 294.

Figure 25-13. Multi-block Transfer with Source Address Auto-reloaded and Contiguous Destination Address

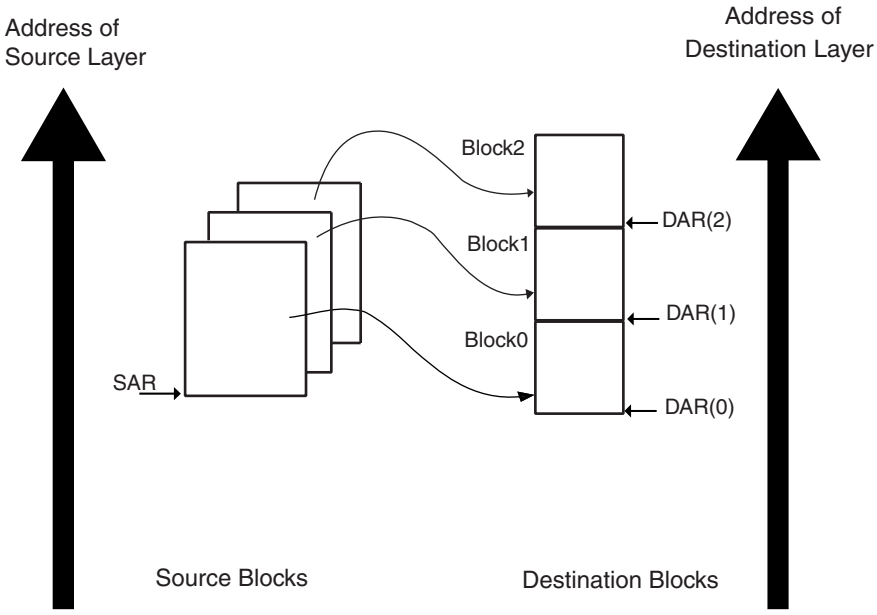
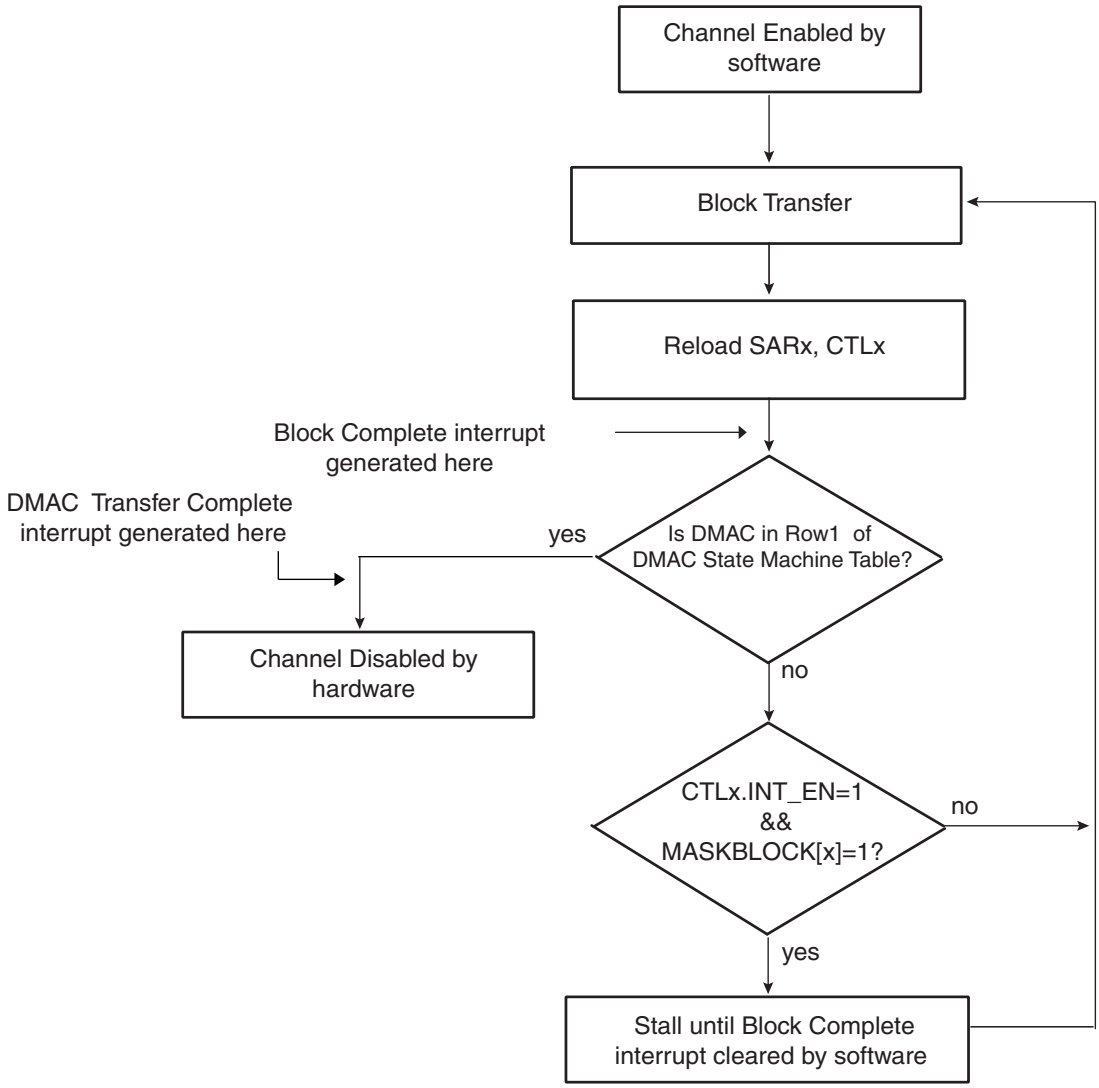


Figure 25-14. DMA Transfer for Source Address Auto-reloaded and Contiguous Destination Address



25.3.5.7 Multi-block DMA Transfer with Linked List for Source and Contiguous Destination Address (Row 8)

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_CTLx register location of the block 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 TT_FC of the DMAC_CTLx register.
   b. Set up the transfer characteristics, such as:
   – i. Transfer width for the source in the SRC_TR_WIDTH field.
   – ii. Transfer width for the destination in the DST_TR_WIDTH field.
   – iii. Source master layer in the SMS field where source resides.
   – iv. Destination master layer in the DMS field where destination resides.

– v. Incrementing/decrementing or fixed address for source in SINC field.

– vi. Incrementing/decrementing or fixed address for destination DINC field.

3. Write the starting destination address in the DMAC_DARx register for channel x.

Note: The values in the LLI.DMAC_DARx 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 HS_SEL_SRC/HS_SEL_DST bits, respectively. Writing a '0' activates the hardware handshaking interface to handle source/destination requests for the specific channel. Writing a '1' 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 DEST_PER bits, respectively.
5. Make sure that all LLI.DMAC_CTLx register locations of the LLI (except the last) are set as shown in Row 8 of Table 25-2 on page 277, while the LLI.DMAC_CTLx register of the last Linked List item must be set as described in Row 1 of Table 25-2. Figure 25-8 on page 284 shows a Linked List example with two list items.
6. Make sure that the LLI.DMAC_LLPx 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_SARx register location of all LLIs in memory point to the start source block address proceeding that LLI fetch.
8. Make sure that the LLI.DMAC_CTLx.DONE field of the LLI.DMAC_CTLx register locations of all LLIs in memory is cleared.
9. If gather is enabled (DMAC_CTLx.S_GATH_EN is enabled), program the DMAC_SGRx register for channel x.
10. If scatter is enabled (DMAC_CTLx.D_SCAT_EN is enabled), program the DMAC_DSRx register for channel x.
11. Clear any pending interrupts on the channel from the previous DMA transfer by writing to the Interrupt Clear registers: DMAC_ClearTfr, DMAC_ClearBlock, DMAC_ClearSrcTran, DMAC_ClearDstTran, DMAC_ClearErr. Reading the Interrupt Raw Status and Interrupt Status registers confirms that all interrupts have been cleared.
12. Program the DMAC_CTLx, DMAC_CFGx registers according to Row 8 as shown in Table 25-2 on page 277
13. Program the DMAC_LLPx register with DMAC_LLPx(0), the pointer to the first Linked List item.
14. Finally, enable the channel by writing a '1' to the DMAC_ChEnReg.CH_EN bit. The transfer is performed. Make sure that bit 0 of the DMAC_DmaCfgReg register is enabled.
15. The DMAC fetches the first LLI from the location pointed to by DMAC_LLPx(0).

Note: The LLI.DMAC_SARx, LLI.DMAC_DARx, LLI.DMAC_LLPx and LLI.DMAC_CTLx registers are fetched. The LLI.DMAC_DARx register location of the LLI although fetched is not used. The DMAC_DARx register in the DMAC remains unchanged.

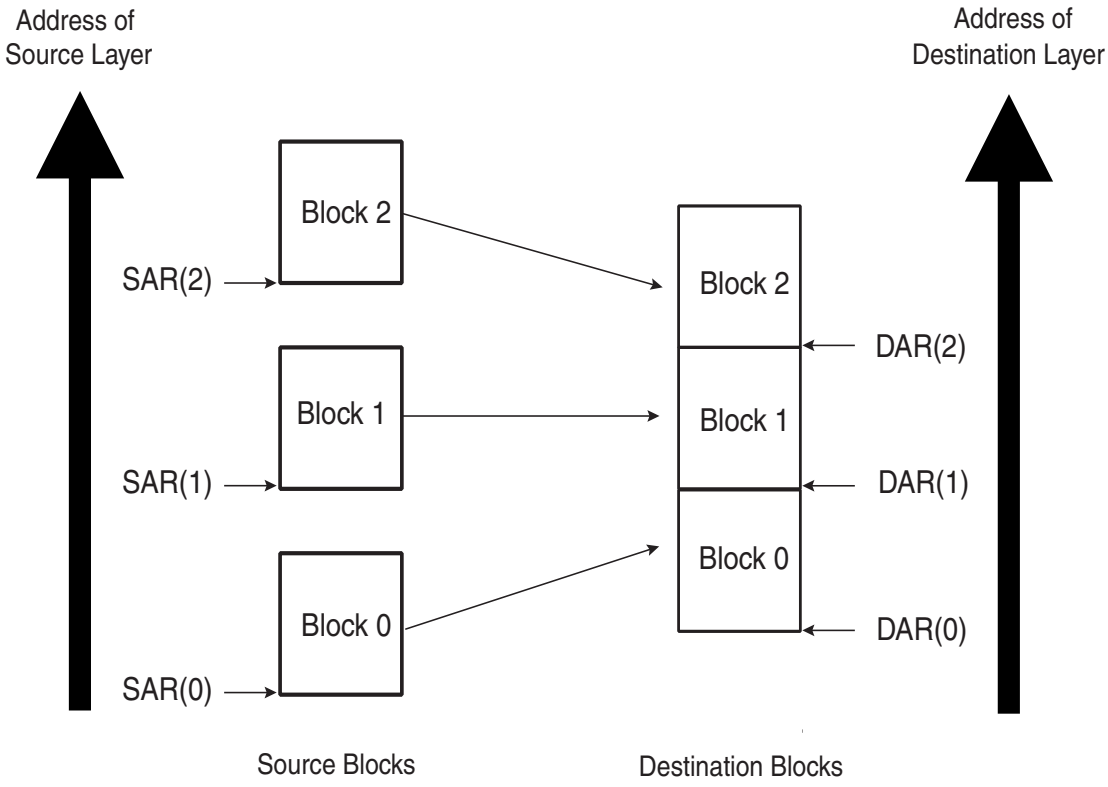
16. Source and destination requests single and burst DMAC transactions to transfer the block of data (assuming non-memory peripherals). The DMAC acknowledges at the

completion of every transaction (burst and single) in the block and carry out the block transfer.

17. The DMAC does not wait for the block interrupt to be cleared, but continues and fetches the next LLI from the memory location pointed to by current DMAC_LLPx register and automatically reprograms the DMAC_SARx, DMAC_CTLx and DMAC_LLPx channel registers. The DMAC_DARx register is left unchanged. The DMA transfer continues until the DMAC samples the DMAC_CTLx and DMAC_LLPx registers at the end of a block transfer match that described in Row 1 of Table 25-2 on page 277. The DMAC then knows that the previous block transferred was the last block in the DMA transfer.

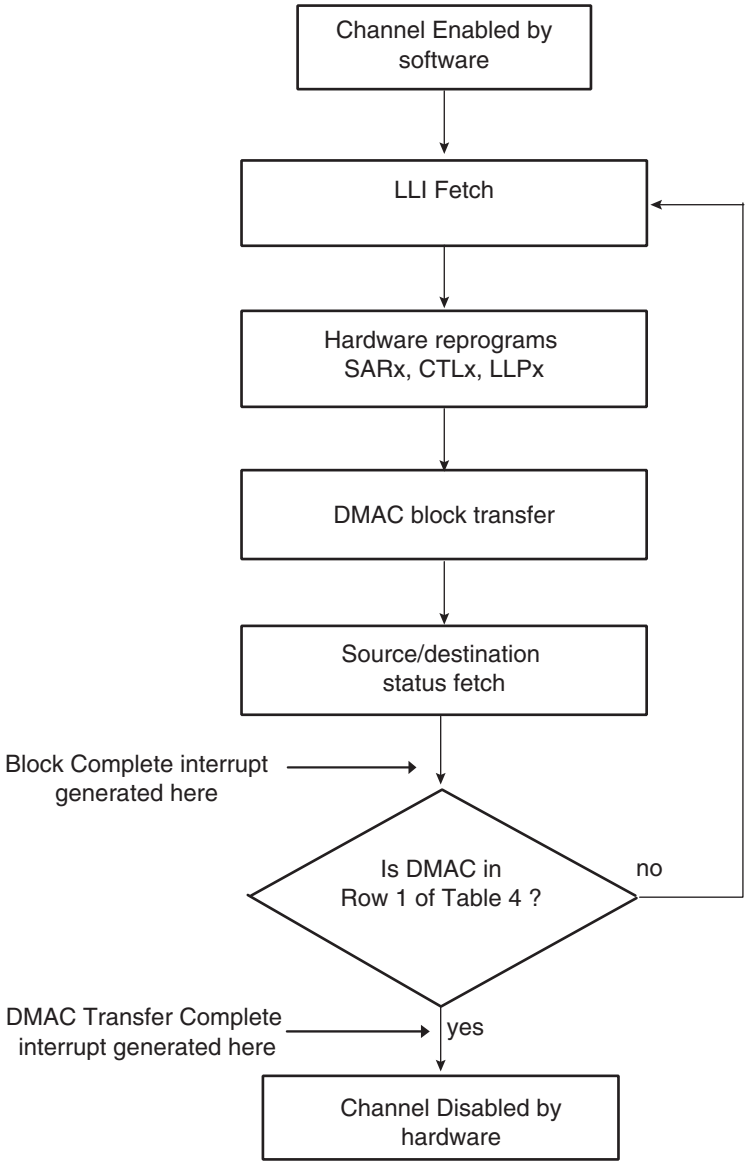
The DMAC transfer might look like that shown in Figure 25-15 on page 296 Note that the destination address is decrementing.

Figure 25-15. DMA Transfer with Linked List Source Address and Contiguous Destination Address



The DMA transfer flow is shown in Figure 25-16 on page 297.

Figure 25-16. DMA Transfer Flow for Source Address Auto-reloaded and Contiguous Destination Address



25.3.6 Disabling a Channel Prior to Transfer Completion

Under normal operation, software enables a channel by writing a '1' to the Channel Enable Register, DMAC_ChEnReg.CH_EN, and hardware disables a channel on transfer completion by clearing the DMAC_ChEnReg.CH_EN register bit.

The recommended way for software to disable a channel without losing data is to use the CH_SUSP bit in conjunction with the FIFO_EMPTY bit in the Channel Configuration Register (DMAC_CFGx) register.

1. If software wishes to disable a channel prior to the DMA transfer completion, then it can set the DMAC_CFGx.CH_SUSP 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_CFGx.FIFO_EMPTY bit until it indicates that the channel FIFO is empty.
3. The DMAC_ChEnReg.CH_EN bit can then be cleared by software once the channel FIFO is empty.

When DMAC_CTLx.SRC_TR_WIDTH is less than DMAC_CTLx.DST_TR_WIDTH and the DMAC_CFGx.CH_SUSP bit is high, the DMAC_CFGx.FIFO_EMPTY is asserted once the contents of the FIFO do not permit a single word of DMAC_CTLx.DST_TR_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_TR_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 '0' to the DMAC_CFGx.CH_SUSP register. The DMA transfer completes in the normal manner.

Note: If a channel is disabled by software, an active single or burst transaction is not guaranteed to receive an acknowledgement.

25.3.6.1 Abnormal Transfer Termination

A DMAC DMA transfer may be terminated abruptly by software by clearing the channel enable bit, DMAC_ChEnReg.CH_EN. This does not mean that the channel is disabled immediately after the DMAC_ChEnReg.CH_EN bit is cleared over the AHB slave interface. Consider this as a request to disable the channel. The DMAC_ChEnReg.CH_EN must be polled and then it must be confirmed that the channel is disabled by reading back 0. A case where the channel is not be disabled after a channel disable request is where either the source or destination has received a split or retry response. The DMAC must keep re-attempting the transfer to the system HADDR that originally received the split or retry response until an OKAY response is returned. To do otherwise is an AMBA protocol violation.

Software may terminate all channels abruptly by clearing the global enable bit in the DMAC Configuration Register (DMAC_DmaCfgReg[0]). Again, this does not mean that all channels are disabled immediately after the DMAC_DmaCfgReg[0] is cleared over the AHB slave interface. Consider this as a request to disable all channels. The DMAC_ChEnReg 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 burst transaction is not guaranteed to receive an acknowledgement.

25.4 DMA Controller (DMAC) User Interface

Table 25-3. Register Mapping

Offset	Register	Name	Access	Reset
0x0	Channel 0 Source Address Register	DMAC_SAR0	Read-write	0x0
0x4	Reserved			-
0x8	Channel 0 Destination Address Register	DMAC_DAR0	Read-write	0x0
0xC	Reserved			-
0x10	Channel 0 Linked List Pointer Register	DMAC_LLP0	Read-write	0x0
0x14	Reserved			-
0x18	Channel 0 Control Register Low	DMAC_CTL0L	Read-write	
0x1C	Channel 0 Control Register High	DMAC_CTL0H	Read-write	
0x20 - 0x3C	Reserved			
0x40	Channel 0 Configuration Register low	DMAC_CFG0L	Read-write	0x00000c00
0x44	Channel 0 Configuration Register High	DMAC_CFG0H	Read-write	0x00000004
0x48	Channel 0 Source Gather Register	DMAC_SGR0	Read-write	0x0
0x4C	Reserved			
0x50	Channel 0 Destination Scatter Register	DMAC_DSR0	Read-write	0x0
0x54	Reserved			
0x58	Channel 1 Source Address Register	DMAC_SAR1	Read-write	0x0
0x5C	Reserved			
0x60	Channel 1 Destination Address Register	DMAC_DAR1	Read-write	0x0
0x64	Reserved			
0x68	Channel 1 Linked List Pointer Register	DMAC_LLP1	Read-write	0x0
0x7C	Reserved			
0x70	Channel 1 Control Register Low	DMAC_CTL1L	Read-write	
0x74	Channel 1 Control Register High	DMAC_CTL1H	Read-write	
0x78 - 0x94	Reserved			
0x98	Channel 1 Configuration Register Low	DMAC_CFG1L	Read-write	0x00000c20
0x9C	Channel 1 Configuration Register High	DMAC_CFG1H	Read-write	0x00000004
0xa0	Channel 1 Source Gather Register	DMAC_SGR1	Read-write	0x0
0xa4	Reserved			
0xa8	Channel 1 Destination Scatter Register	DMAC_DSR1	Read-write	0x0
0xac..0x2bc	Reserved			
0x2c0	Raw Status for IntTfr Interrupt	DMAC_RawTfr	Read	0x0
0x2c4	Reserved			
0x2c8	Raw Status for IntBlock Interrupt	DMAC_RawBlock	Read	0x0
0x2cc	Reserved			

Table 25-3. Register Mapping

Offset	Register	Name	Access	Reset
0x2d0	Raw Status for IntSrcTran Interrupt	DMAC_RawSrcTran	Read	0x0
0x2d4	Reserved			
0x2d8	Raw Status for IntDstTran Interrupt	DMAC_RawDstTran	Read	0x0
0x2dc	Reserved			
0x2e0	Raw Status for IntErr Interrupt	DMAC_RawErr	Read	0x0
0x2e4	Reserved			
0x2e8	Status for IntTfr Interrupt	DMAC_StatusTfr	Read	0x0
0x2ec	Reserved			
0x2f0	Status for IntBlock Interrupt	DMAC_StatusBlock	Read	0x0
0x2f4	Reserved			
0x2f8	Status for IntSrcTran Interrupt	DMAC_StatusSrcTran	Read	0x0
0x2fc	Reserved			
0x300	Status for IntDstTran Interrupt	DMAC_StatusDstTran	Read	0x0
0x304	Reserved			
0x308	Status for IntErr Interrupt	DMAC_StatusErr	Read	0x0
0x30c	Reserved			
0x310	Mask for IntTfr Interrupt	DMAC_MaskTfr	Read-write	0x0
0x314	Reserved			
0x318	Mask for IntBlock Interrupt	DMAC_MaskBlock	Read-write	0x0
0x31c	Reserved			
0x320	Mask for IntSrcTran Interrupt	DMAC_MaskSrcTran	Read-write	0x0
0x324	Reserved			
0x328	Mask for IntDstTran Interrupt	DMAC_MaskDstTran	Read-write	0x0
0x32c	Reserved			
0x330	Mask for IntErr Interrupt	DMAC_MaskErr	Read-write	0x0
0x334	Reserved			
0x338	Clear for IntTfr Interrupt	DMAC_ClearTfr	Write	0x0
0x33c	Reserved			
0x340	Clear for IntBlock Interrupt	DMAC_ClearBlock	Write	0x0
0x344	Reserved			
0x348	Clear for IntSrcTran Interrupt	DMAC_ClearSrcTran	Write	0x0
0x34c	Reserved			
0x350	Clear for IntDstTran Interrupt	DMAC_ClearDstTran	Write	0x0
0x354	Reserved			
0x358	Clear for IntErr Interrupt	DMAC_ClearErr	Write	0x0
0x35c	Reserved			

Table 25-3. Register Mapping

Offset	Register	Name	Access	Reset
0x360	Status for each interrupt type	DMAC_StatusInt	Read	0x0
0x364	Reserved			
0x368	Source Software Transaction Request Register	DMAC_ReqSrcReg	Read-write	0x0
0x36c	Reserved			
0x370	Destination Software Transaction Request Register	DMAC_ReqDstReg	Read-write	0x0
0x374	Reserved			
0x378	Single Source Transaction Request Register	DMAC_SglReqSrcReg	Read-write	0x0
0x37c	Reserved			
0x380	Single Destination Transaction Request Register	DMAC_SglReqDstReg	Read-write	0x0
0x384	Reserved			
0x388	Last Source Transaction Request Register	DMAC_LstSrcReqReg	Read-write	0x0
0x38c	Reserved			
0x390	Last Destination Transaction Request Register	DMAC_LstDstReqReg	Read-write	0x0
0x394	Reserved			
0x398	DMA Configuration Register	DMAC_DmaCfgReg	Read-write	0x0
0x39c	Reserved			
0x3a0	Channel Enable Register	DMAC_ChEnReg	Read-write	0x0
0x3a4-0x3b8	Reserved			

25.4.1 Channel x Source Address Register

Name: DMAC_SARx

Addresses:0x00800000 [0], 0x00800058 [1]

Access: Read-write

Reset: 0x0

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

The address offset for each channel is: [x *0x58]

For example, SAR0: 0x000, SAR1: 0x058, etc.

- **SADD: Source Address of DMA Transfer**

The starting AMBA source address is programmed by software before the DMA channel is enabled or by a LLI update before the start of the DMA transfer. As the DMA transfer is in progress, this register is updated to reflect the source address of the current AMBA transfer.

Updated after each source AMBA transfer. The SINC field in the DMAC_CTLx register determines whether the address increments, decrements, or is left unchanged on every source AMBA transfer throughout the block transfer.

25.4.2 Channel x Destination Address Register

Name: DMAC_DARx

Addresses:0x00800008 [0], 0x00800060 [1]

Access: Read-write

Reset: 0x0

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

The address offset for each channel is: 0x08+[x * 0x58]

For example, DAR0: 0x008, DAR1: 0x060, etc.

- **DADD: Destination Address of DMA transfer**

The starting AMBA destination address is programmed by software before the DMA channel is enabled or by a LLI update before the start of the DMA transfer. As the DMA transfer is in progress, this register is updated to reflect the destination address of the current AMBA transfer.

Updated after each destination AMBA transfer. The DINC field in the DMAC_CTLx register determines whether the address increments, decrements or is left unchanged on every destination AMBA transfer throughout the block transfer.

25.4.3 Linked List Pointer Register for Channel x

Name: DMAC_LLPx

Addresses:0x00800010 [0], 0x00800068 [1]

Access: Read-write

Reset: 0x0

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

The address offset for each channel is: 0x10+[x * 0x58]

For example, LLP0: 0x010, LLP1: 0x068, etc.

- **LOC: Address of the next LLI**

Starting address in memory of next LLI if block chaining is enabled. Note that the two LSBs of the starting address are not stored because the address is assumed to be aligned to a 32-bit boundary.

The user need to program this register to point to the first Linked List Item (LLI) in memory prior to enabling the channel if block chaining is enabled.

The LLP register has two functions:

1. The logical result of the equation LLP.LOC != 0 is used to set up the type of DMA transfer (single or multi-block).

If LLP.LOC is set to 0x0, then transfers using linked lists are NOT enabled. This register must be programmed prior to enabling the channel in order to set up the transfer type.

It (LLP.LOC != 0) contains the pointer to the next Linked Listed Item for block chaining using linked lists.

2. The DMAC_LLPx register is also used to point to the address where write back of the control and source/destination status information occurs after block completion.

25.4.4 Control Register for Channel x Low

Name: DMAC_CTLxL

Addresses:0x00800018 [0], 0x00800070 [1]

Access: Read-write

Reset: 0x0

31	30	29	28	27	26	25	24
–	–	–	LLP_S_EN	LLP_D_EN	SMS		DMS
23	22	21	20	19	18	17	16
DMS	TT_FC			-	D_SCAT_EN	S_GATH_EN	SRC_MSIZE
15	14	13	12	11	10	9	8
SRC_MSIZE		DEST_MSIZE			SINC		DINC
7	6	5	4	3	2	1	0
DINC	SRC_TR_WIDTH			DST_TR_WIDTH			INT_EN

The address offset for each channel is: 0x18+[x * 0x58]

For example, CTL0: 0x018, CTL1: 0x070, etc.

This register contains fields that control the DMA transfer. The DMAC_CTLxL register is part of the block descriptor (linked list item) when block chaining is enabled. It can be varied on a block-by-block basis within a DMA transfer when block chaining is enabled.

- **INT_EN: Interrupt Enable Bit**

If set, then all five interrupt generating sources are enabled.

- **DST_TR_WIDTH: Destination Transfer Width**
- **SRC_TR_WIDTH: Source Transfer Width**

SRC_TR_WIDTH/DST_TR_WIDTH	Size (bits)
000	8
001	16
010	32
Other	Reserved

- **DINC: Destination Address Increment**

Indicates whether to increment or decrement the destination address on every destination AMBA transfer. If your device is writing data to a destination peripheral FIFO with a fixed address, then set this field to “No change”.

00 = Increment

01 = Decrement

1x = No change

- **SINC*:* Source Address Increment**

Indicates whether to increment or decrement the source address on every source AMBA transfer. If your device is fetching data from a source peripheral FIFO with a fixed address, then set this field to “No change”.

00 = Increment

01 = Decrement

1x = No change

- **DEST_MSIZE: Destination Burst Transaction Length**

Number of data items, each of width *DMAC_CTLx.DST_TR_WIDTH*, to be written to the destination every time a destination burst transaction request is made from either the corresponding hardware or software handshaking interface.

- **SRC_MSIZE: Source Burst Transaction Length**

Number of data items, each of width *DMAC_CTLx.SRC_TR_WIDTH*, to be read from the source every time a source burst transaction request is made from either the corresponding hardware or software handshaking interface.

- **S_GATH_EN*:* Source Gather Enable Bit**

0 = Gather is disabled.

1 = Gather is enabled.

Gather on the source side is only applicable when the DMAC_CTLx.SINC bit indicates an incrementing or decrementing address control.

- **D_SCAT_EN*:* Destination Scatter Enable Bit**

0 = Scatter is disabled.

1 = Scatter is enabled.

Scatter on the destination side is only applicable when the DMAC_CTLx.DINC bit indicates an incrementing or decrementing address control.

- **TT_FC: Transfer Type and Flow Control**

The following transfer types are supported.

• Memory to Memory

• Memory to Peripheral

• Peripheral to Memory

Flow Control can be assigned to the DMAC, the source peripheral, or the destination peripheral.

TT_FC	Transfer Type	Flow Controller
000	Memory to Memory	DMAC
001	Memory to Peripheral	DMAC
010	Peripheral to Memory	DMAC
011	Peripheral to Peripheral	DMAC
100	Peripheral to Memory	Peripheral
101	Peripheral to Peripheral	Source Peripheral
110	Memory to Peripheral	Peripheral
111	Peripheral to Peripheral	Destination Peripheral

- **DMS: Destination Master Select**

Identifies the Master Interface layer where the destination device (peripheral or memory) resides.

00 = AHB master 1

01 = Reserved

10 = Reserved

11 = Reserved

- **SMS: Source Master Select**

Identifies the Master Interface layer where the source device (peripheral or memory) is accessed from.

00 = AHB master 1

01 = Reserved

10 = Reserved

11 = Reserved

- **LLP_D_EN**

Block chaining is only enabled on the destination side if the *LLP_D_EN* field is high and DMAC_LLPx.LOC is non-zero.

- **LLP_S_EN**

Block chaining is only enabled on the source side if the *LLP_S_EN* field is high and DMAC_LLPx.LOC is non-zero.

25.4.5 Control Register for Channel x High

Name: DMAC_CTLxH

Addresses:0x0080001C [0], 0x00800074 [1]

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
–	–	–	DONE	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	BLOCK_TS				

- **BLOCK_TS: Block Transfer Size**

When the DMAC is flow controller, this field is written by the user before the channel is enabled to indicate the block size.

The number programmed into BLOCK_TS indicates the total number of single transactions to perform for every block transfer. The width of the single transaction is determined by DMAC_CTLx.SRC_TR_WIDTH.

- **DONE: Done Bit**

Software can poll the LLI DMAC_CTLx.DONE bit to see when a block transfer is complete. The LLI DMAC_CTLx.DONE bit should be cleared when the linked lists are setup in memory prior to enabling the channel

25.4.6 Configuration Register for Channel x Low

Name: DMAC_CFGxL

Addresses:0x00800040 [0], 0x00800098 [1]

Access: Read-write

Reset: 0x0

31	30	29	28	27	26	25	24
RELOAD_DS	RELOAD_SR	MAX_ABRST					
23	22	21	20	19	18	17	16
MAX_ABRST				SR_HS_POL	DS_HS_POL	LOCK_B	LOCK_CH
15	14	13	12	11	10	9	8
LOCK_B_L		LOCK_CH_L		HS_SEL_SR	HS_SEL_DS	FIFO_EMPT	CH_SUSP
7	6	5	4	3	2	1	0
CH_PRIOR			–	–	–	–	–

The address offset for each channel is: 0x40+[x * 0x58]

For example, CFG0: 0x040, CFG1: 0x098, etc.

- **CH_PRIOR: Channel Priority**

A priority of 7 is the highest priority, and 0 is the lowest. This field must be programmed within the following range [0, x – 1]

A programmed value outside this range causes erroneous behavior.

- **CH_SUSP: Channel Suspend**

Suspends all DMA data transfers from the source until this bit is cleared. There is no guarantee that the current transaction will complete. Can also be used in conjunction with DMAC_CFGx.FIFO_EMPTY to cleanly disable a channel without losing any data.

0 = Not Suspended.

1 = Suspend. Suspend DMA transfer from the source.

- **FIFO_EMPTY**

Indicates if there is data left in the channel's FIFO. Can be used in conjunction with DMAC_CFGx.CH_SUSP to cleanly disable a channel.

1 = Channel's FIFO empty

0 = Channel's FIFO not empty

- **HS_SEL_DST: Destination Software or Hardware Handshaking Select**

This register selects which of the handshaking interfaces, hardware or software, is active for destination requests on this channel.

0 = Hardware handshaking interface. Software-initiated transaction requests are ignored.

1 = Software handshaking interface. Hardware Initiated transaction requests are ignored.

If the destination peripheral is memory, then this bit is ignored.

- **HS_SEL_SRC: Source Software or Hardware Handshaking Select**

This register selects which of the handshaking interfaces, hardware or software, is active for source requests on this channel.

0 = Hardware handshaking interface. Software-initiated transaction requests are ignored.

1 = Software handshaking interface. Hardware-initiated transaction requests are ignored.

If the source peripheral is memory, then this bit is ignored.

- **LOCK_CH_L*:* Channel Lock Level**

Indicates the duration over which DMAC_CFGx.LOCK_CH bit applies.

00 = Over complete DMA transfer

01 = Over complete DMA block transfer

1x = Over complete DMA transaction

- **LOCK_B_L: Bus Lock Level**

Indicates the duration over which DMAC_CFGx.LOCK_B bit applies.

00 = Over complete DMA transfer

01 = Over complete DMA block transfer

1x = Over complete DMA transaction

- **LOCK_CH: Channel Lock Bit**

When the channel is granted control of the master bus interface and if the DMAC_CFGx.LOCK_CH bit is asserted, then no other channels are granted control of the master bus interface for the duration specified in DMAC_CFGx.LOCK_CH_L. Indicates to the master bus interface arbiter that this channel wants exclusive access to the master bus interface for the duration specified in DMAC_CFGx.LOCK_CH_L.

- **LOCK_B: Bus Lock Bit**

When active, the AMBA bus master signal hlock is asserted for the duration specified in DMAC_CFGx.LOCK_B_L.

- **DS_HS_POL: Destination Handshaking Interface Polarity**

0 = Active high

1 = Active low

- **SR_HS_POL: Source Handshaking Interface Polarity**

0 = Active high

1 = Active low

- **MAX_ABRST: Maximum AMBA Burst Length**

Maximum AMBA burst length that is used for DMA transfers on this channel. A value of ‘0’ indicates that software is not limiting the maximum AMBA burst length for DMA transfers on this channel.

- **RELOAD_SR: Automatic Source Reload**

The DMAC_SARx register can be automatically reloaded from its initial value at the end of every block for multi-block transfers. A new block transfer is then initiated.

- **RELOAD_DS: Automatic Destination Reload**

The DMAC_DARx register can be automatically reloaded from its initial value at the end of every block for multi-block transfers. A new block transfer is then initiated.

25.4.7 Configuration Register for Channel x High

Name: DMAC_CFGxH

Addresses:0x00800044 [0], 0x0080009C [1]

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
–	DEST_PER				SRC_PER		
7	6	5	4	3	2	1	0
SRC_PER	–	–	PROTCTL			FIFO_MODE	FCMODE

- **FCMODE: Flow Control Mode**

Determines when source transaction requests are serviced when the Destination Peripheral is the flow controller.

0 = Source transaction requests are serviced when they occur. Data pre-fetching is enabled.

1 = Source transaction requests are not serviced until a destination transaction request occurs. In this mode the amount of data transferred from the source is limited such that it is guaranteed to be transferred to the destination prior to block termination by the destination. Data pre-fetching is disabled.

- **FIFO_MODE: R/W 0x0 FIFO Mode Select**

Determines how much space or data needs to be available in the FIFO before a burst transaction request is serviced.

0 = Space/data available for single AMBA transfer of the specified transfer width.

1 = Space/data available is greater than or equal to half the FIFO depth for destination transfers and less than half the FIFO depth for source transfers. The exceptions are at the end of a burst transaction request or at the end of a block transfer.

- **PROTCTL: Protection Control**

bits used to drive the AMBA HPROT[3:1] bus. The *AMBA Specification* recommends that the default value of HPROT indicates a non-cached, nonbuffered, privileged data access. The reset value is used to indicate such an access.

- HPROT[0] is tied high as all transfers are data accesses as there are no opcode fetches. There is a one-to-one mapping of these register bits to the HPROT[3:1] master interface signals. **SRC_PER: Source Hardware Handshaking Interface**

Assigns a hardware handshaking interface (0 - DMAH_NUM_HS_INT-1) to the source of channel *x* if the DMAC_CFGx.HS_SEL_SRC field is 0. Otherwise, this field is ignored. The channel can then communicate with the source peripheral connected to that interface via the assigned hardware handshaking interface.

For correct DMAC operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.

- **DEST_PER: Destination Hardware Handshaking Interface**

Assigns a hardware handshaking interface (0 - DMAH_NUM_HS_INT-1) to the destination of channel *x* if the DMAC_CFGx.HS_SEL_DST field is 0. Otherwise, this field is ignored. The channel can then communicate with the destination peripheral connected to that interface via the assigned hardware handshaking interface.

For correct DMA operation, only one peripheral (source or destination) should be assigned to the same handshaking interface.

25.4.8 Source Gather Register for Channel x

Name: DMAC_SGRx

Addresses:0x00800048 [0], 0x008000A0 [1]

Access: Read-write

Reset: 0x0

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

The address offset for each channel is: 0x48+[x * 0x58]

For example, SGR0: 0x048, SGR1: 0x0a0, etc.

The DMAC_CTLx.SINC field controls whether the address increments or decrements. When the DMAC_CTLx.SINC field indicates a fixed-address control, then the address remains constant throughout the transfer and the DMAC_SGRx register is ignored.

- **SGI: Source Gather Interval**

Source gather count field specifies the number of contiguous source transfers of DMAC_CTLx.SRC_TR_WIDTH between successive gather intervals. This is defined as a gather boundary.

- **SGC: Source Gather Count**

Source gather interval field (DMAC_SGRx.SGI) – specifies the source address increment/decrement in multiples of DMAC_CTLx.SRC_TR_WIDTH on a gather boundary when gather mode is enabled for the source transfer.

25.4.9 Destination Scatter Register for Channel x

Name: DMAC_DSRx

Addresses:0x00800050 [0], 0x008000A8 [1]

Access: Read-write

Reset: 0x0

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

The address offset for each channel is: 0x50+[x * 0x58]

For example, DSR0: 0x050, DSR1: 0x0a8, etc.

The DMAC_CTLx.DINC field controls whether the address increments or decrements. When the DMAC_CTLx.DINC field indicates a fixed address control then the address remains constant throughout the transfer and the DMAC_DSRx register is ignored.

- **DSI: Destination Scatter Interval**

Destination scatter interval field (DMAC_DSRx.DSI) – specifies the destination address increment/decrement in multiples of DMAC_CTLx.DST_TR_WIDTH on a scatter boundary when scatter mode is enabled for the destination transfer.

- **DSC: Destination Scatter Count**

Destination scatter count field (DMAC_DSRx.DSC) – specifies the number of contiguous destination transfers of DMAC_CTLx.DST_TR_WIDTH between successive scatter boundaries.

25.4.10 Interrupt Registers

The following sections describe the registers pertaining to interrupts, their status, and how to clear them. For each channel, there are five types of interrupt sources:

- IntTfr: DMA Transfer Complete Interrupt

This interrupt is generated on DMA transfer completion to the destination peripheral.

- IntBlock: Block Transfer Complete Interrupt

This interrupt is generated on DMA block transfer completion to the destination peripheral.

- IntSrcTran: Source Transaction Complete Interrupt

This interrupt is generated after completion of the last AMBA transfer of the requested single/burst transaction from the handshaking interface on the source side.

If the source for a channel is memory, then that channel never generates a IntSrcTran interrupt and hence the corresponding bit in this field is not set.

- IntDstTran: Destination Transaction Complete Interrupt

This interrupt is generated after completion of the last AMBA transfer of the requested single/burst transaction from the handshaking interface on the destination side.

If the destination for a channel is memory, then that channel never generates the IntDstTran interrupt and hence the corresponding bit in this field is not set.

- IntErr: Error Interrupt

This interrupt is generated when an ERROR response is received from an AHB slave on the HRESP bus during a DMA transfer. In addition, the DMA transfer is cancelled and the channel is disabled.

25.4.11 Interrupt Raw Status Registers

Name: DMAC_RawTfr, DMAC_RawBlock, DMAC_RawSrcTran, DMAC_RawDstTran, DMAC_RawErr

Address:0x008002C0

Address:0x008002C8

Address:0x008002D0

Address:0x008002D8

Address:0x008002E0

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
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	RAW1	RAW0

- **RAWx: Raw Interrupt for Each Channel**

Interrupt events are stored in these Raw Interrupt Status Registers before masking: DMAC_RawTfr, DMAC_RawBlock, DMAC_RawSrcTran, DMAC_RawDstTran, DMAC_RawErr. Each Raw Interrupt Status register has a bit allocated per channel, for example, DMAC_RawTfr[2] is Channel 2's raw transfer complete interrupt. Each bit in these registers is cleared by writing a 1 to the corresponding location in the DMAC_ClearTfr, DMAC_ClearBlock, DMAC_ClearSrcTran, DMAC_ClearDstTran, DMAC_ClearErr registers.

25.4.12 Interrupt Status Registers

Name: DMAC_StatusTfr, DMAC_StatusBlock, DMAC_StatusSrcTran, DMAC_StatusDstTran, DMAC_StatusErr

Address:0x008002E8

Address:0x008002F0

Address:0x008002F8

Address:0x00800300

Address:0x00800308

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
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	STATUS1	STATUS0

- **TSTATUSx:**

All interrupt events from all channels are stored in these Interrupt Status Registers after masking: DMAC_StatusTfr, DMAC_StatusBlock, DMAC_StatusSrcTran, DMAC_StatusDstTran, DMAC_StatusErr. Each Interrupt Status register has a bit allocated per channel, for example, DMAC_StatusTfr[2] is Channel 2's status transfer complete interrupt.The contents of these registers are used to generate the interrupt signals leaving the DMAC.

25.4.13 Interrupt Status Registers

Name: DMAC_MaskTfr, DMAC_MaskBlock, DMAC_MaskSrcTran, DMAC_MaskDstTran, DMAC_MaskErr

Address:0x00800310

Address:0x00800318

Address:0x00800320

Address:0x00800328

Address:0x00800330

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
–	–	–	–	–	–	INT_M_WE1	INT_M_WE0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	INT_MASK1	INT_MASK0

The contents of the Raw Status Registers are masked with the contents of the Mask Registers: DMAC_MaskTfr, DMAC_MaskBlock, DMAC_MaskSrcTran, DMAC_MaskDstTran, DMAC_MaskErr. Each Interrupt Mask register has a bit allocated per channel, for example, DMAC_MaskTfr[2] is the mask bit for Channel 2's transfer complete interrupt.

A channel's INT_MASK bit is only written if the corresponding mask write enable bit in the INT_MASK_WE field is asserted on the same AMBA write transfer. This allows software to set a mask bit without performing a read-modified write operation.

For example, writing hex 01x1 to the DMAC_MaskTfr register writes a 1 into DMAC_MaskTfr[0], while DMAC_MaskTfr[7:1] remains unchanged. Writing hex 00*xx* leaves DMAC_MaskTfr[7:0] unchanged.

Writing a 1 to any bit in these registers unmasks the corresponding interrupt, thus allowing the DMAC to set the appropriate bit in the Status Registers.

- **INT_MASKx: Interrupt Mask**

0 = Masked

1 = Unmasked

- **INT_M_WEx: Interrupt Mask Write Enable**

0 = Write disabled

1 = Write enabled

25.4.14 Interrupt Clear Registers

Name: DMAC_ClearTfr, DMAC_ClearBlock, DMAC_ClearSrcTran, DMAC_ClearDstTran,DMAC_ClearErr

Address:0x00800338

Address:0x00800340

Address:0x00800348

Address:0x00800350

Address:0x00800358

Access: 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
–	–	–	–	–	–	CLEAR1	CLEAR0

- **CLEARx: Interrupt Clear**

0 = No effect

1 = Clear interrupt

Each bit in the Raw Status and Status registers is cleared on the same cycle by writing a 1 to the corresponding location in the Clear registers: DMAC_ClearTfr, DMAC_ClearBlock, DMAC_ClearSrcTran, DMAC_ClearDstTran, DMAC_ClearErr. Each Interrupt Clear register has a bit allocated per channel, for example, DMAC_ClearTfr[2] is the clear bit for Channel 2's transfer complete interrupt. Writing a 0 has no effect. These registers are not readable.

25.4.15 Combined Interrupt Status Registers

Name: DMAC_StatusInt

Address:0x00800360

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
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	ERR	DSTT	SRCT	BLOCK	TFR

The contents of each of the five Status Registers (DMAC_StatusTfr, DMAC_StatusBlock, DMAC_StatusSrcTran, DMAC_StatusDstTran, DMAC_StatusErr) is OR'd to produce a single bit per interrupt type in the Combined Status Register (DMAC_StatusInt).

- **TFR**

OR of the contents of DMAC_StatusTfr Register.

- **BLOCK**

OR of the contents of DMAC_StatusBlock Register.

- **SRCT**

OR of the contents of DMAC_StatusSrcTran Register.

- **DSTT**

OR of the contents of DMAC_StatusDstTran Register.

- **ERR**

OR of the contents of DMAC_StatusErr Register.

25.4.16 Source Software Transaction Request Register

Name: DMAC_ReqSrcReg

Address:0x00800368

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
–	–	–	–	–	–	REQ_WE1	REQ_WE0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	SRC_REQ1	SRC_REQ0

A bit is assigned for each channel in this register. DMAC_ReqSrcReg[*n*] is ignored when software handshaking is not enabled for the source of channel *n*.

A channel SRC_REQ bit is written only if the corresponding channel write enable bit in the REQ_WE field is asserted on the same AMBA write transfer.

For example, writing 0x101 writes a 1 into DMAC_ReqSrcReg[0], while DMAC_ReqSrcReg[2:1] remains unchanged. Writing hex 0x0*yy* leaves DMAC_ReqSrcReg[2:0] unchanged. This allows software to set a bit in the DMAC_ReqSrcReg register without performing a read-modified write

- **SRC_REQx: Source Request**

- **REQ_WEx: Request Write Enable**

0 = Write disabled

1 = Write enabled

25.4.17 Destination Software Transaction Request Register

Name: DMAC_ReqDstReg

Address:0x00800370

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
–	–	–	–	–	–	REQ_WE1	REQ_WE0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	DST_REQ1	DST_REQ0

A bit is assigned for each channel in this register. DMAC_ReqDstReg[*n*] is ignored when software handshaking is not enabled for the source of channel *n*.

A channel DST_REQ bit is written only if the corresponding channel write enable bit in the REQ_WE field is asserted on the same AMBA write transfer.

- **DST_REQx: Destination Request**

- **REQ_WEx: Request Write Enable**

0 = Write disabled

1 = Write enabled

25.4.18 Single Source Transaction Request Register

Name: DMAC_SglReqSrcReg

Address:0x00800378

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
–	–	–	–	–	–	REQ_WE1	REQ_WE0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	S_SG_REQ1	S_SG_REQ0

A bit is assigned for each channel in this register. DMAC_SglReqSrcReg[*n*] is ignored when software handshaking is not enabled for the source of channel *n*.

A channel S_SG_REQ bit is written only if the corresponding channel write enable bit in the REQ_WE field is asserted on the same AMBA write transfer.

- **S_SG_REQx: Source Single Request**

- **REQ_WEx: Request Write Enable**

0 = Write disabled

1 = Write enabled

25.4.19 Single Destination Transaction Request Register

Name: DMAC_SglReqDstReg

Address:0x00800380

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
–	–	–	–	–	–	REQ_WE1	REQ_WE0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	D_SG_REQ1	D_SG_REQ0

A bit is assigned for each channel in this register. DMAC_SglReqDstReg[*n*] is ignored when software handshaking is not enabled for the source of channel *n*.

A channel D_SG_REQ bit is written only if the corresponding channel write enable bit in the REQ_WE field is asserted on the same AMBA write transfer.

- **D_SG_REQx: Destination Single Request**

- **REQ_WEx: Request Write Enable**

0 = Write disabled

1 = Write enabled

25.4.20 Last Source Transaction Request Register

Name: DMAC_LstSrcReqReg

Address:0x00800388

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
–	–	–	–	–	–	LSTSR_WE1	LSTSR_WE0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	LSTSRC1	LSTSRC0

A bit is assigned for each channel in this register. LstSrcReqReg[*n*] is ignored when software handshaking is not enabled for the source of channel *n*.

A channel LSTSRC bit is written only if the corresponding channel write enable bit in the LSTSR_WE field is asserted on the same AMBA write transfer.

- **LSTSRCx: Source Last Transaction Request**

- **LSTSR_WEx: Source Last Transaction Request Write Enable**

0 = Write disabled

1 = Write enabled

25.4.21 Last Destination Transaction Request Register

Name: DMAC_LstDstReqReg

Address:0x00800390

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
–	–	–	–	–	–	LSTDS_WE1	LSTDS_WE0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	LSTDST1	LSTDST0

A bit is assigned for each channel in this register. LstDstReqReg[*n*] is ignored when software handshaking is not enabled for the source of channel *n*.

A channel LSTDST bit is written only if the corresponding channel write enable bit in the LSTDS_WE field is asserted on the same AMBA write transfer.

- **LSTDSTx: Destination Last Transaction Request**

- **LSTDS_WEx: Destination Last Transaction Request Write Enable**

0 = Write disabled

1 = Write enabled

25.4.22 DMAC Configuration Register

Name: DMAC_DmaCfgReg

Address:0x00800398

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
–	–	–	–	–	–	–	DMA_EN

- **DMA_EN: DMA Controller Enable**

0 = DMAC Disabled

1 = DMAC Enabled.

This register is used to enable the DMAC, which must be done before any channel activity can begin.

If the global channel enable bit is cleared while any channel is still active, then DMAC_DmaCfgReg.DMA_EN still returns ‘1’ to indicate that there are channels still active until hardware has terminated all activity on all channels, at which point the DMAC_DmaCfgReg.DMA_EN bit returns ‘0’.

25.4.23 DMAC Channel Enable Register

Name: DMAC_ChEnReg

Address:0x008003A0

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
–	–	–	–	–	–	CH_EN_WE1	CH_EN_WE0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	CH_EN1	CH_EN0

- **CH_ENx:**

0 = Disable the Channel

1 = Enable the Channel

Enables/Disables the channel. Setting this bit enables a channel, clearing this bit disables the channel.

The DMAC_ChEnReg.CH_EN bit is automatically cleared by hardware to disable the channel after the last AMBA transfer of the DMA transfer to the destination has completed.Software can therefore poll this bit to determine when a DMA transfer has completed.

- **CH_EN_WEx:**

The channel enable bit, CH_EN, is only written if the corresponding channel write enable bit, CH_EN_WE, is asserted on the same AMBA write transfer.

For example, writing 0x101 writes a 1 into DMAC_ChEnReg[0], while DMAC_ChEnReg[7:1] remains unchanged.

26. Peripheral DMA Controller (PDC)

26.1 Overview

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 20 channels. The full-duplex peripherals feature 18 mono directional channels used in pairs (transmit only or receive only). The half-duplex peripherals feature 2 bi-directional channels.

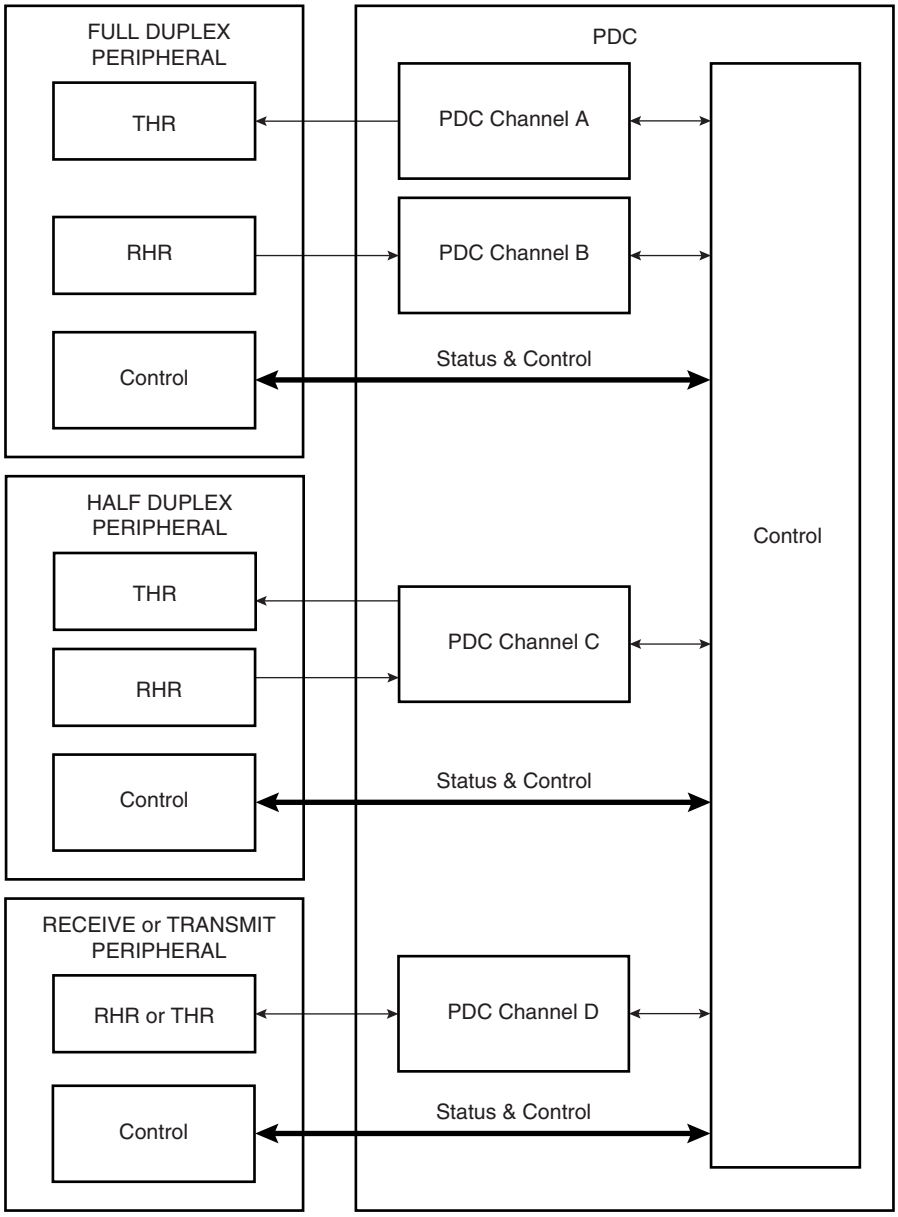
The user interface of each PDC channel is integrated into the user interface of the peripheral it serves. The user interface of mono directional channels (receive only or transmit only), contains two 32-bit memory pointers and two 16-bit counters, one set (pointer, counter) for current transfer and one 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.

26.2 Block Diagram

Figure 26-1. Block Diagram



26.3 Functional Description

26.3.1 Configuration

The PDC channel user interface enables the user to configure and control data transfers for each channel. The user interface of each PDC channel is integrated into the associated peripheral user interface.

The user interface of a serial peripheral, whether it is full or half duplex, contains four 32-bit pointers (RPR, RNPR, TPR, TNPR) and four 16-bit counter registers (RCR, RNCR, TCR, TNCR). However, the transmit and receive parts of each type are programmed differently: the

transmit and receive parts of a full duplex peripheral can be programmed at the same time, whereas only one part (transmit or receive) of a half duplex peripheral can be programmed at a time.

32-bit pointers define the access location in memory for current and next transfer, whether it is for read (transmit) or write (receive). 16-bit counters define the size of current and next transfers. It is possible, at any moment, to read the number of transfers left for each channel.

The PDC has dedicated status registers which indicate if the transfer is enabled or disabled for each channel. The status for each channel is located in the associated peripheral status register. Transfers can be enabled and/or disabled by setting TXTEN/TXTDIS and RXTEN/RXTDIS in the peripheral's Transfer Control Register.

At the end of a transfer, the PDC channel sends status flags to its associated peripheral. These flags are visible in the peripheral status register (ENDRX, ENDTX, RXBUFF, and TXBUFE). Refer to Section 26.3.3 and to the associated peripheral user interface.

26.3.2 Memory Pointers

Each full duplex peripheral is connected to the PDC by a receive channel and a transmit channel. Both channels have 32-bit memory pointers that point respectively to a receive area and to a transmit area in on- and/or off-chip memory.

Each half duplex peripheral is connected to the PDC by a bidirectional channel. This channel has two 32-bit memory pointers, one for current transfer and the other for next transfer. These pointers point to transmit or receive data depending on the operating mode of the peripheral.

Depending on the type of transfer (byte, half-word or word), the memory pointer is incremented respectively by 1, 2 or 4 bytes.

If a memory pointer address changes in the middle of a transfer, the PDC channel continues operating using the new address.

26.3.3 Transfer Counters

Each channel has two 16-bit counters, one for current transfer and the other one for next transfer. These counters define the size of data to be transferred by the channel. The current transfer counter is decremented first as the data addressed by current memory pointer starts to be transferred. When the current transfer counter reaches zero, the channel checks its next transfer counter. If the value of next counter is zero, the channel stops transferring data and sets the appropriate flag. But if the next counter value is greater 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.

26.3.4 Data Transfers

The serial peripheral triggers its associated PDC channels' transfers using transmit enable (TXEN) and receive enable (RXEN) flags in the transfer control register integrated in the peripheral's user interface.

When the peripheral receives an external data, it sends a Receive Ready signal to its PDC receive channel which then requests access to the Matrix. When access is granted, the PDC receive channel starts reading the peripheral Receive Holding Register (RHR). The read data are stored in an internal buffer and then written to memory.

When the peripheral is about to send data, it sends a Transmit Ready to its PDC transmit channel which then requests access to the Matrix. When access is granted, the PDC transmit channel reads data from memory and puts them to Transmit Holding Register (THR) of its associated peripheral. The same peripheral sends data according to its mechanism.

26.3.5 PDC Flags and Peripheral Status Register

Each peripheral connected to the PDC sends out receive ready and transmit ready flags and the PDC sends back flags to the peripheral. All these flags are only visible in the Peripheral Status Register.

Depending on the type of peripheral, half or full duplex, the flags belong to either one single channel or two different channels.

26.3.5.1 Receive Transfer End

This flag is set when PERIPH_RCR register reaches zero and the last data has been transferred to memory.

It is reset by writing a non zero value in PERIPH_RCR or PERIPH_RNCR.

26.3.5.2 Transmit Transfer End

This flag is set when PERIPH_TCR register reaches zero and the last data has been written into peripheral THR.

It is reset by writing a non zero value in PERIPH_TCR or PERIPH_TNCR.

26.3.5.3 Receive Buffer Full

This flag is set when PERIPH_RCR register reaches zero with PERIPH_RNCR also set to zero and the last data has been transferred to memory.

It is reset by writing a non zero value in PERIPH_TCR or PERIPH_TNCR.

26.3.5.4 Transmit Buffer Empty

This flag is set when PERIPH_TCR register reaches zero with PERIPH_TNCR also set to zero and the last data has been written into peripheral THR.

It is reset by writing a non zero value in PERIPH_TCR or PERIPH_TNCR.

26.4 Peripheral DMA Controller (PDC) User Interface

Table 26-1. 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.)

26.4.1 Receive Pointer Register

Register Name: PERIPH_RPR

Access Type: Read-write

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

- **RXPTR: Receive Pointer Register**

RXPTR must be set to receive buffer address.

When a half duplex peripheral is connected to the PDC, RXPTR = TXPTR.

26.4.2 Receive Counter Register

Register Name: PERIPH_RCR

Access Type: Read-write

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

- **RXCTR: Receive Counter Register**

RXCTR must be set to receive buffer size.

When a half duplex peripheral is connected to the PDC, RXCTR = TXCTR.

0 = Stops peripheral data transfer to the receiver

1 - 65535 = Starts peripheral data transfer if corresponding channel is active

26.4.3 Transmit Pointer Register

Register Name: PERIPH_TPR

Access Type: Read-write

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

- **TXPTR: Transmit Counter Register**

TXPTR must be set to transmit buffer address.

When a half duplex peripheral is connected to the PDC, RXPTR = TXPTR.

26.4.4 Transmit Counter Register

Register Name: PERIPH_TCR

Access Type: Read-write

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

- **TXCTR: Transmit Counter Register**

TXCTR must be set to transmit buffer size.

When a half duplex peripheral is connected to the PDC, RXCTR = TXCTR.

0 = Stops peripheral data transfer to the transmitter

1- 65535 = Starts peripheral data transfer if corresponding channel is active

26.4.5 Receive Next Pointer Register

Register Name: PERIPH_RNPR

Access Type: Read-write

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

- **RXNPTR: Receive Next Pointer**

RXNPTR contains next receive buffer address.

When a half duplex peripheral is connected to the PDC, RXNPTR = TXNPTR.

26.4.6 Receive Next Counter Register

Register Name: PERIPH_RNCR

Access Type: Read-write

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

- **RXNCTR: Receive Next Counter**

RXNCTR contains next receive buffer size.

When a half duplex peripheral is connected to the PDC, RXNCTR = TXNCTR.

26.4.7 Transmit Next Pointer Register

Register Name: PERIPH_TNPR

Access Type: Read-write

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

- **TXNPTR: Transmit Next Pointer**

TXNPTR contains next transmit buffer address.

When a half duplex peripheral is connected to the PDC, RXNPTR = TXNPTR.

26.4.8 Transmit Next Counter Register

Register Name: PERIPH_TNCR

Access Type: Read-write

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

- **TXNCTR: Transmit Counter Next**

TXNCTR contains next transmit buffer size.

When a half duplex peripheral is connected to the PDC, RXNCTR = TXNCTR.

26.4.9 Transfer Control Register

Register Name: PERIPH_PTCR

Access Type: Write-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.

26.4.10 Transfer Status Register

Register Name: PERIPH_PTSR

Access Type: Read-only

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

- **RXTEN: Receiver Transfer Enable**

0 = PDC Receiver channel requests are disabled.

1 = PDC Receiver channel requests are enabled.

- **TXTEN: Transmitter Transfer Enable**

0 = PDC Transmitter channel requests are disabled.

1 = PDC Transmitter channel requests are enabled.

27. Clock Generator

27.1 Overview

The Clock Generator is made up of 2 PLLs, a Main 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

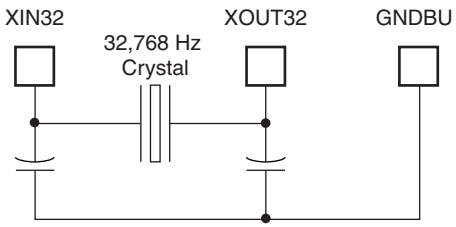
The Clock Generator User Interface is embedded within the Power Management Controller one and is described in Section 28.9. However, the Clock Generator registers are named CKGR_.

- PLLACK is the output of the Divider and PLL A block.
- PLLBCK is the output of the Divider and PLL B block.

27.2 Slow Clock Crystal Oscillator

The Clock Generator integrates a 32,768 Hz low-power oscillator. The XIN32 and XOUT32 pins must be connected to a 32,768 Hz crystal. Two external capacitors must be wired as shown in Figure 27-1.

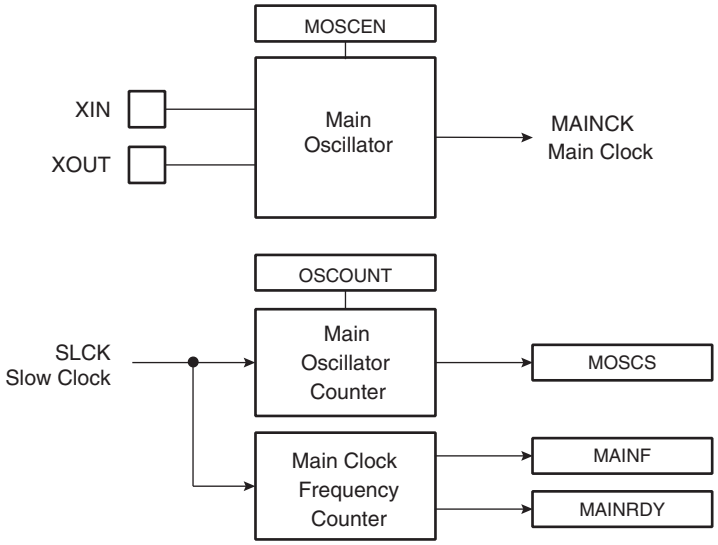
Figure 27-1. Typical Slow Clock Crystal Oscillator Connection



27.3 Main Oscillator

Figure 27-2 shows the Main Oscillator block diagram.

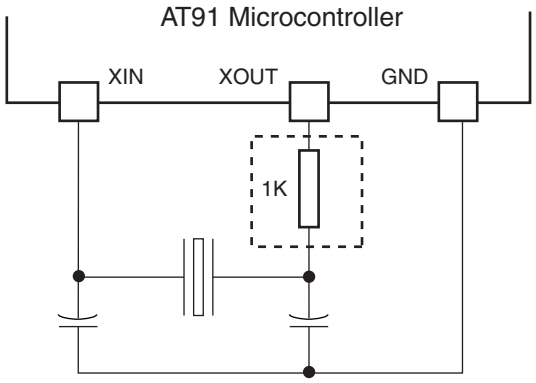
Figure 27-2. Main Oscillator Block Diagram



27.3.1 Main Oscillator Connections

The Clock Generator integrates a Main Oscillator that is designed for a 3 to 20 MHz fundamental crystal. The typical crystal connection is illustrated in Figure 27-3. The 1 kΩ resistor is only required for crystals with frequencies lower than 8 MHz. For further details on the electrical characteristics of the Main Oscillator, see the section “DC Characteristics” of the product datasheet.

Figure 27-3. Typical Crystal Connection



27.3.2 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.

27.3.3 Main Oscillator Control

To minimize the power required to start up the system, the main oscillator is disabled after reset and slow clock is selected.

The software enables or disables the main oscillator so as to reduce power consumption by clearing the MOSCEN bit in the Main Oscillator Register (CKGR_MOR).

When disabling the main oscillator by clearing the MOSCEN bit in CKGR_MOR, the MOSCS bit in PMC_SR is automatically cleared, indicating the main clock is off.

When enabling the main oscillator, the user must initiate the main oscillator counter with a value corresponding to the startup time of the oscillator. This startup time depends on the crystal frequency connected to the main oscillator.

When the MOSCEN bit and the OSCOUNT are written in CKGR_MOR to enable the main oscillator, the MOSCS bit in PMC_SR (Status Register) is cleared and the counter starts counting down on the slow clock divided by 8 from the OSCOUNT value. Since the OSCOUNT value is coded with 8 bits, the maximum startup time is about 62 ms.

When the counter reaches 0, the MOSCS bit is set, indicating that the main clock is valid. Setting the MOSCS bit in PMC_IMR can trigger an interrupt to the processor.

27.3.4 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.

27.3.5 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.

27.4 Divider and PLL Block

The PLL embeds an input divider to increase the accuracy of the resulting clock signals. However, the user must respect the PLL minimum input frequency when programming the divider.

Figure 27-4 shows the block diagram of the divider and PLL block.

Figure 27-4. Divider and PLL Block Diagram

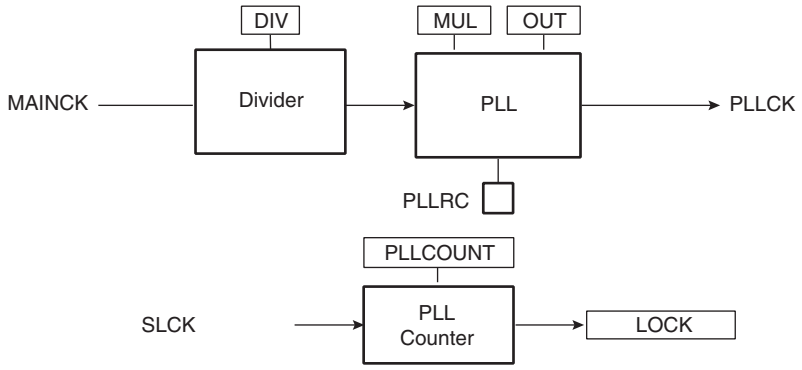
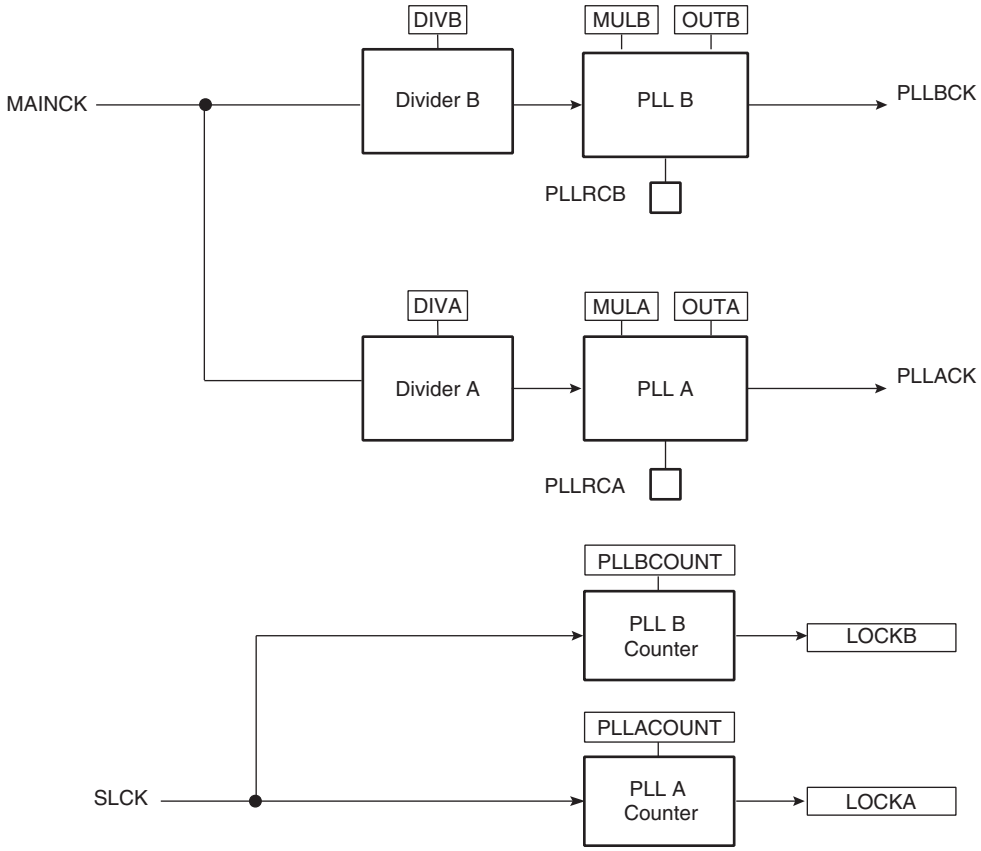


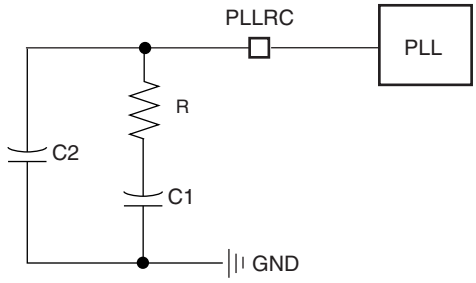
Figure 27-5. Divider and PLL Block Diagram



27.4.1 PLL Filter

The PLL requires connection to an external second-order filter through the PLLRCA and/or PLLRCB pin. Figure 27-6 shows a schematic of these filters.

Figure 27-6. PLL Capacitors and Resistors



Values of R, C1 and C2 to be connected to the PLLRC pin must be calculated as a function of the PLL input frequency, the PLL output frequency and the phase margin. A trade-off has to be found between output signal overshoot and startup time.

27.4.2 Divider and Phase Lock Loop Programming

The divider can be set between 1 and 255 in steps of 1. When a divider field (DIV) is set to 0, the output of the corresponding divider and the PLL output is a continuous signal at level 0. On reset, each DIV field is set to 0, thus the corresponding PLL input clock is set to 0.

The PLL allows multiplication of the divider's outputs. The PLL clock signal has a frequency that depends on the respective source signal frequency and on the parameters DIV and MUL. The factor applied to the source signal frequency is (MUL + 1)/DIV. When MUL is written to 0, the corresponding PLL is disabled and its power consumption is saved. Re-enabling the PLL can be performed by writing a value higher than 0 in the MUL field.

Whenever the PLL is re-enabled or one of its parameters is changed, the LOCK bit (LOCKA or LOCKB) in PMC_SR is automatically cleared. The values written in the PLLCOUNT field (PLLACOUNT or PLLBCOUNT) in CKGR_PLLR (CKGR_PLLAR or CKGR_PLLBR), are loaded in the PLL counter. The PLL counter then decrements at the speed of the Slow Clock until it reaches 0. At this time, the LOCK bit is set in PMC_SR and can trigger an interrupt to the processor. The user has to load the number of Slow Clock cycles required to cover the PLL transient time into the PLLCOUNT field. The transient time depends on the PLL filter. The initial state of the PLL and its target frequency can be calculated using a specific tool provided by Atmel.

During the PLLA or PLLB initialization, the PMC_PLLICPR register must be programmed correctly.

28. Power Management Controller (PMC)

28.1 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), must be switched off when entering processor in Idle Mode.
- Peripheral Clocks, typically MCK, provided to the embedded peripherals (USART, SSC, SPI, TWI, TC, MCI, etc.) and independently controllable. In order to reduce the number of clock names in a product, the Peripheral Clocks are named MCK in the product datasheet.
- UHP Clock (UHPCK), required by USB Host Port operations.
- UDP Clock (UDPCK), required by USB Device Port operations.
- Programmable Clock Outputs can be selected from the clocks provided by the clock generator and driven on the PCKx pins.

28.2 Master Clock Controller

The Master Clock Controller provides selection and division of the Master Clock (MCK). MCK is the clock provided to all the peripherals and the memory controller.

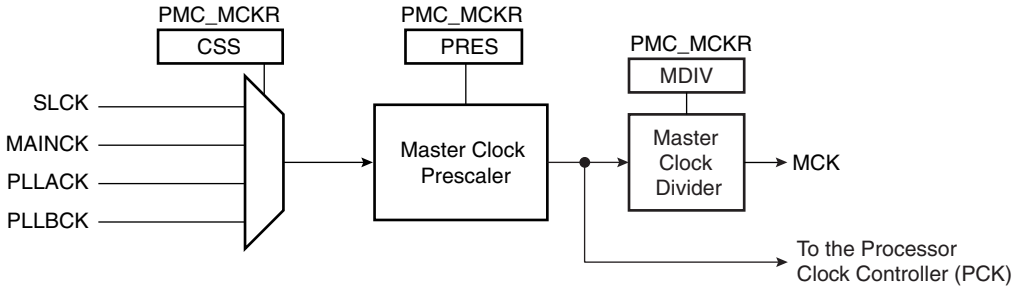
The Master Clock is selected from one of the clocks provided by the Clock Generator. Selecting the Slow Clock provides a Slow Clock signal to the whole device. Selecting the Main Clock saves power consumption of the PLLs.

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.

Each time PMC_MCKR is written to define a new Master Clock, the MCKRDY bit is cleared in PMC_SR. It reads 0 until the Master Clock is established. Then, the MCKRDY bit is set and can trigger an interrupt to the processor. This feature is useful when switching from a high-speed clock to a lower one to inform the software when the change is actually done.

Figure 28-1. Master Clock Controller



28.3 Processor Clock Controller

The PMC features a Processor Clock Controller (PCK) that implements the Processor Idle Mode. The Processor Clock can be disabled by writing the System Clock Disable Register (PMC_SCDR). The status of this clock (at least for debug purposes) can be read in the System Clock Status Register (PMC_SCSR).

The Processor Clock PCK is enabled after a reset and is automatically re-enabled by any enabled interrupt. The Processor Idle Mode is achieved by disabling the Processor Clock which is automatically re-enabled by any enabled fast or normal interrupt, or by the reset of the product. and entering 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 entered with CP15 coprocessor operation. Refer to the Atmel application note, Optimizing Power Consumption of AT91SAM9261-based Systems, lit. number 6217.

When the Processor Clock is disabled, the current instruction is finished before the clock is stopped, but this does not prevent data transfers from other masters of the system bus.

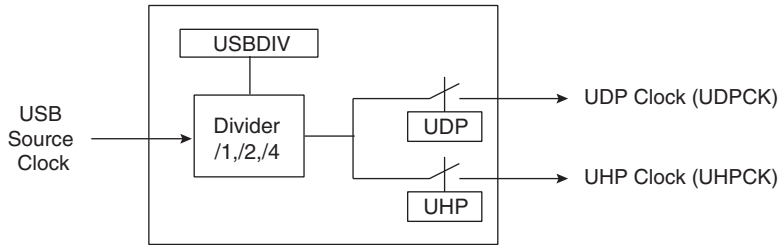
28.4 USB Clock Controller

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 28-2).

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.
- The USB device clock can be enabled by setting the UDP bit in PMC_SCER. To save power on this peripheral when it is not used, the user can set the UDP bit in PMC_SCDR. The UDP bit in PMC_SCSR gives the activity of this clock. The USB device port require both the 48 MHz signal and the Master Clock. The Master Clock may be controlled via the Master Clock Controller.

Figure 28-2. USB Clock Controller



28.5 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.

28.6 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.

28.7 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. ICPPLLA in PMC_PLLICPR register must be set to 1 before configuring 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. ICPPLLB in PMC_PLLICPR register must be set to 1 before configuring 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 0 which means that master clock is equal to slow clock.

The MDIV field is used to control the Master Clock divider. 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 28.8.2. “Clock Switching Waveforms” on page 356.

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, PLLCK, PLLACK, PLLBCK.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 0 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.

28.8 Clock Switching Details

28.8.1 Master Clock Switching Timings

Table 28-1 and Table 28-2 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 28-1. 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 28-2. 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

28.8.2 Clock Switching Waveforms

Figure 28-3. Switch Master Clock from Slow Clock to PLL Clock

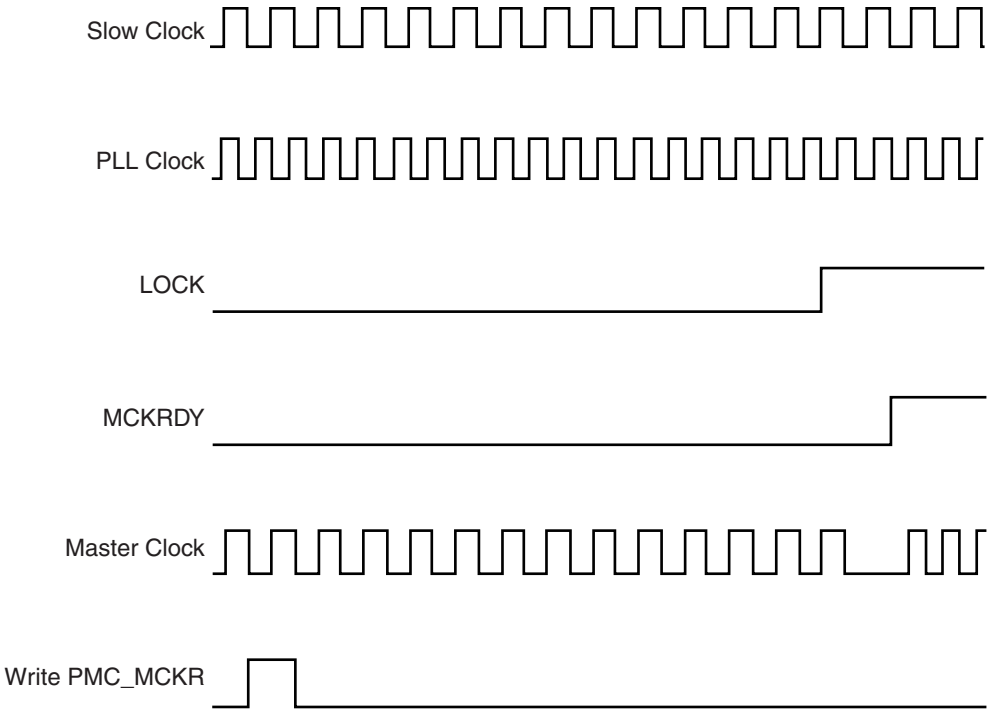


Figure 28-4. Switch Master Clock from Main Clock to Slow Clock

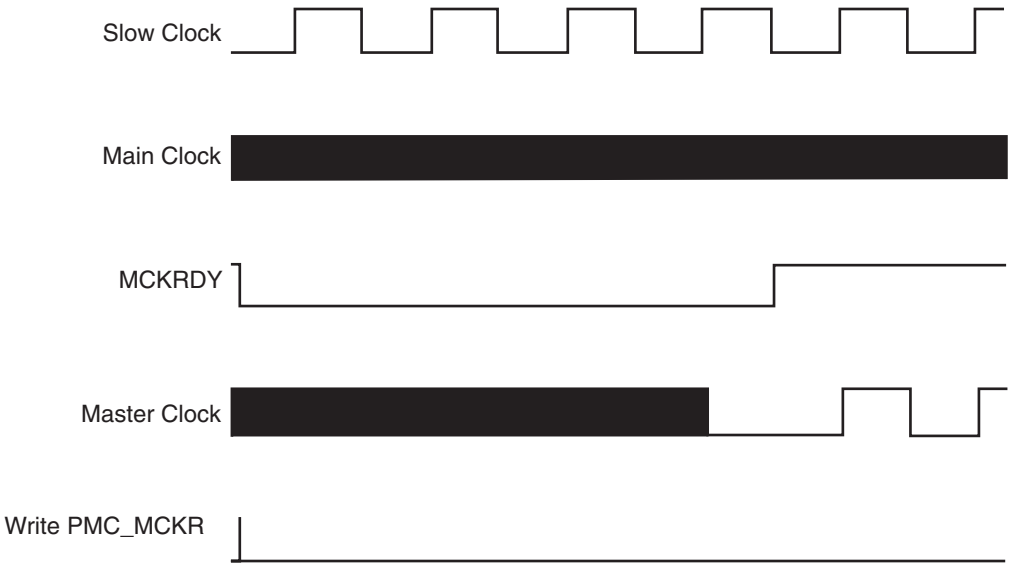


Figure 28-5. Change PLLA Programming

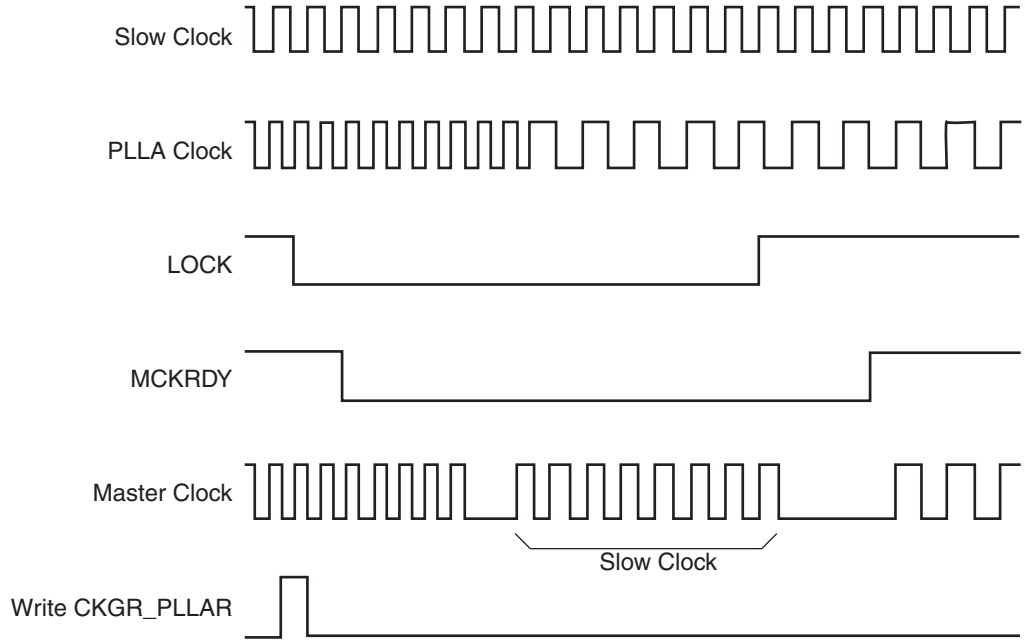


Figure 28-6. Change PLLB Programming

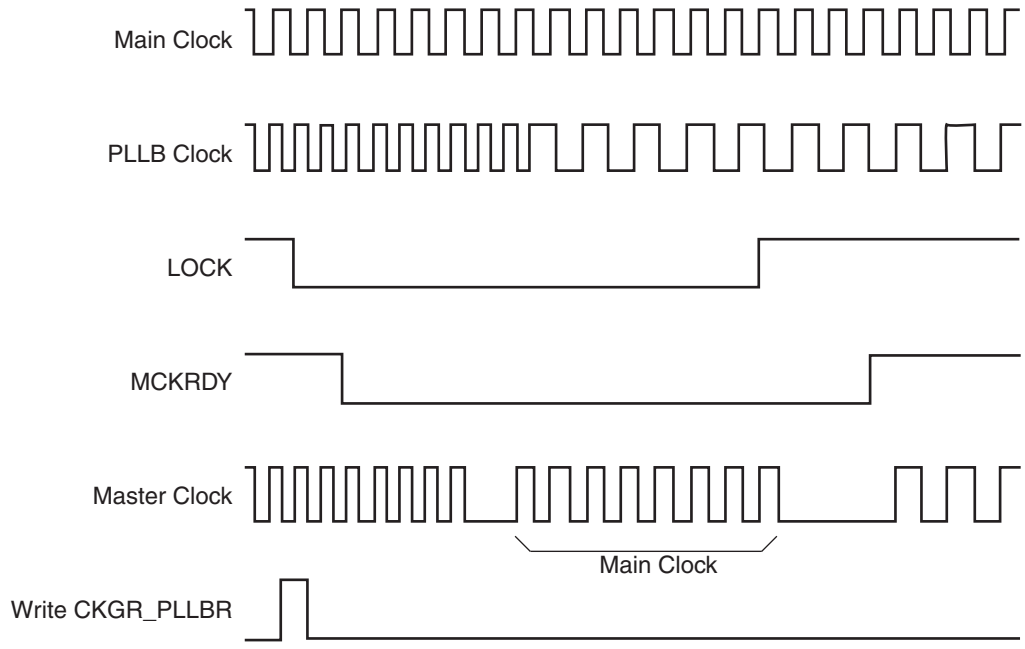
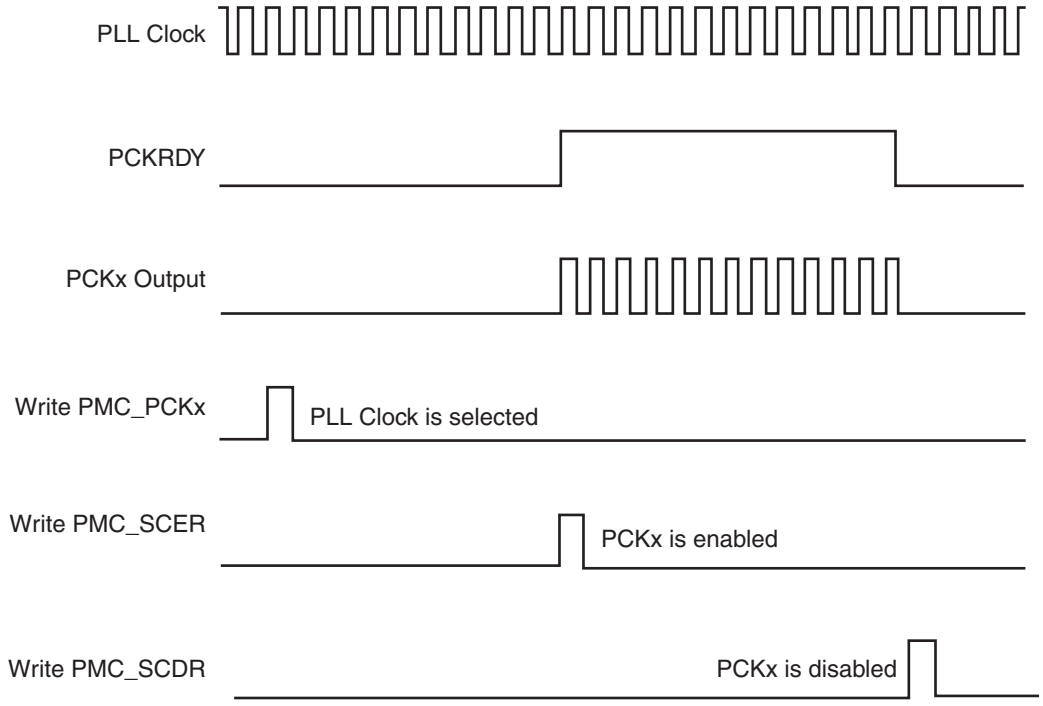


Figure 28-7. Programmable Clock Output Programming



28.9 Power Management Controller (PMC) User Interface

Table 28-3. Register Mapping

Offset	Register	Name	Access	Reset Value
0x0000	System Clock Enable Register	PMC_SCER	Write-only	–
0x0004	System Clock Disable Register	PMC_SCDR	Write-only	–
0x0008	System Clock Status Register	PMC _SCSR	Read-only	0x03 0x01
0x000C	Reserved	–	–	–
0x0010	Peripheral Clock Enable Register	PMC _PCER	Write-only	–
0x0014	Peripheral Clock Disable Register	PMC_PCDR	Write-only	–
0x0018	Peripheral Clock Status Register	PMC_PCSR	Read-only	0x0
0x001C	Reserved	–	–	–
0x0020	Main Oscillator Register	CKGR_MOR	Read-write	0x0
0x0024	Main Clock Frequency Register	CKGR_MCFR	Read-only	0x0
0x0028	PLL A Register	CKGR_PLLAR	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	–	–	–
0x0080	Charge Pump Current Register	PMC_PLLICPR	Write-only	--
0x0084 - 0x00FC	Reserved	–	–	–

28.9.1 PMC System Clock Enable Register

Register Name:PMC_SCER

Address: 0xFFFFFC00

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	PCK3	PCK2	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	UHP	–	–	–	–	–	–

- **UHP: USB Host Port Clock Enable**

0 = No effect.

1 = Enables the 12 and 48 MHz clock of the USB Host Port.

- **UDP: USB Device Port Clock Enable**

0 = No effect.

1 = Enables the 48 MHz clock of the USB Device Port.

- **PCKx: Programmable Clock x Output Enable**

0 = No effect.

1 = Enables the corresponding Programmable Clock output.

28.9.2 PMC System Clock Disable Register

Register Name:PMC_SCDR

Address: 0xFFFFFC04

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	PCK3	PCK2	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	UHP	–	–	–	–	–	PCK

- **PCK: Processor Clock Disable**

0 = No effect.

1 = Disables the Processor clock. This is used to enter the processor in Idle Mode.

- **UHP: USB Host Port Clock Disable**

0 = No effect.

1 = Disables the 12 and 48 MHz clock of the USB Host Port.

- **UDP: USB Device Port Clock Disable**

0 = No effect.

1 = Disables the 48 MHz clock of the USB Device Port.

- **PCKx: Programmable Clock x Output Disable**

0 = No effect.

1 = Disables the corresponding Programmable Clock output.

28.9.3 PMC System Clock Status Register

Register Name:PMC_SCSR

Address: 0xFFFFFC08

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	PCK3	PCK2	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	UHP	–	–	–	–	–	PCK

- **PCK: Processor Clock Status**

0 = The Processor clock is disabled.

1 = The Processor clock is enabled.

- **UHP: USB Host Port Clock Status**

0 = The 12 and 48 MHz clock (UHPCK) of the USB Host Port is disabled.

1 = The 12 and 48 MHz clock (UHPCK) of the USB Host Port is enabled.

- **UDP: USB Device Port Clock Status**

0 = The 48 MHz clock (UDPCK) of the USB Device Port is disabled.

1 = The 48 MHz clock (UDPCK) of the USB Device Port is enabled.

- **PCKx: Programmable Clock x Output Status**

0 = The corresponding Programmable Clock output is disabled.

1 = The corresponding Programmable Clock output is enabled.

28.9.4 PMC Peripheral Clock Enable Register

Register Name:PMC_PCER

Address: 0xFFFFFC10

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	-	-

- **PIDx: Peripheral Clock x Enable**

0 = No effect.

1 = Enables the corresponding peripheral clock.

Note: PID2 to PID31 refer to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet.

Note: Programming the control bits of the Peripheral ID that are not implemented has no effect on the behavior of the PMC.

28.9.5 PMC Peripheral Clock Disable Register

Register Name:PMC_PCDR

Address: 0xFFFFFC14

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	-	-

- **PIDx: Peripheral Clock x Disable**

0 = No effect.

1 = Disables the corresponding peripheral clock.

Note: PID2 to PID31 refer to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet.

28.9.6 PMC Peripheral Clock Status Register

Register Name:PMC_PCSR

Address: 0xFFFFFC18

Access Type: Read-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	–	–

- **PIDx: Peripheral Clock x Status**

0 = The corresponding peripheral clock is disabled.

1 = The corresponding peripheral clock is enabled.

Note: PID2 to PID31 refer to identifiers as defined in the section “Peripheral Identifiers” in the product datasheet.

28.9.7 PMC Clock Generator Main Oscillator Register

Register Name:CKGR_MOR

Address: 0xFFFFFC20

Access Type: Read-write

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

- **MOSCEN: Main Oscillator Enable**

A crystal must be connected between XIN and XOUT.

0 = The Main Oscillator is disabled.

1 = The Main Oscillator is enabled. OSCBYPASS must be set to 0.

When MOSCEN is set, the MOSCS flag is set once the Main Oscillator startup time is achieved.

- **OSCBYPASS: Oscillator Bypass**

0 = No effect.

1 = The Main Oscillator is bypassed. MOSCEN must be set to 0. An external clock must be connected on XIN.

When OSCBYPASS is set, the MOSCS flag in PMC_SR is automatically set.

Clearing MOSCEN and OSCBYPASS bits allows resetting the MOSCS flag.

- **OSCOUNT: Main Oscillator Start-up Time**

Specifies the number of Slow Clock cycles multiplied by 8 for the Main Oscillator start-up time.

28.9.8 PMC Clock Generator Main Clock Frequency Register

Register Name:CKGR_MCFR

Address: 0xFFFFFC24

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
MAINF							

7	6	5	4	3	2	1	0
MAINF							

- **MAINF: Main Clock Frequency**

Gives the number of Main Clock cycles within 16 Slow Clock periods.

- **MAINRDY: Main Clock Ready**

0 = MAINF value is not valid or the Main Oscillator is disabled.

1 = The Main Oscillator has been enabled previously and MAINF value is available.

28.9.9 PMC Clock Generator PLL A Register

Register Name:CKGR_PLLAR

Address: 0xFFFFFC28

Access Type: Read-write

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

23	22	21	20	19	18	17	16
MULA							

15	14	13	12	11	10	9	8
OUTA		PLLACOUNT					

7	6	5	4	3	2	1	0
DIVA							

Possible limitations on PLL A input frequencies and multiplier factors should be checked before using the PMC.

Warning: Bit 29 must always be set to 1 when programming the CKGR_PLLAR register.

- **DIVA: Divider A**

DIVA	Divider Selected
0	Divider output is 0
1	Divider is bypassed
2 - 255	Divider output is the Main Clock divided by DIVA.

- **PLLACOUNT: PLL A Counter**

Specifies the number of Slow Clock cycles before the LOCKA bit is set in PMC_SR after CKGR_PLLAR is written.

- **OUTA: PLL A Clock Frequency Range**

To optimize clock performance, this field must be programmed as specified in “PLL Characteristics” in the Electrical Characteristics section of the product datasheet.

- **MULA: PLL A Multiplier**

0 = The PLL A is deactivated.

1 up to 2047 = The PLL A Clock frequency is the PLL A input frequency multiplied by MULA + 1.

28.9.10 PMC Clock Generator PLL B Register

Register Name:CKGR_PLLBR

Address: 0xFFFFFC2C

Access Type: Read-write

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

23	22	21	20	19	18	17	16
MULB							

15	14	13	12	11	10	9	8
OUTB		PLLBCOUNT					

7	6	5	4	3	2	1	0
DIVB							

Possible limitations on PLLB 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: PLL B Clock Frequency Range**

To optimize clock performance, this field must be programmed as specified in “PLL Characteristics” in the Electrical Characteristics section of the product datasheet.

- **MULB: PLL B Multiplier**

0 = The PLL B is deactivated.

1 up to 2047 = The PLL B Clock frequency is the PLL B input frequency multiplied by MULB + 1.

- **USBDIV: Divider for USB Clock**

USBDIV		Divider for USB Clock(s)
0	0	Divider output is PLLB clock output.
0	1	Divider output is PLLB clock output divided by 2.
1	0	Divider output is PLLB clock output divided by 4.
1	1	Reserved.

28.9.11 PMC Master Clock Register

Register Name:PMC_MCKR

Address: 0xFFFFFC30

Access Type: Read-write

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

- **CSS: Master Clock Selection**

CSS		Clock Source Selection
0	0	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

- **MDIV: Master Clock Division**

MDIV		Master Clock Division
0	0	Master Clock is Processor Clock.
0	1	Master Clock is Processor Clock divided by 2.
1	0	Master Clock is Processor Clock divided by 4.
1	1	Reserved.

28.9.12 PMC Programmable Clock Register

Register Name: PMC_PCKx

Address: 0xFFFFFC40

Access Type: Read-write

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

- **CSS: Master Clock Selection**

CSS		Clock Source Selection
0	0	Slow Clock is selected.
0	1	Main Clock is selected.
1	0	PLL A Clock is selected.
1	1	PLL B Clock is selected.

- **PRES: Programmable Clock Prescaler**

PRES			Programmable Clock
0	0	0	Selected clock
0	0	1	Selected clock divided by 2
0	1	0	Selected clock divided by 4
0	1	1	Selected clock divided by 8
1	0	0	Selected clock divided by 16
1	0	1	Selected clock divided by 32
1	1	0	Selected clock divided by 64
1	1	1	Reserved

28.9.13 PMC Interrupt Enable Register

Register Name:PMC_IER

Address: 0xFFFFFC60

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	PCKRDY3	PCKRDY2	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: Main Oscillator Status Interrupt Enable**
- **LOCKA: PLL A Lock Interrupt Enable**
- **LOCKB: PLL B Lock Interrupt Enable**
- **MCKRDY: Master Clock Ready Interrupt Enable**
- **PCKRDYx: Programmable Clock Ready x Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

28.9.14 PMC Interrupt Disable Register

Register Name:PMC_IDR

Address: 0xFFFFFC64

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	PCKRDY3	PCKRDY2	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: Main Oscillator Status Interrupt Enable**
- **LOCKA: PLL A Lock Interrupt Enable**
- **LOCKB: PLL B Lock Interrupt Enable**
- **MCKRDY: Master Clock Ready Interrupt Disable**
- **PCKRDYx: Programmable Clock Ready x Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

28.9.15 PMC Status Register

Register Name:PMC_SR

Address: 0xFFFFFC68

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	PCKRDY3	PCKRDY2	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: 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.

- **PCKRDYx: Programmable Clock Ready Status**

0 = Programmable Clock x is not ready.

1 = Programmable Clock x is ready.

28.9.16 PMC Interrupt Mask Register

Register Name:PMC_IMR

Address: 0xFFFFFC6C

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	PCKRDY3	PCKRDY2	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCKB	LOCKA	MOSCS

- **MOSCS: Main Oscillator Status Interrupt Mask**
- **LOCKA: PLL A Lock Interrupt Disable**
- **LOCKB: PLL B Lock Interrupt Disable**
- **MCKRDY: Master Clock Ready Interrupt Mask**
- **PCKRDYx: Programmable Clock Ready x Interrupt Mask**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

28.9.17 PLL Charge Pump Current Register

Register Name:PMC_PLLICPR

Address: 0xFFFFFC80

Access Type: Write-only

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

- **ICPPLLA: Charge pump current**

Must be set to 1.

- **ICPPLLB: Charge pump current**

Must be set to 1.

29. Advanced Interrupt Controller (AIC)

29.1 Overview

The Advanced Interrupt Controller (AIC) is an 8-level priority, individually maskable, vectored interrupt controller, providing handling of up to thirty-two interrupt sources. It is designed to substantially reduce the software and real-time overhead in handling internal and external interrupts.

The AIC drives the nFIQ (fast interrupt request) and the nIRQ (standard interrupt request) inputs of an ARM processor. Inputs of the AIC are either internal peripheral interrupts or external interrupts coming from the product's pins.

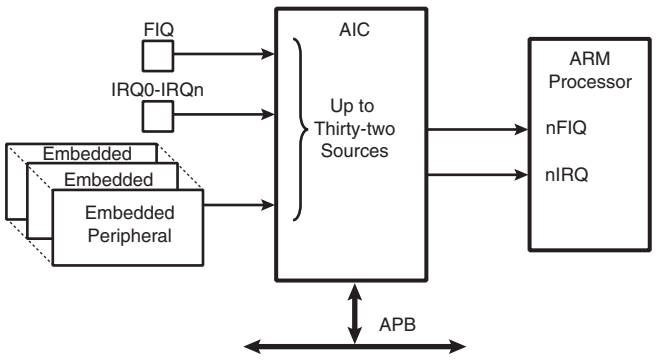
The 8-level Priority Controller allows the user to define the priority for each interrupt source, thus permitting higher priority interrupts to be serviced even if a lower priority interrupt is being treated.

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

The fast forcing feature redirects any internal or external interrupt source to provide a fast interrupt rather than a normal interrupt.

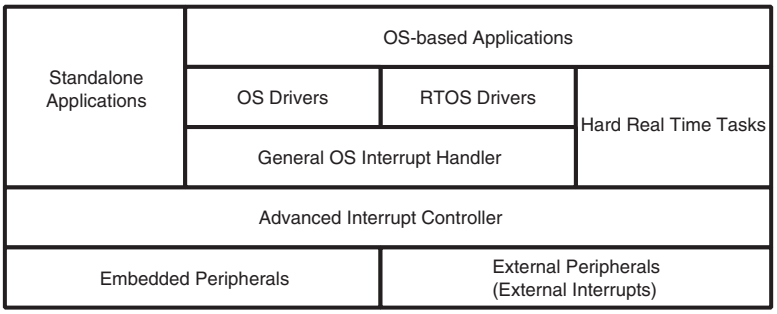
29.2 Block Diagram

Figure 29-1. Block Diagram



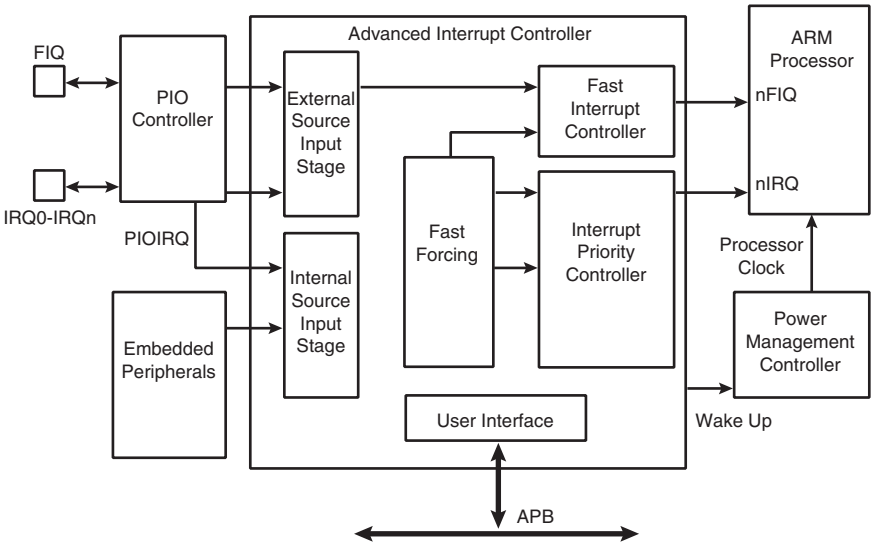
29.3 Application Block Diagram

Figure 29-2. Description of the Application Block



29.4 AIC Detailed Block Diagram

Figure 29-3. AIC Detailed Block Diagram



29.5 I/O Line Description

Table 29-1. I/O Line Description

Pin Name	Pin Description	Type
FIQ	Fast Interrupt	Input
IRQ0 - IRQn	Interrupt 0 - Interrupt n	Input

29.6 Product Dependencies

29.6.1 I/O Lines

The interrupt signals FIQ and IRQ0 to IRQn are normally multiplexed through the PIO controllers. Depending on the features of the PIO controller used in the product, the pins must be programmed in accordance with their assigned interrupt function. This is not applicable when the PIO controller used in the product is transparent on the input path.

29.6.2 Power Management

The Advanced Interrupt Controller is continuously clocked. The Power Management Controller has no effect on the Advanced Interrupt Controller behavior.

The assertion of the Advanced Interrupt Controller outputs, either nIRQ or nFIQ, wakes up the ARM processor while it is in Idle Mode. The General Interrupt Mask feature enables the AIC to wake up the processor without asserting the interrupt line of the processor, thus providing synchronization of the processor on an event.

29.6.3 Interrupt Sources

The Interrupt Source 0 is always located at FIQ. If the product does not feature an FIQ pin, the Interrupt Source 0 cannot be used.

The Interrupt Source 1 is always located at System Interrupt. This is the result of the OR-wiring of the system peripheral interrupt lines. When a system interrupt occurs, the service routine must first distinguish the cause of the interrupt. This is performed by reading successively the status registers of the above mentioned system peripherals.

The interrupt sources 2 to 31 can either be connected to the interrupt outputs of an embedded user peripheral or to external interrupt lines. The external interrupt lines can be connected directly, or through the PIO Controller.

The PIO Controllers are considered as user peripherals in the scope of interrupt handling. Accordingly, the PIO Controller interrupt lines are connected to the Interrupt Sources 2 to 31.

The peripheral identification defined at the product level corresponds to the interrupt source number (as well as the bit number controlling the clock of the peripheral). Consequently, to simplify the description of the functional operations and the user interface, the interrupt sources are named FIQ, SYS, and PID2 to PID31.

29.7 Functional Description

29.7.1 Interrupt Source Control

29.7.1.1 Interrupt Source Mode

The Advanced Interrupt Controller independently programs each interrupt source. The SRCTYPE field of the corresponding AIC_SMR (Source Mode Register) selects the interrupt condition of each source.

The internal interrupt sources wired on the interrupt outputs of the embedded peripherals can be programmed either in level-sensitive mode or in edge-triggered mode. The active level of the internal interrupts is not important for the user.

The external interrupt sources can be programmed either in high level-sensitive or low level-sensitive modes, or in positive edge-triggered or negative edge-triggered modes.

29.7.1.2 Interrupt Source Enabling

Each interrupt source, including the FIQ in source 0, can be enabled or disabled by using the command registers; AIC_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.

29.7.1.3 Interrupt Clearing and Setting

All interrupt sources programmed to be edge-triggered (including the FIQ in source 0) can be individually set or cleared by writing respectively the AIC_ISCR and AIC_ICCR registers. Clearing or setting interrupt sources programmed in level-sensitive mode has no effect.

The clear operation is perfunctory, as the software must perform an action to reinitialize the “memorization” circuitry activated when the source is programmed in edge-triggered mode. However, the set operation is available for auto-test or software debug purposes. It can also be used to execute an AIC-implementation of a software interrupt.

The AIC features an automatic clear of the current interrupt when the AIC_IVR (Interrupt Vector Register) is read. Only the interrupt source being detected by the AIC as the current interrupt is affected by this operation. (See “Priority Controller” on page 383.) 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 387.)

The automatic clear of the interrupt source 0 is performed when AIC_FVR is read.

29.7.1.4 Interrupt Status

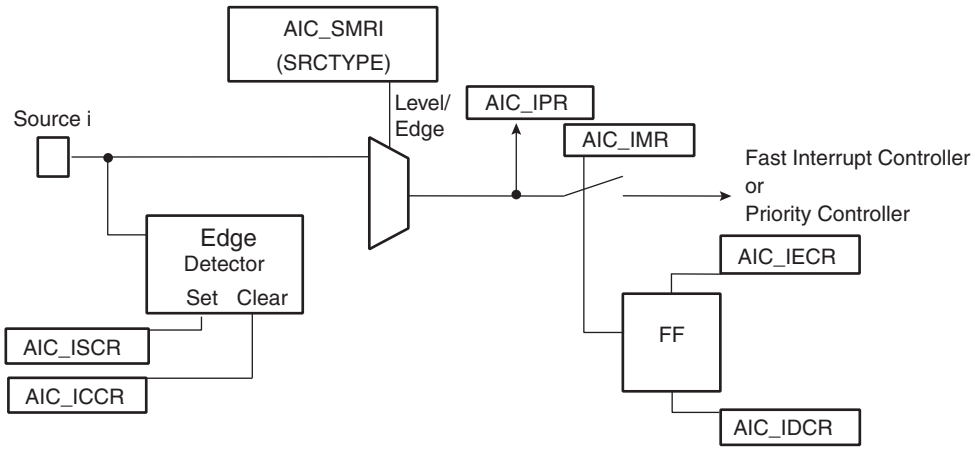
For each interrupt, the AIC operation originates in AIC_IPR (Interrupt Pending Register) and its mask in AIC_IMR (Interrupt Mask Register). AIC_IPR enables the actual activity of the sources, whether masked or not.

The AIC_ISR register reads the number of the current interrupt (see “Priority Controller” on page 383) and the register AIC_CISR gives an image of the signals nIRQ and nFIQ driven on the processor.

Each status referred to above can be used to optimize the interrupt handling of the systems.

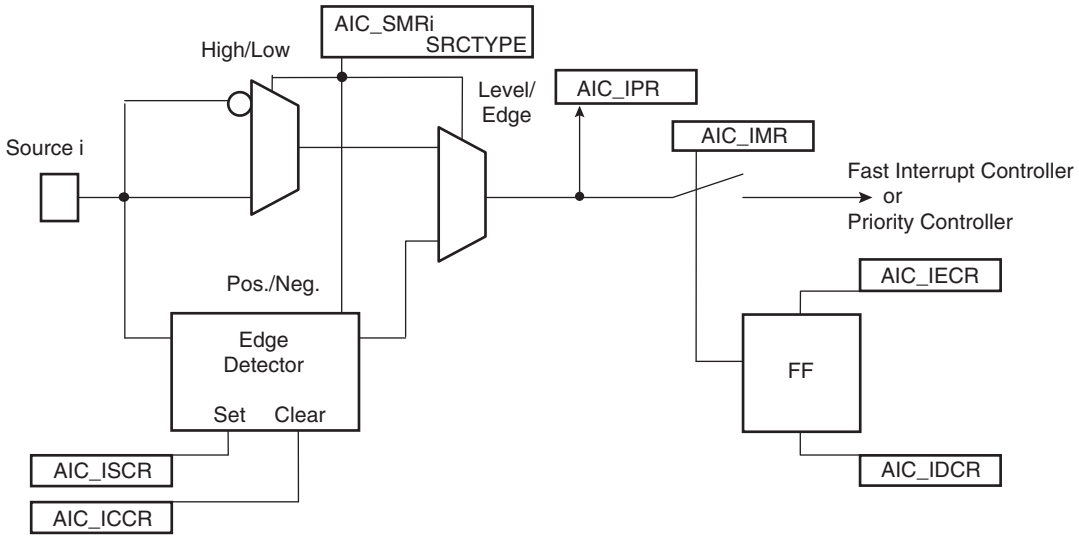
29.7.1.5 Internal Interrupt Source Input Stage

Figure 29-4. Internal Interrupt Source Input Stage



29.7.1.6 External Interrupt Source Input Stage

Figure 29-5. External Interrupt Source Input Stage



29.7.2 Interrupt Latencies

Global interrupt latencies depend on several parameters, including:

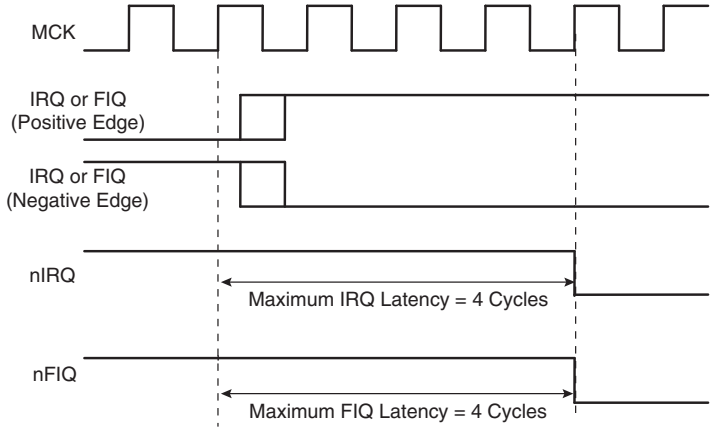
- The time the software masks the interrupts.
- Occurrence, either at the processor level or at the AIC level.
- The execution time of the instruction in progress when the interrupt occurs.
- The treatment of higher priority interrupts and the resynchronization of the hardware signals.

This section addresses only the hardware resynchronizations. It gives details of the latency times between the event on an external interrupt leading in a valid interrupt (edge or level) or the assertion of an internal interrupt source and the assertion of the nIRQ or nFIQ line on the processor. The resynchronization time depends on the programming of the interrupt source and on its type (internal or external). For the standard interrupt, resynchronization times are given assuming there is no higher priority in progress.

The PIO Controller multiplexing has no effect on the interrupt latencies of the external interrupt sources.

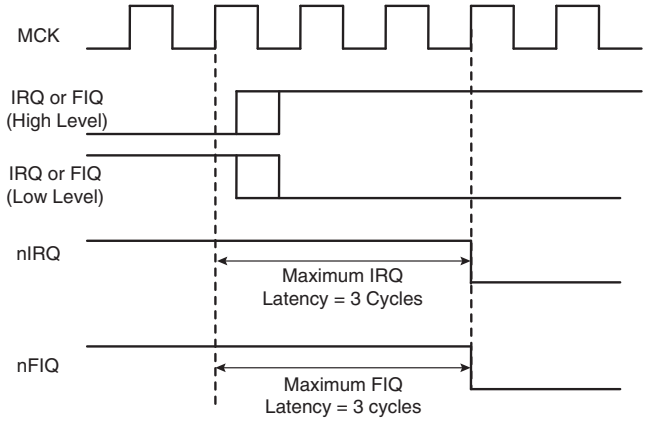
29.7.2.1 External Interrupt Edge Triggered Source

Figure 29-6. External Interrupt Edge Triggered Source



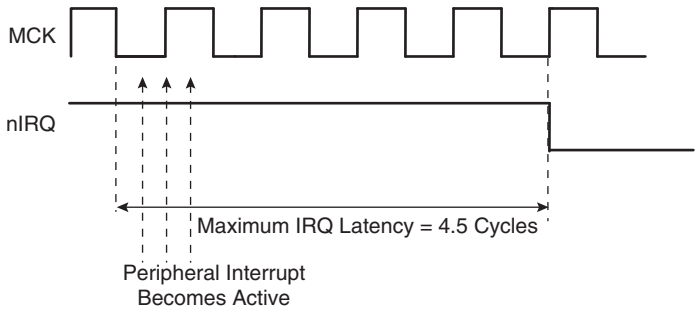
29.7.2.2 External Interrupt Level Sensitive Source

Figure 29-7. External Interrupt Level Sensitive Source



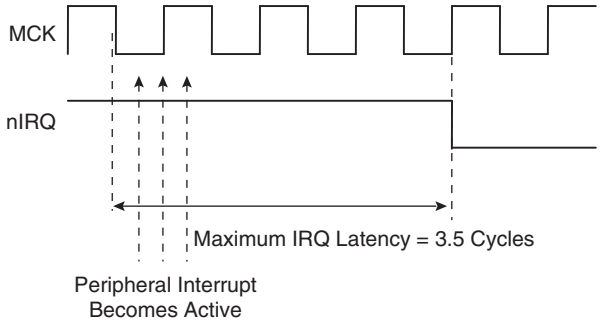
29.7.2.3 Internal Interrupt Edge Triggered Source

Figure 29-8. Internal Interrupt Edge Triggered Source



29.7.2.4 Internal Interrupt Level Sensitive Source

Figure 29-9. Internal Interrupt Level Sensitive Source



29.7.3 Normal Interrupt

29.7.3.1 Priority Controller

An 8-level priority controller drives the nIRQ line of the processor, depending on the interrupt conditions occurring on the interrupt sources 1 to 31 (except for those programmed in Fast Forcing).

Each interrupt source has a programmable priority level of 7 to 0, which is user-definable by writing the PRIOR field of the corresponding AIC_SMR (Source Mode Register). Level 7 is the highest priority and level 0 the lowest.

As soon as an interrupt condition occurs, as defined by the SRCTYPE field of the AIC_SMR (Source Mode Register), the nIRQ line is asserted. As a new interrupt condition might have happened on other interrupt sources since the nIRQ has been asserted, the priority controller determines the current interrupt at the time the AIC_IVR (Interrupt Vector Register) is read. **The read of AIC_IVR is the entry point of the interrupt handling** which allows the AIC to consider that the interrupt has been taken into account by the software.

The current priority level is defined as the priority level of the current interrupt.

If several interrupt sources of equal priority are pending and enabled when the AIC_IVR is read, the interrupt with the lowest interrupt source number is serviced first.

The nIRQ line can be asserted only if an interrupt condition occurs on an interrupt source with a higher priority. If an interrupt condition happens (or is pending) during the interrupt treatment in progress, it is delayed until the software indicates to the AIC the end of the current service by writing the AIC_EOICR (End of Interrupt Command Register). **The write of AIC_EOICR is the exit point of the interrupt handling**.

29.7.3.2 Interrupt Nesting

The priority controller utilizes interrupt nesting in order for the high priority interrupt to be handled during the service of lower priority interrupts. This requires the interrupt service routines of the lower interrupts to re-enable the interrupt at the processor level.

When an interrupt of a higher priority happens during an already occurring interrupt service routine, the nIRQ line is re-asserted. If the interrupt is enabled at the core level, the current execution is interrupted and the new interrupt service routine should read the AIC_IVR. At this time, the current interrupt number and its priority level are pushed into an embedded hardware stack, so that they are saved and restored when the higher priority interrupt servicing is finished and the AIC_EOICR is written.

The AIC is equipped with an 8-level wide hardware stack in order to support up to eight interrupt nestings pursuant to having eight priority levels.

29.7.3.3 Interrupt Vectoring

The interrupt handler addresses corresponding to each interrupt source can be stored in the registers AIC_SVR1 to AIC_SVR31 (Source Vector Register 1 to 31). When the processor reads AIC_IVR (Interrupt Vector Register), the value written into AIC_SVR corresponding to the current interrupt is returned.

This feature offers a way to branch in one single instruction to the handler corresponding to the current interrupt, as AIC_IVR is mapped at the absolute address 0xFFFF F100 and thus accessible from the ARM interrupt vector at address 0x0000 0018 through the following instruction:

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

When the processor executes this instruction, it loads the read value in AIC_IVR in its program counter, thus branching the execution on the correct interrupt handler.

This feature is often not used when the application is based on an operating system (either real time or not). Operating systems often have a single entry point for all the interrupts and the first task performed is to discern the source of the interrupt.

However, it is strongly recommended to port the operating system on AT91 products by supporting the interrupt vectoring. This can be performed by defining all the AIC_SVR of the interrupt source to be handled by the operating system at the address of its interrupt handler. When doing so, the interrupt vectoring permits a critical interrupt to transfer the execution on a specific very fast handler and not onto the operating system's general interrupt handler. This facilitates the support of hard real-time tasks (input/outputs of voice/audio buffers and software peripheral handling) to be handled efficiently and independently of the application running under an operating system.

29.7.3.4 Interrupt Handlers

This section gives an overview of the fast interrupt handling sequence when using the AIC. It is assumed that the programmer understands the architecture of the ARM processor, and especially the processor interrupt modes and the associated status bits.

It is assumed that:

1. The Advanced Interrupt Controller has been programmed, AIC_SVR registers are loaded with corresponding interrupt service routine addresses and interrupts are enabled.
2. The instruction at the ARM interrupt exception vector address is required to work with the vectoring

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

When nIRQ is asserted, if the bit "I" of CPSR is 0, the sequence is as follows:

1. The CPSR is stored in SPSR_irq, the current value of the Program Counter is loaded in the Interrupt link register (R14_irq) and the Program Counter (R15) is loaded with 0x18. In the following cycle during fetch at address 0x1C, the ARM core adjusts R14_irq, decrementing it by four.
2. The ARM core enters Interrupt mode, if it has not already done so.
3. When the instruction loaded at address 0x18 is executed, the program counter is loaded with the value read in AIC_IVR. Reading the AIC_IVR has the following effects:
   - Sets the current interrupt to be the pending and enabled interrupt with the highest priority. The current level is the priority level of the current interrupt.
   - De-asserts the nIRQ line on the processor. Even if vectoring is not used, AIC_IVR must be read in order to de-assert nIRQ.
   - Automatically clears the interrupt, if it has been programmed to be edge-triggered.
   - Pushes the current level and the current interrupt number on to the stack.
   - Returns the value written in the AIC_SVR corresponding to the current interrupt.
4. The previous step has the effect of branching to the corresponding interrupt service routine. This should start by saving the link register (R14_irq) and SPSR_IRQ. The link register must be decremented by four when it is saved if it is to be restored directly into the program counter at the end of the interrupt. For example, the instruction `SUB PC, LR, #4` may be used.
5. Further interrupts can then be unmasked by clearing the "I" bit in CPSR, allowing re-assertion of the nIRQ to be taken into account by the core. This can happen if an interrupt with a higher priority than the current interrupt occurs.
6. The interrupt handler can then proceed as required, saving the registers that 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).

29.7.4 Fast Interrupt

29.7.4.1 Fast Interrupt Source

The interrupt source 0 is the only source which can raise a fast interrupt request to the processor except if fast forcing is used. The interrupt source 0 is generally connected to a FIQ pin of the product, either directly or through a PIO Controller.

29.7.4.2 Fast Interrupt Control

The fast interrupt logic of the AIC has no priority controller. The mode of interrupt source 0 is programmed with the AIC_SMR0 and the field PRIOR of this register is not used even if it reads what has been written. The field SRCTYPE of AIC_SMR0 enables programming the fast interrupt source to be positive-edge triggered or negative-edge triggered or high-level sensitive or low-level sensitive

Writing 0x1 in the AIC_IECR (Interrupt Enable Command Register) and AIC_IDCR (Interrupt Disable Command Register) respectively enables and disables the fast interrupt. The bit 0 of AIC_IMR (Interrupt Mask Register) indicates whether the fast interrupt is enabled or disabled.

29.7.4.3 Fast Interrupt Vectoring

The fast interrupt handler address can be stored in AIC_SVR0 (Source Vector Register 0). The value written into this register is returned when the processor reads AIC_FVR (Fast Vector Register). This offers a way to branch in one single instruction to the interrupt handler, as AIC_FVR is mapped at the absolute address 0xFFFF F104 and thus accessible from the ARM fast interrupt vector at address 0x0000 001C through the following instruction:

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

When the processor executes this instruction it loads the value read in AIC_FVR in its program counter, thus branching the execution on the fast interrupt handler. It also automatically performs the clear of the fast interrupt source if it is programmed in edge-triggered mode.

29.7.4.4 Fast Interrupt Handlers

This section gives an overview of the fast interrupt handling sequence when using the AIC. It is assumed that the programmer understands the architecture of the ARM processor, and especially the processor interrupt modes and associated status bits.

Assuming that:

1. The Advanced Interrupt Controller has been programmed, AIC_SVR0 is loaded with the fast interrupt service routine address, and the interrupt source 0 is enabled.
2. The Instruction at address 0x1C (FIQ exception vector address) is required to vector the fast interrupt:

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

3. The user does not need nested fast interrupts.

When nFIQ is asserted, if the bit “F” of CPSR is 0, the sequence is:

1. The CPSR is stored in SPSR_fiq, the current value of the program counter is loaded in the FIQ link register (R14_FIQ) and the program counter (R15) is loaded with 0x1C. In

the following cycle, during fetch at address 0x20, the ARM core adjusts R14_fiq, decrementing it by four.

2. The ARM core enters FIQ mode.
3. When the instruction loaded at address 0x1C is executed, the program counter is loaded with the value read in AIC_FVR. Reading the AIC_FVR has effect of automatically clearing the fast interrupt, if it has been programmed to be edge triggered. In this case only, it de-asserts the nFIQ line on the processor.
4. The previous step enables branching to the corresponding interrupt service routine. It is not necessary to save the link register R14_fiq and SPSR_fiq if nested fast interrupts are not needed.
5. The Interrupt Handler can then proceed as required. It is not necessary to save registers R8 to R13 because FIQ mode has its own dedicated registers and the user R8 to R13 are banked. The other registers, R0 to R7, must be saved before being used, and restored at the end (before the next step). Note that if the fast interrupt is programmed to be level sensitive, the source of the interrupt must be cleared during this phase in order to de-assert the interrupt source 0.
6. Finally, the Link Register R14_fiq is restored into the PC after decrementing it by four (with instruction `SUB PC, LR, #4` for example). This has the effect of returning from the interrupt to whatever was being executed before, loading the CPSR with the SPSR and masking or unmasking the fast interrupt depending on the state saved in the SPSR.

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

Another way to handle the fast interrupt is to map the interrupt service routine at the address of the ARM vector 0x1C. This method does not use the vectoring, so that reading AIC_FVR must be performed at the very beginning of the handler operation. However, this method saves the execution of a branch instruction.

29.7.4.5 Fast Forcing

The Fast Forcing feature of the advanced interrupt controller provides redirection of any normal Interrupt source on the fast interrupt controller.

Fast Forcing is enabled or disabled by writing to the Fast Forcing Enable Register (AIC_FFER) and the Fast Forcing Disable Register (AIC_FFDR). Writing to these registers results in an update of the Fast Forcing Status Register (AIC_FFSR) that controls the feature for each internal or external interrupt source.

When Fast Forcing is disabled, the interrupt sources are handled as described in the previous pages.

When Fast Forcing is enabled, the edge/level programming and, in certain cases, edge detection of the interrupt source is still active but the source cannot trigger a normal interrupt to the processor and is not seen by the priority handler.

If the interrupt source is programmed in level-sensitive mode and an active level is sampled, Fast Forcing results in the assertion of the nFIQ line to the core.

If the interrupt source is programmed in edge-triggered mode and an active edge is detected, Fast Forcing results in the assertion of the nFIQ line to the core.

The Fast Forcing feature does not affect the Source 0 pending bit in the Interrupt Pending Register (AIC_IPR).

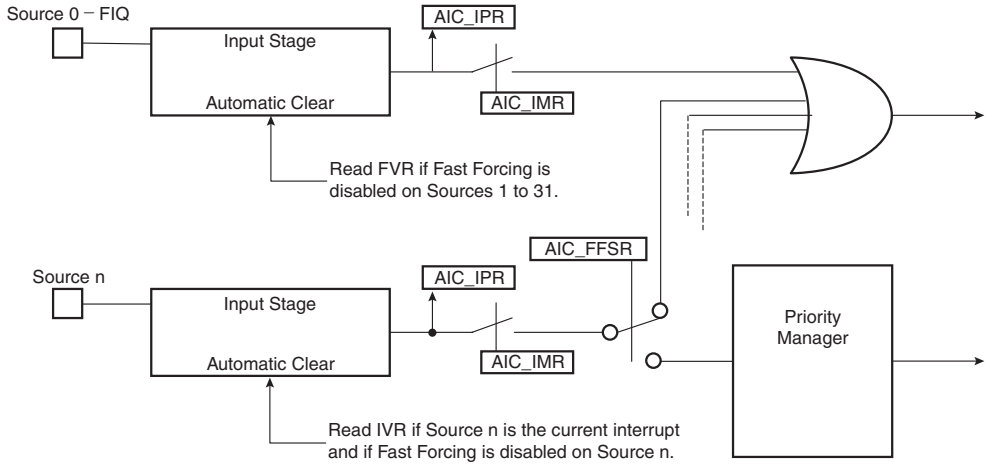
The FIQ Vector Register (AIC_FVR) reads the contents of the Source Vector Register 0 (AIC_SVR0), whatever the source of the fast interrupt may be. The read of the FVR does not clear the Source 0 when the fast forcing feature is used and the interrupt source should be cleared by writing to the Interrupt Clear Command Register (AIC_ICCR).

All enabled and pending interrupt sources that have the fast forcing feature enabled and that are programmed in edge-triggered mode must be cleared by writing to the Interrupt Clear Command Register. In doing so, they are cleared independently and thus lost interrupts are prevented.

The read of AIC_IVR does not clear the source that has the fast forcing feature enabled.

The source 0, reserved to the fast interrupt, continues operating normally and becomes one of the Fast Interrupt sources.

Figure 29-10. Fast Forcing



29.7.5 Protect Mode

The Protect Mode permits reading the Interrupt Vector Register without performing the associated automatic operations. This is necessary when working with a debug system. When a debugger, working either with a Debug Monitor or the ARM processor's ICE, stops the applications and updates the opened windows, it might read the AIC User Interface and thus the IVR. This has undesirable consequences:

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

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

This is avoided by using the Protect Mode. Writing 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.

29.7.6 Spurious Interrupt

The Advanced Interrupt Controller features protection against spurious interrupts. A spurious interrupt is defined as being the assertion of an interrupt source long enough for the AIC to assert the nIRQ, but no longer present when AIC_IVR is read. This is most prone to occur when:

- An external interrupt source is programmed in level-sensitive mode and an active level occurs for only a short time.
- An internal interrupt source is programmed in level sensitive and the output signal of the corresponding embedded peripheral is activated for a short time. (As in the case for the Watchdog.)
- An interrupt occurs just a few cycles before the software begins to mask it, thus resulting in a pulse on the interrupt source.

The AIC detects a spurious interrupt at the time the AIC_IVR is read while no enabled interrupt source is pending. When this happens, the AIC returns the value stored by the programmer in AIC_SPU (Spurious Vector Register). The programmer must store the address of a spurious interrupt handler in AIC_SPU as part of the application, to enable an as fast as possible return to the normal execution flow. This handler writes in AIC_EOICR and performs a return from interrupt.

29.7.7 General Interrupt Mask

The AIC features a General Interrupt Mask bit to prevent interrupts from reaching the processor. Both the nIRQ and the nFIQ lines are driven to their inactive state if the bit GMSK in AIC_DCR (Debug Control Register) is set. However, this mask does not prevent waking up the processor if it has entered Idle Mode. This function facilitates synchronizing the processor on a next event and, as soon as the event occurs, performs subsequent operations without having to handle an interrupt. It is strongly recommended to use this mask with caution.

29.8 Advanced Interrupt Controller (AIC) User Interface

29.8.1 Base Address

The AIC is mapped at the address **0xFFFF F000**. It has a total 4-Kbyte addressing space. This permits the vectoring feature, as the PC-relative load/store instructions of the ARM processor support only a ± 4-Kbyte offset.

Table 29-2. 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(3)	AIC_IPR	Read-only	0x0(1)
0x110	Interrupt Mask Register(3)	AIC_IMR	Read-only	0x0
0x114	Core Interrupt Status Register	AIC_CISR	Read-only	0x0
0x118 - 0x11C	Reserved	---	---	---
0x120	Interrupt Enable Command Register(3)	AIC_IECR	Write-only	---
0x124	Interrupt Disable Command Register(3)	AIC_IDCR	Write-only	---
0x128	Interrupt Clear Command Register(3)	AIC_ICCR	Write-only	---
0x12C	Interrupt Set Command Register(3)	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(3)	AIC_FFER	Write-only	---
0x144	Fast Forcing Disable Register(3)	AIC_FFDR	Write-only	---
0x148	Fast Forcing Status Register(3)	AIC_FFSR	Read-only	0x0
0x14C - 0x1E0	Reserved	---	---	---
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.

3. PID2...PID31 bit fields refer to the identifiers as defined in the Peripheral Identifiers Section of the product datasheet.

4. Values in the Version Register vary with the version of the IP block implementation.

29.8.2 AIC Source Mode Register

Register Name: AIC_SMR0..AIC_SMR31

Address: 0xFFFFF000

Access Type: Read-write

Reset Value: 0x0

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

- **PRIOR: Priority Level**

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

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

The priority level is not used for the FIQ in the related SMR register AIC_SMRx.

- **SRCTYPE: Interrupt Source Type**

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

SRCTYPE		Internal Interrupt Sources	External Interrupt Sources
0	0	High level Sensitive	Low level Sensitive
0	1	Positive edge triggered	Negative edge triggered
1	0	High level Sensitive	High level Sensitive
1	1	Positive edge triggered	Positive edge triggered

29.8.3 AIC Source Vector Register

Register Name: AIC_SVR0..AIC_SVR31

Address: 0xFFFFF080

Access Type: Read-write

Reset Value: 0x0

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

- **VECTOR: Source Vector**

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

29.8.4 AIC Interrupt Vector Register

Register Name: AIC_IVR

Address: 0xFFFFF100

Access Type: Read-only

Reset Value: 0x0

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

- **IRQV: Interrupt Vector Register**

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

The Source Vector Register is indexed using the current interrupt number when the Interrupt Vector Register is read.

When there is no current interrupt, the Interrupt Vector Register reads the value stored in AIC_SPU.

29.8.5 AIC FIQ Vector Register

Register Name: AIC_FVR

Address: 0xFFFFF104

Access Type: Read-only

Reset Value: 0x0

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

- **FIQV: FIQ Vector Register**

The FIQ Vector Register contains the vector programmed by the user in the Source Vector Register 0. When there is no fast interrupt, the FIQ Vector Register reads the value stored in AIC_SPU.

29.8.6 AIC Interrupt Status Register

Register Name: AIC_ISR

Address: 0xFFFFF108

Access Type: Read-only

Reset Value: 0x0

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

- **IRQID: Current Interrupt Identifier**

The Interrupt Status Register returns the current interrupt source number.

29.8.7 AIC Interrupt Pending Register

Register Name: AIC_IPR

Address: 0xFFFFF10C

Access Type: Read-only

Reset Value: 0x0

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Pending**

0 = Corresponding interrupt is not pending.

1 = Corresponding interrupt is pending.

29.8.8 AIC Interrupt Mask Register

Register Name: AIC_IMR

Address: 0xFFFFF110

Access Type: Read-only

Reset Value: 0x0

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Mask**

0 = Corresponding interrupt is disabled.

1 = Corresponding interrupt is enabled.

29.8.9 AIC Core Interrupt Status Register

Register Name: AIC_CISR

Address: 0xFFFFF114

Access Type: Read-only

Reset Value: 0x0

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

- **NFIQ: NFIQ Status**

0 = nFIQ line is deactivated.

1 = nFIQ line is active.

- **NIRQ: NIRQ Status**

0 = nIRQ line is deactivated.

1 = nIRQ line is active.

29.8.10 AIC Interrupt Enable Command Register

Register Name: AIC_IECR

Address: 0xFFFFF120

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Enable**

0 = No effect.

1 = Enables corresponding interrupt.

29.8.11 AIC Interrupt Disable Command Register

Register Name: AIC_IDCR

Address: 0xFFFFF124

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Disable**

0 = No effect.

1 = Disables corresponding interrupt.

29.8.12 AIC Interrupt Clear Command Register

Register Name: AIC_ICCR

Address: 0xFFFFF128

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Clear**

0 = No effect.

1 = Clears corresponding interrupt.

29.8.13 AIC Interrupt Set Command Register

Register Name: AIC_ISCR

Address: 0xFFFFF12C

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: Interrupt Set**

0 = No effect.

1 = Sets corresponding interrupt.

29.8.14 AIC End of Interrupt Command Register

Register Name: AIC_EOICR

Address: 0xFFFFF130

Access Type: Write-only

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

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

29.8.15 AIC Spurious Interrupt Vector Register

Register Name: AIC_SPU

Address: 0xFFFFF134

Access Type: Read-write

Reset Value: 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.

29.8.16 AIC Debug Control Register

Register Name: AIC_DCR

Address: 0xFFFFF138

Access Type: Read-write

Reset Value: 0x0

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

- **PROT: Protection Mode**

0 = The Protection Mode is disabled.

1 = The Protection Mode is enabled.

- **GMSK: General Mask**

0 = The nIRQ and nFIQ lines are normally controlled by the AIC.

1 = The nIRQ and nFIQ lines are tied to their inactive state.

29.8.17 AIC Fast Forcing Enable Register

Register Name: AIC_FFER

Address: 0xFFFFF140

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

- **SYS, PID2-PID31: Fast Forcing Enable**

0 = No effect.

1 = Enables the fast forcing feature on the corresponding interrupt.

29.8.18 AIC Fast Forcing Disable Register

Register Name: AIC_FFDR

Address: 0xFFFFF144

Access Type: Write-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

- **SYS, PID2-PID31: Fast Forcing Disable**

0 = No effect.

1 = Disables the Fast Forcing feature on the corresponding interrupt.

29.8.19 AIC Fast Forcing Status Register

Register Name: AIC_FFSR

Address: 0xFFFFF148

Access Type: Read-only

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

- **SYS, PID2-PID31: Fast Forcing Status**

0 = The Fast Forcing feature is disabled on the corresponding interrupt.

1 = The Fast Forcing feature is enabled on the corresponding interrupt.

30. Debug Unit (DBGU)

30.1 Overview

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 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.

30.2 Block Diagram

Figure 30-1. Debug Unit Functional Block Diagram

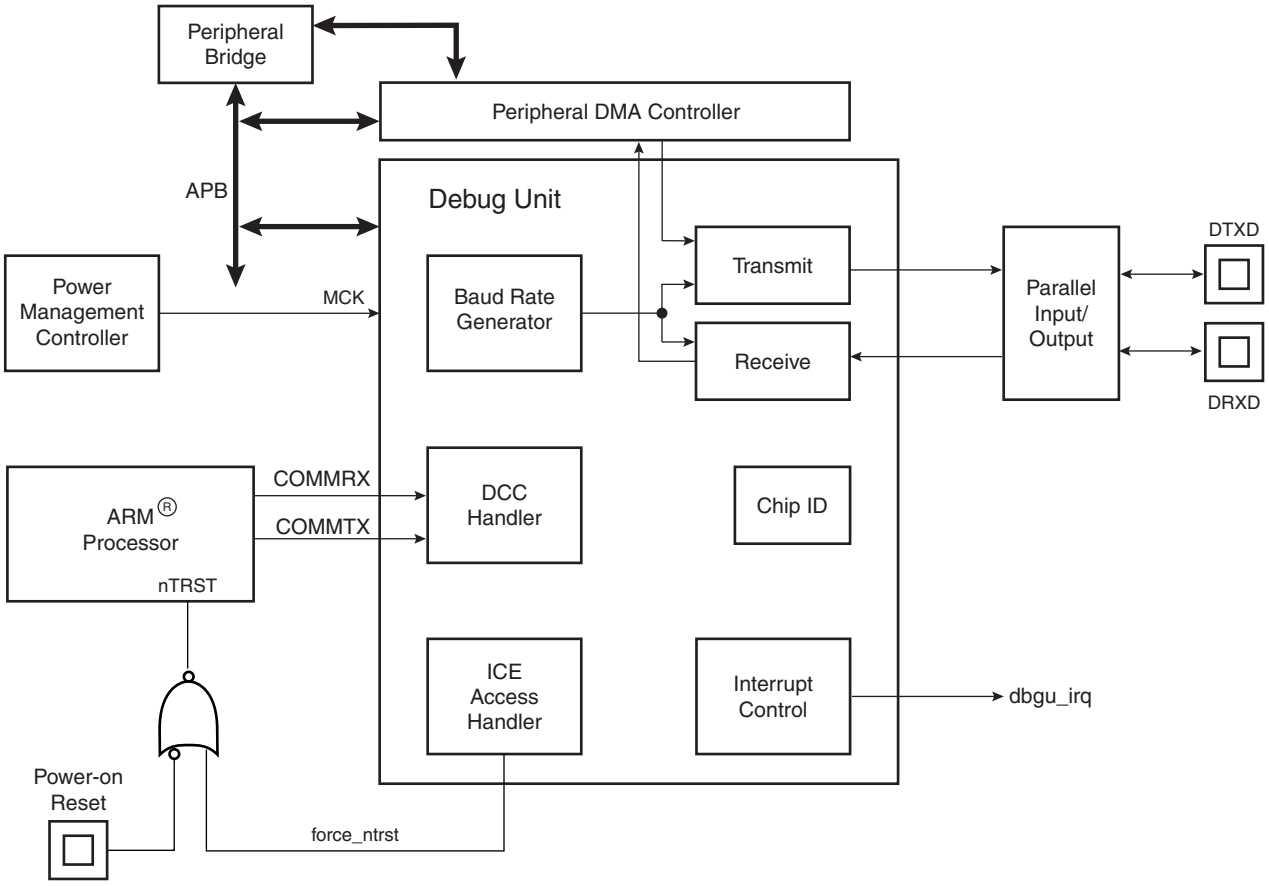
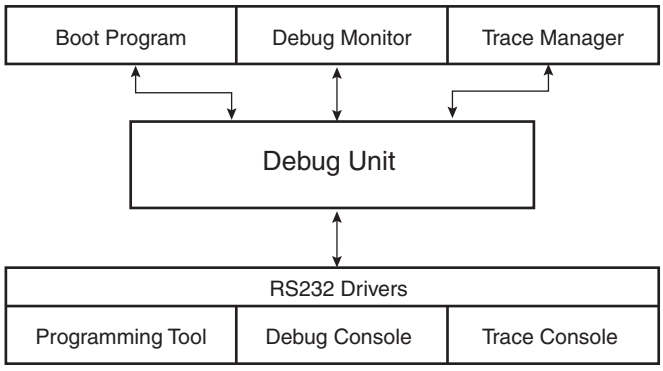


Table 30-1. Debug Unit Pin Description

Pin Name	Description	Type
DRXD	Debug Receive Data	Input
DTXD	Debug Transmit Data	Output

Figure 30-2. Debug Unit Application Example



30.3 Product Dependencies

30.3.1 I/O Lines

Depending on product integration, the Debug Unit pins may be multiplexed with PIO lines. In this case, the programmer must first configure the corresponding PIO Controller to enable I/O lines operations of the Debug Unit.

30.3.2 Power Management

Depending on product integration, the Debug Unit clock may be controllable through the Power Management Controller. In this case, the programmer must first configure the PMC to enable the Debug Unit clock. Usually, the peripheral identifier used for this purpose is 1.

30.3.3 Interrupt Source

Depending on product integration, the Debug Unit interrupt line is connected to one of the interrupt sources of the Advanced Interrupt Controller. Interrupt handling requires programming of the AIC before configuring the Debug Unit. Usually, the Debug Unit interrupt line connects to the interrupt source 1 of the AIC, which may be shared with the real-time clock, the system timer interrupt lines and other system peripheral interrupts, as shown in Figure 30-1. This sharing requires the programmer to determine the source of the interrupt when the source 1 is triggered.

30.4 UART Operations

The Debug Unit operates as a UART, (asynchronous mode only) and supports only 8-bit character handling (with parity). It has no clock pin.

The Debug Unit's UART is made up of a receiver and a transmitter that operate independently, and a common baud rate generator. Receiver timeout and transmitter time guard are not implemented. However, all the implemented features are compatible with those of a standard USART.

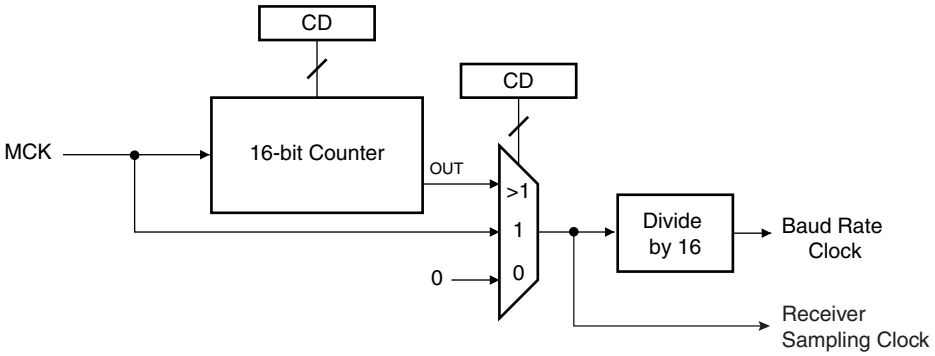
30.4.1 Baud Rate Generator

The baud rate generator provides the bit period clock named baud rate clock to both the receiver and the transmitter.

The baud rate clock is the master clock divided by 16 times the value (CD) written in DBGU_BRGR (Baud Rate Generator Register). If DBGU_BRGR is set to 0, the baud rate clock is disabled and the Debug Unit's UART remains inactive. The maximum allowable baud rate is Master Clock divided by 16. The minimum allowable baud rate is Master Clock divided by (16 x 65536).

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

Figure 30-3. Baud Rate Generator



30.4.2 Receiver

30.4.2.1 Receiver Reset, Enable and Disable

After device reset, the Debug Unit receiver is disabled and must be enabled before being used. The receiver can be enabled by writing the control register DBGU_CR with the bit RXEN at 1. At this command, the receiver starts looking for a start bit.

The programmer can disable the receiver by writing DBGU_CR with the bit RXDIS at 1. If the receiver is waiting for a start bit, it is immediately stopped. However, if the receiver has already detected a start bit and is receiving the data, it waits for the stop bit before actually stopping its operation.

The programmer can also put the receiver in its reset state by writing DBGU_CR with the bit RSTRX at 1. In doing so, the receiver immediately stops its current operations and is disabled, whatever its current state. If RSTRX is applied when data is being processed, this data is lost.

30.4.2.2 Start Detection and Data Sampling

The Debug Unit only supports asynchronous operations, and this affects only its receiver. The Debug Unit receiver detects the start of a received character by sampling the DRXD signal until it detects a valid start bit. A low level (space) on DRXD is interpreted as a valid start bit if it is detected for more than 7 cycles of the sampling clock, which is 16 times the baud rate. Hence, a space that is longer than 7/16 of the bit period is detected as a valid start bit. A space which is 7/16 of a bit period or shorter is ignored and the receiver continues to wait for a valid start bit.

When a valid start bit has been detected, the receiver samples the DRXD at the theoretical 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 30-4. Start Bit Detection

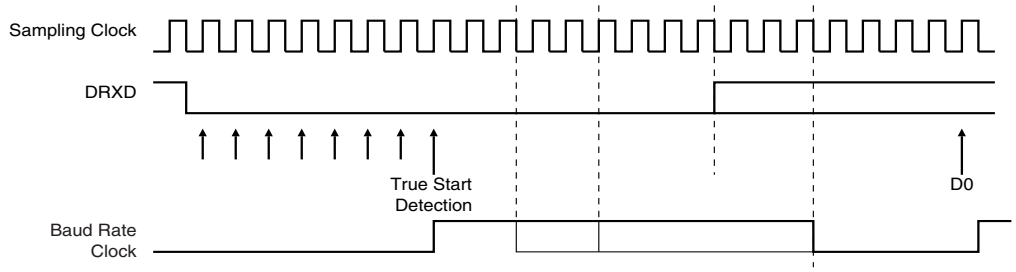
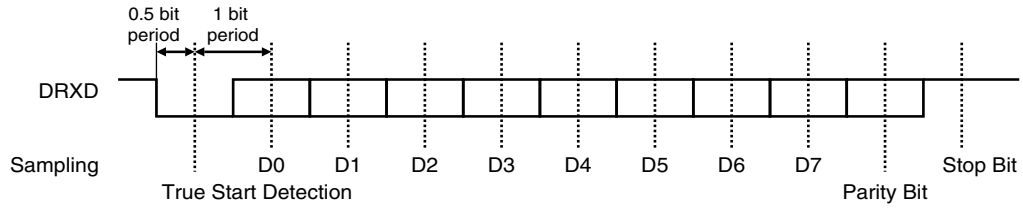


Figure 30-5. Character Reception

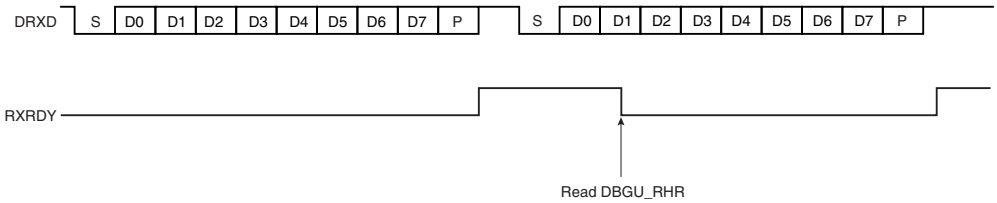
Example: 8-bit, parity enabled 1 stop



30.4.2.3 *Receiver Ready*

When a complete character is received, it is transferred to the DBGU_RHR and the RXRDY status bit in DBGU_SR (Status Register) is set. The bit RXRDY is automatically cleared when the receive holding register DBGU_RHR is read.

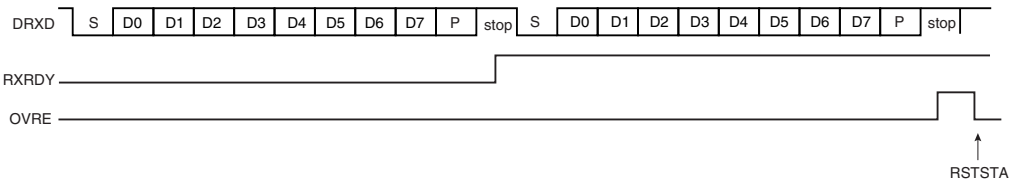
Figure 30-6. Receiver Ready



30.4.2.4 *Receiver Overrun*

If DBGU_RHR has not been read by the software (or the Peripheral Data Controller) since the last transfer, the RXRDY bit is still set and a new character is received, the OVRE status bit in DBGU_SR is set. OVRE is cleared when the software writes the control register DBGU_CR with the bit RSTSTA (Reset Status) at 1.

Figure 30-7. Receiver Overrun

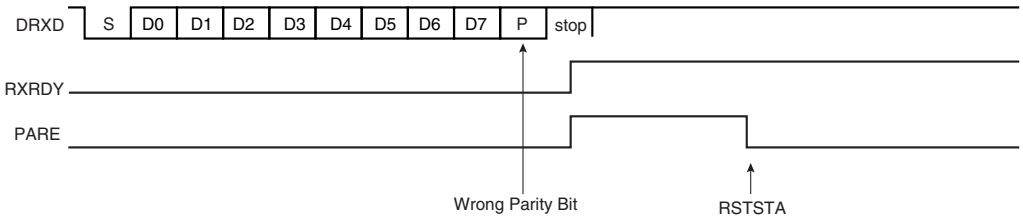


30.4.2.5 *Parity Error*

Each time a character is received, the receiver calculates the parity of the received data bits, in accordance with the field PAR in DBGU_MR. It then compares the result with the received parity

bit. If different, the parity error bit PARE in DBGU_SR is set at the same time the RXRDY is set. The parity bit is cleared when the control register DBGU_CR is written with the bit RSTSTA (Reset Status) at 1. If a new character is received before the reset status command is written, the PARE bit remains at 1.

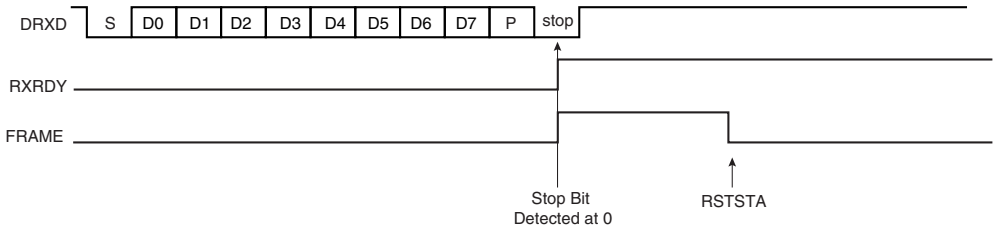
Figure 30-8. Parity Error



30.4.2.6 Receiver Framing Error

When a start bit is detected, it generates a character reception when all the data bits have been sampled. The stop bit is also sampled and when it is detected at 0, the FRAME (Framing Error) bit in DBGU_SR is set at the same time the RXRDY bit is set. The bit FRAME remains high until the control register DBGU_CR is written with the bit RSTSTA at 1.

Figure 30-9. Receiver Framing Error



30.4.3 Transmitter

30.4.3.1 Transmitter Reset, Enable and Disable

After device reset, the Debug Unit transmitter is disabled and it must be enabled before being used. The transmitter is enabled by writing the control register DBGU_CR with the bit TXEN at 1. From this command, the transmitter waits for a character to be written in the Transmit Holding Register DBGU_THR before actually starting the transmission.

The programmer can disable the transmitter by writing DBGU_CR with the bit TXDIS at 1. If the transmitter is not operating, it is immediately stopped. However, if a character is being processed into the Shift Register and/or a character has been written in the Transmit Holding Register, the characters are completed before the transmitter is actually stopped.

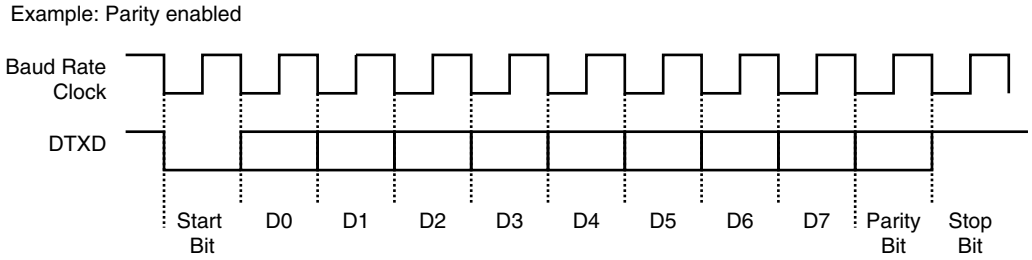
The programmer can also put the transmitter in its reset state by writing the DBGU_CR with the bit RSTTX at 1. This immediately stops the transmitter, whether or not it is processing characters.

30.4.3.2 Transmit Format

The Debug Unit transmitter drives the pin DTXD at the baud rate clock speed. The line is driven depending on the format defined in the Mode Register and the data stored in the Shift Register. One start bit at level 0, then the 8 data bits, from the lowest to the highest bit, one optional parity bit and one stop bit at 1 are consecutively shifted out as shown on the following figure. The field

PARE in the mode register DBGU_MR defines whether or not a parity bit is shifted out. When a parity bit is enabled, it can be selected between an odd parity, an even parity, or a fixed space or mark bit.

Figure 30-10. Character Transmission

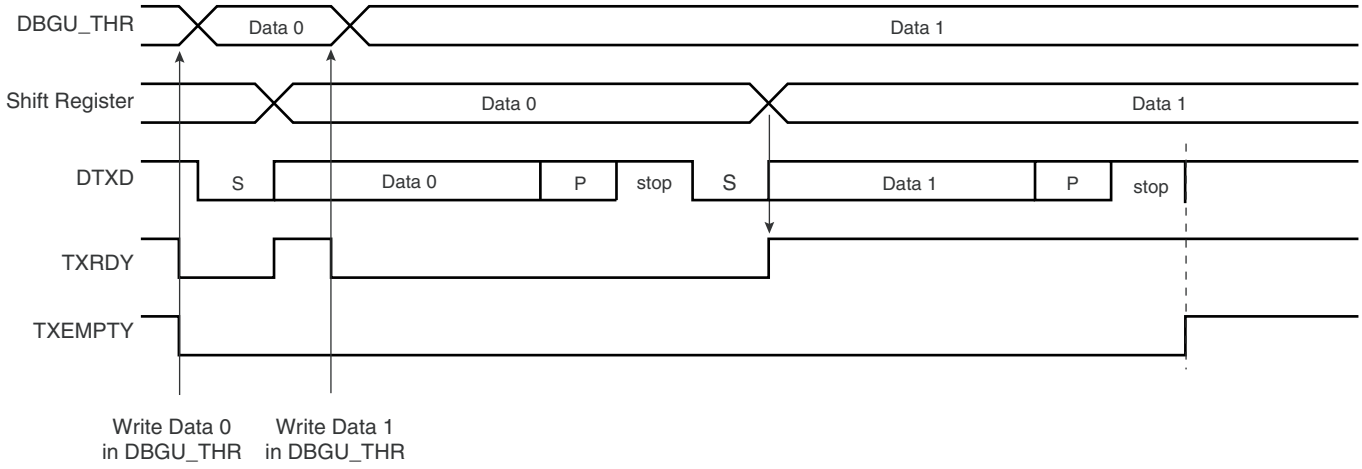


30.4.3.3 Transmitter Control

When the transmitter is enabled, the bit TXRDY (Transmitter Ready) is set in the status register DBGU_SR. The transmission starts when the programmer writes in the Transmit Holding Register DBGU_THR, and after the written character is transferred from DBGU_THR to the Shift Register. The bit TXRDY remains high until a second character is written in DBGU_THR. As soon as the first character is completed, the last character written in DBGU_THR is transferred into the shift register and TXRDY rises again, showing that the holding register is empty.

When both the Shift Register and the DBGU_THR are empty, i.e., all the characters written in DBGU_THR have been processed, the bit TXEMPTY rises after the last stop bit has been completed.

Figure 30-11. Transmitter Control



30.4.4 Peripheral Data Controller

Both the receiver and the transmitter of the Debug Unit's UART are generally connected to a Peripheral Data Controller (PDC) channel.

The peripheral data controller channels are programmed via registers that are mapped within the Debug Unit user interface from the offset 0x100. The status bits are reported in the Debug Unit status register DBGU_SR and can generate an interrupt.

The RXRDY bit triggers the PDC channel data transfer of the receiver. This results in a read of the data in DBGU_RHR. The TXRDY bit triggers the PDC channel data transfer of the transmitter. This results in a write of a data in DBGU_THR.

30.4.5 Test Modes

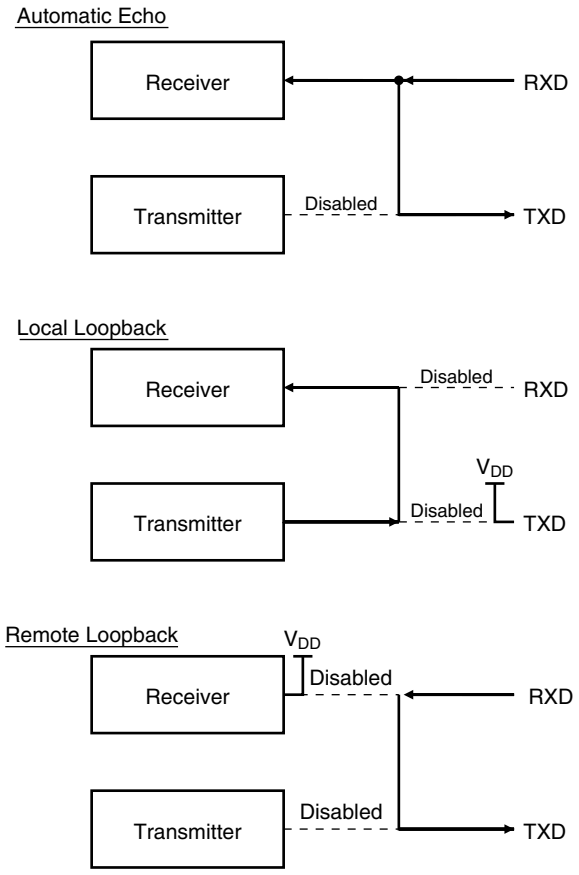
The Debug Unit supports three tests modes. These modes of operation are programmed by using the field CHMODE (Channel Mode) in the mode register DBGU_MR.

The Automatic Echo mode allows bit-by-bit retransmission. When a bit is received on the DRXD line, it is sent to the DTXD line. The transmitter operates normally, but has no effect on the DTXD line.

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

The Remote Loopback mode directly connects the DRXD pin to the DTXD line. The transmitter and the receiver are disabled and have no effect. This mode allows a bit-by-bit retransmission.

Figure 30-12. Test Modes



30.4.6 Debug Communication Channel Support

The Debug Unit handles the signals COMMRX and COMMTX that come from the Debug Communication Channel of the ARM Processor and are driven by the In-circuit Emulator.

The Debug Communication Channel contains two registers that are accessible through the ICE Breaker on the JTAG side and through the coprocessor 0 on the ARM Processor side.

As a reminder, the following instructions are used to read and write the Debug Communication Channel:

```
MRC    p14, 0, Rd, c1, c0, 0
```

Returns the debug communication data read register into Rd

```
MCR    p14, 0, Rd, c1, c0, 0
```

Writes the value in Rd to the debug communication data write register.

The bits COMMRX and COMMTX, which indicate, respectively, that the read register has been written by the debugger but not yet read by the processor, and that the write register has been written by the processor and not yet read by the debugger, are wired on the two highest bits of the status register DBGU_SR. These bits can generate an interrupt. This feature permits handling under interrupt a debug link between a debug monitor running on the target system and a debugger.

30.4.7 Chip Identifier

The Debug Unit features two chip identifier registers, DBGU_CIDR (Chip ID Register) and DBGU_EXID (Extension ID). Both registers contain a hard-wired value that is read-only. The first register contains the following fields:

- EXT - shows the use of the extension identifier register
- NVPTYP and NVPSIZ - identifies the type of embedded non-volatile memory and its size
- ARCH - identifies the set of embedded peripherals
- SRAMSIZ - indicates the size of the embedded SRAM
- EPROC - indicates the embedded ARM processor
- VERSION - gives the revision of the silicon

The second register is device-dependent and reads 0 if the bit EXT is 0.

30.4.8 ICE Access Prevention

The Debug Unit allows blockage of access to the system through the ARM processor's ICE interface. This feature is implemented via the register Force NTRST (DBGU_FNR), that allows assertion of the NTRST signal of the ICE Interface. Writing the bit FNTRST (Force NTRST) to 1 in this register prevents any activity on the TAP controller.

On standard devices, the bit FNTRST resets to 0 and thus does not prevent ICE access.

This feature is especially useful on custom ROM devices for customers who do not want their on-chip code to be visible.

30.5 Debug Unit (DBGU) User Interface

Table 30-2. 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
0x004C - 0x00FC	Reserved	–	–	–
0x0100 - 0x0124	PDC Area	–	–	–

30.5.1 Debug Unit Control Register

Name: DBGU_CR

Address: 0xFFFFEE00

Access Type: Write-only

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

- **RSTRX: Reset Receiver**

0 = No effect.

1 = The receiver logic is reset and disabled. If a character is being received, the reception is aborted.

- **RSTTX: Reset Transmitter**

0 = No effect.

1 = The transmitter logic is reset and disabled. If a character is being transmitted, the transmission is aborted.

- **RXEN: Receiver Enable**

0 = No effect.

1 = The receiver is enabled if RXDIS is 0.

- **RXDIS: Receiver Disable**

0 = No effect.

1 = The receiver is disabled. If a character is being processed and RSTRX is not set, the character is completed before the receiver is stopped.

- **TXEN: Transmitter Enable**

0 = No effect.

1 = The transmitter is enabled if TXDIS is 0.

- **TXDIS: Transmitter Disable**

0 = No effect.

1 = The transmitter is disabled. If a character is being processed and a character has been written the DBGU_THR and RSTTX is not set, both characters are completed before the transmitter is stopped.

- **RSTSTA: Reset Status Bits**

0 = No effect.

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

30.5.2 Debug Unit Mode Register

Name: DBGU_MR

Address: 0xFFFFEE04

Access Type: Read-write

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

- **PAR: Parity Type**

PAR			Parity Type
0	0	0	Even parity
0	0	1	Odd parity
0	1	0	Space: parity forced to 0
0	1	1	Mark: parity forced to 1
1	x	x	No parity

- **CHMODE: Channel Mode**

CHMODE		Mode Description
0	0	Normal Mode
0	1	Automatic Echo
1	0	Local Loopback
1	1	Remote Loopback

30.5.3 Debug Unit Interrupt Enable Register

Name: DBGU_IER

Address: 0xFFFFEE08

Access Type: Write-only

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

- **RXRDY: Enable RXRDY Interrupt**
- **TXRDY: Enable TXRDY Interrupt**
- **ENDRX: Enable End of Receive Transfer Interrupt**
- **ENDTX: Enable End of Transmit Interrupt**
- **OVRE: Enable Overrun Error Interrupt**
- **FRAME: Enable Framing Error Interrupt**
- **PARE: Enable Parity Error Interrupt**
- **TXEMPTY: Enable TXEMPTY Interrupt**
- **TXBUFE: Enable Buffer Empty Interrupt**
- **RXBUFF: Enable Buffer Full Interrupt**
- **COMMTX: Enable COMMTX (from ARM) Interrupt**
- **COMMRX: Enable COMMRX (from ARM) Interrupt**

0 = No effect.

1 = Enables the corresponding interrupt.

30.5.4 Debug Unit Interrupt Disable Register

Name: DBGU_IDR

Address: 0xFFFFEE0C

Access Type: Write-only

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

- **RXRDY: Disable RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **ENDRX: Disable End of Receive Transfer Interrupt**
- **ENDTX: Disable End of Transmit Interrupt**
- **OVRE: Disable Overrun Error Interrupt**
- **FRAME: Disable Framing Error Interrupt**
- **PARE: Disable Parity Error Interrupt**
- **TXEMPTY: Disable TXEMPTY Interrupt**
- **TXBUFE: Disable Buffer Empty Interrupt**
- **RXBUFF: Disable Buffer Full Interrupt**
- **COMMTX: Disable COMMTX (from ARM) Interrupt**
- **COMMRX: Disable COMMRX (from ARM) Interrupt**

0 = No effect.

1 = Disables the corresponding interrupt.

30.5.5 Debug Unit Interrupt Mask Register

Name: DBGU_IMR

Address: 0xFFFFEE10

Access Type: Read-only

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

- **RXRDY: Mask RXRDY Interrupt**
- **TXRDY: Disable TXRDY Interrupt**
- **ENDRX: Mask End of Receive Transfer Interrupt**
- **ENDTX: Mask End of Transmit Interrupt**
- **OVRE: Mask Overrun Error Interrupt**
- **FRAME: Mask Framing Error Interrupt**
- **PARE: Mask Parity Error Interrupt**
- **TXEMPTY: Mask TXEMPTY Interrupt**
- **TXBUFE: Mask TXBUFE Interrupt**
- **RXBUFF: Mask RXBUFF Interrupt**
- **COMMTX: Mask COMMTX Interrupt**
- **COMMRX: Mask COMMRX Interrupt**

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

30.5.6 Debug Unit Status Register

Name: DBGU_SR

Address: 0xFFFFEE14

Access Type: Read-only

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

- **RXRDY: Receiver Ready**

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

1 = At least one complete character has been received, transferred to DBGU_RHR and not yet read.

- **TXRDY: Transmitter Ready**

0 = A character has been written to DBGU_THR and not yet transferred to the Shift Register, or the transmitter is disabled.

1 = There is no character written to DBGU_THR not yet transferred to the Shift Register.

- **ENDRX: End of Receiver Transfer**

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

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

- **ENDTX: End of Transmitter Transfer**

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

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

- **OVRE: Overrun Error**

0 = No overrun error has occurred since the last RSTSTA.

1 = At least one overrun error has occurred since the last RSTSTA.

- **FRAME: Framing Error**

0 = No framing error has occurred since the last RSTSTA.

1 = At least one framing error has occurred since the last RSTSTA.

- **PARE: Parity Error**

0 = No parity error has occurred since the last RSTSTA.

1 = At least one parity error has occurred since the last RSTSTA.

- **TXEMPTY: Transmitter Empty**

0 = There are characters in DBGU_THR, or characters being processed by the transmitter, or the transmitter is disabled.

1 = There are no characters in DBGU_THR and there are no characters being processed by the transmitter.

- **TXBUFE: Transmission Buffer Empty**

0 = The buffer empty signal from the transmitter PDC channel is inactive.

1 = The buffer empty signal from the transmitter PDC channel is active.

- **RXBUFF: Receive Buffer Full**

0 = The buffer full signal from the receiver PDC channel is inactive.

1 = The buffer full signal from the receiver PDC channel is active.

- **COMMTX: Debug Communication Channel Write Status**

0 = COMMTX from the ARM processor is inactive.

1 = COMMTX from the ARM processor is active.

- **COMMRX: Debug Communication Channel Read Status**

0 = COMMRX from the ARM processor is inactive.

1 = COMMRX from the ARM processor is active.

30.5.7 Debug Unit Receiver Holding Register

Name: DBGU_RHR

Address: 0xFFFFEE18

Access Type: Read-only

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

- **RXCHR: Received Character**

Last received character if RXRDY is set.

30.5.8 Debug Unit Transmit Holding Register

Name: DBGU_THR

Address: 0xFFFFEE1C

Access Type: Write-only

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

- **TXCHR: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set.

30.5.9 Debug Unit Baud Rate Generator Register

Name: DBGU_BRGR

Address: 0xFFFFEE20

Access Type: Read-write

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

- **CD: Clock Divisor**

CD	Baud Rate Clock
0	Disabled
1	MCK
2 to 65535	MCK / (CD x 16)

30.5.10 Debug Unit Chip ID Register

Name: DBGU_CIDR

Address: 0xFFFFEE40

Access Type: Read-only

31	30	29	28	27	26	25	24
EXT	NVPTYP			ARCH			
23	22	21	20	19	18	17	16
ARCH				SRAMSIZ			
15	14	13	12	11	10	9	8
NVPSIZ2				NVPSIZ			
7	6	5	4	3	2	1	0
EPROC			VERSION				

- **VERSION: Version of the Device**

Current 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.

30.5.11 Debug Unit Chip ID Extension Register

Name: DBGU_EXID

Address: 0xFFFFEE44

Access Type: Read-only

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

- **EXID: Chip ID Extension**

Reads 0 if the bit EXT in DBGU_CIDR is 0.

30.5.12 Debug Unit Force NTRST Register

Name: DBGU_FNR

Address: 0xFFFFEE48

Access Type: Read-write

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

- **FNTRST: Force NTRST**

0 = NTRST of the ARM processor's TAP controller is driven by the power_on_reset signal.

1 = NTRST of the ARM processor's TAP controller is held low.

31. Parallel Input/Output Controller (PIO)

31.1 Overview

The Parallel Input/Output Controller (PIO) manages up to 32 fully programmable input/output lines. Each I/O line may be dedicated as a general-purpose I/O or be assigned to a function of an embedded peripheral. This assures effective optimization of the pins of a product.

Each I/O line is associated with a bit number in all of the 32-bit registers of the 32-bit wide User Interface.

Each I/O line of the PIO Controller features:

- An input change interrupt enabling level change detection on any I/O line.
- A glitch filter providing rejection of pulses lower than one-half of clock cycle.
- Multi-drive capability similar to an open drain I/O line.
- Control of the pull-up of the I/O line.
- Input visibility and output control.

The PIO Controller also features a synchronous output providing up to 32 bits of data output in a single write operation.

31.2 Block Diagram

Figure 31-1. Block Diagram

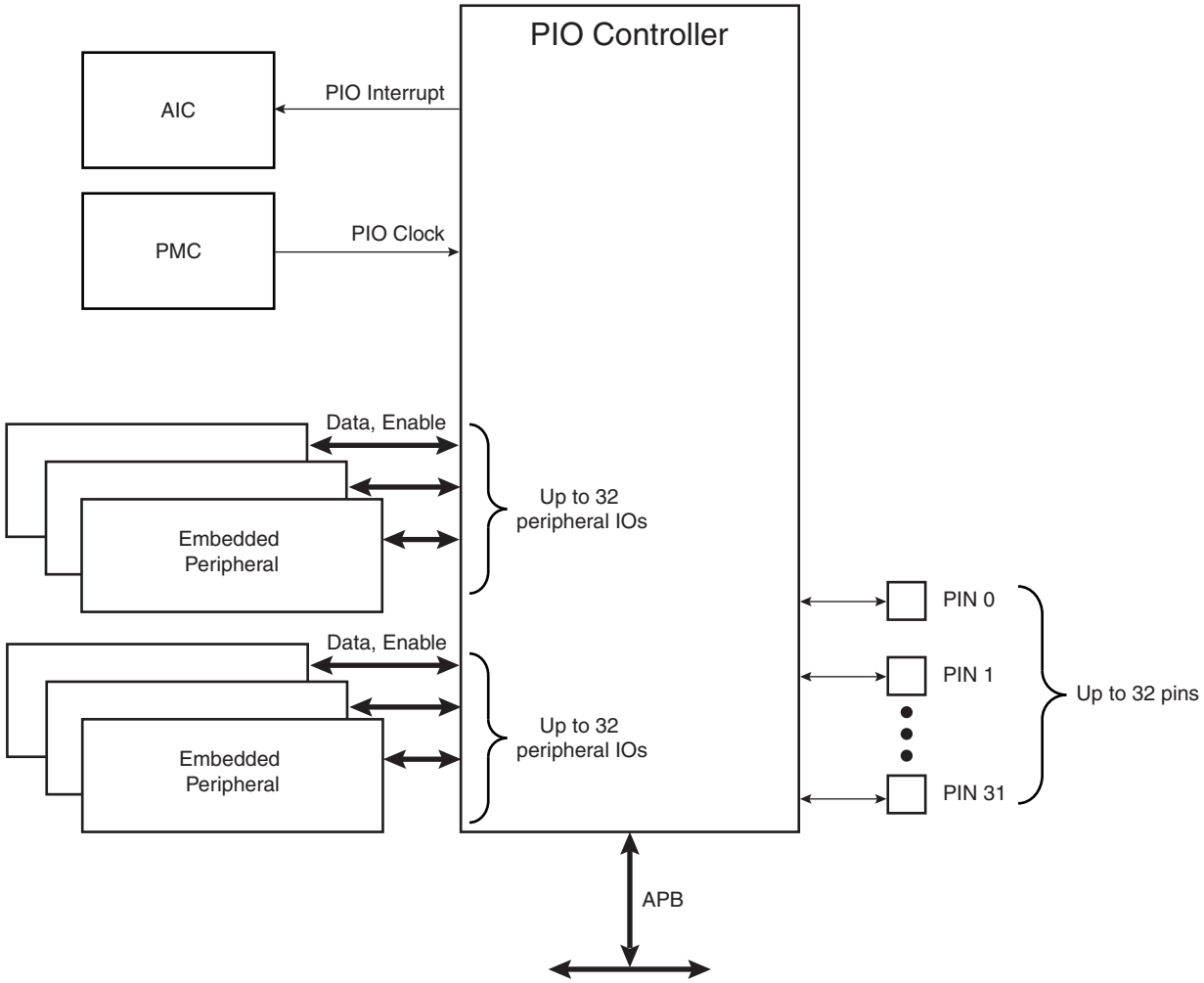
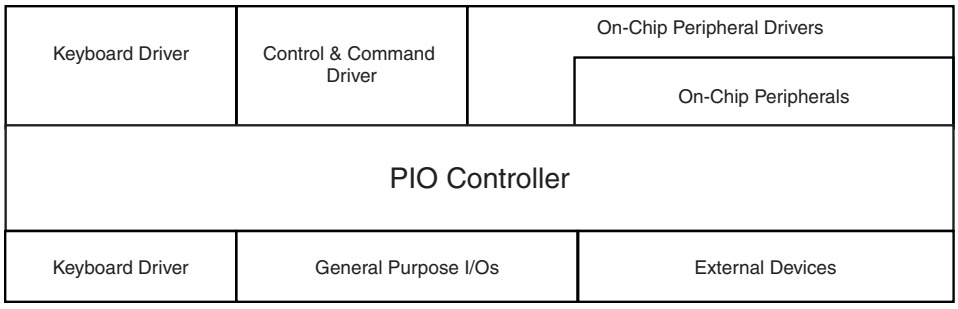


Figure 31-2. Application Block Diagram



31.3 Product Dependencies

31.3.1 Pin Multiplexing

Each pin is configurable, according to product definition as either a general-purpose I/O line only, or as an I/O line multiplexed with one or two peripheral I/Os. As the multiplexing is hardware-defined and thus product-dependent, the hardware designer and programmer must carefully determine the configuration of the PIO controllers required by their application. When an I/O line is general-purpose only, i.e. not multiplexed with any peripheral I/O, programming of the PIO Controller regarding the assignment to a peripheral has no effect and only the PIO Controller can control how the pin is driven by the product.

31.3.2 External Interrupt Lines

The interrupt signals FIQ and IRQ0 to IRQn are most generally multiplexed through the PIO Controllers. However, it is not necessary to assign the I/O line to the interrupt function as the PIO Controller has no effect on inputs and the interrupt lines (FIQ or IRQs) are used only as inputs.

31.3.3 Power Management

The Power Management Controller controls the PIO Controller clock in order to save power. Writing any of the registers of the user interface does not require the PIO Controller clock to be enabled. This means that the configuration of the I/O lines does not require the PIO Controller clock to be enabled.

However, when the clock is disabled, not all of the features of the PIO Controller are available. Note that the Input Change Interrupt and the read of the pin level require the clock to be validated.

After a hardware reset, the PIO clock is disabled by default.

The user must configure the Power Management Controller before any access to the input line information.

31.3.4 Interrupt Generation

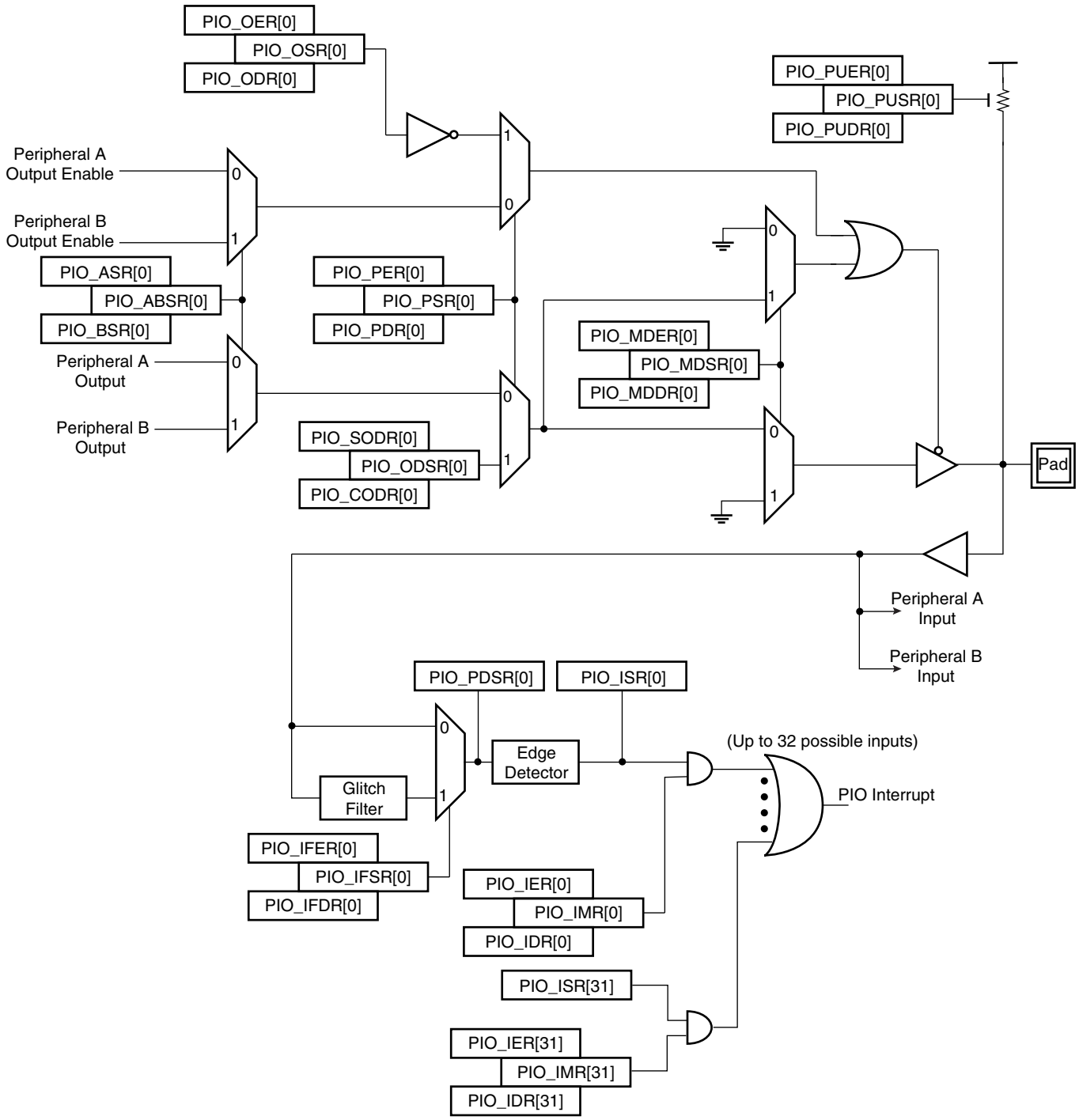
For interrupt handling, the PIO Controllers are considered as user peripherals. This means that the PIO Controller interrupt lines are connected among the interrupt sources 2 to 31. Refer to the PIO Controller peripheral identifier in the product description to identify the interrupt sources dedicated to the PIO Controllers.

The PIO Controller interrupt can be generated only if the PIO Controller clock is enabled.

31.4 Functional Description

The PIO Controller features up to 32 fully-programmable I/O lines. Most of the control logic associated to each I/O is represented in Figure 31-3. In this description each signal shown represents but one of up to 32 possible indexes.

Figure 31-3. I/O Line Control Logic



31.4.1 Pull-up Resistor Control

Each I/O line is designed with an embedded pull-up resistor. The pull-up resistor can be enabled or disabled by writing respectively PIO_PUER (Pull-up Enable Register) and PIO_PUDR (Pull-up Disable 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.

31.4.2 I/O Line or Peripheral Function Selection

When a pin is multiplexed with one or two peripheral functions, the selection is controlled with the registers PIO_PER (PIO Enable Register) and PIO_PDR (PIO Disable Register). The register PIO_PSR (PIO Status Register) is the result of the set and clear registers and indicates whether the pin is controlled by the corresponding peripheral or by the PIO Controller. A value of 0 indicates that the pin is controlled by the corresponding on-chip peripheral selected in the PIO_ABSR (AB Select Status Register). A value of 1 indicates the pin is controlled by the PIO controller.

If a pin is used as a general purpose I/O line (not multiplexed with an on-chip peripheral), PIO_PER and PIO_PDR have no effect and PIO_PSR returns 1 for the corresponding bit.

After reset, most generally, the I/O lines are controlled by the PIO controller, i.e. PIO_PSR resets at 1. However, in some events, it is important that PIO lines are controlled by the peripheral (as in the case of memory chip select lines that must be driven inactive after reset or for address lines that must be driven low for booting out of an external memory). Thus, the reset value of PIO_PSR is defined at the product level, depending on the multiplexing of the device.

31.4.3 Peripheral A or B Selection

The PIO Controller provides multiplexing of up to two peripheral functions on a single pin. The selection is performed by writing PIO_ASR (A Select Register) and PIO_BSR (Select B Register). PIO_ABSR (AB Select Status Register) indicates which peripheral line is currently selected. For each pin, the corresponding bit at level 0 means peripheral A is selected whereas the corresponding bit at level 1 indicates that peripheral B is selected.

Note that multiplexing of peripheral lines A and B only affects the output line. The peripheral input lines are always connected to the pin input.

After reset, PIO_ABSR is 0, thus indicating that all the PIO lines are configured on peripheral A. However, peripheral A generally does not drive the pin as the PIO Controller resets in I/O line mode.

Writing in PIO_ASR and PIO_BSR manages PIO_ABSR regardless of the configuration of the pin. However, assignment of a pin to a peripheral function requires a write in the corresponding peripheral selection register (PIO_ASR or PIO_BSR) in addition to a write in PIO_PDR.

31.4.4 Output Control

When the I/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.

31.4.5 Synchronous Data Output

Controlling all parallel busses using several PIOs requires two successive write operations in the PIO_SODR and PIO_CODR registers. This may lead to unexpected transient values. The PIO controller offers a direct control of PIO outputs by single write access to PIO_ODSR (Output Data Status Register). Only bits unmasked by PIO_OWSR (Output Write Status Register) are written. The mask bits in the PIO_OWSR are set by writing to PIO_OWER (Output Write Enable Register) and cleared by writing to PIO_OWDR (Output Write Disable Register).

After reset, the synchronous data output is disabled on all the I/O lines as PIO_OWSR resets at 0x0.

31.4.6 Multi Drive Control (Open Drain)

Each I/O can be independently programmed in Open Drain by using the Multi Drive feature. This feature permits several drivers to be connected on the I/O line which is driven low only by each device. An external pull-up resistor (or enabling of the internal one) is generally required to guarantee a high level on the line.

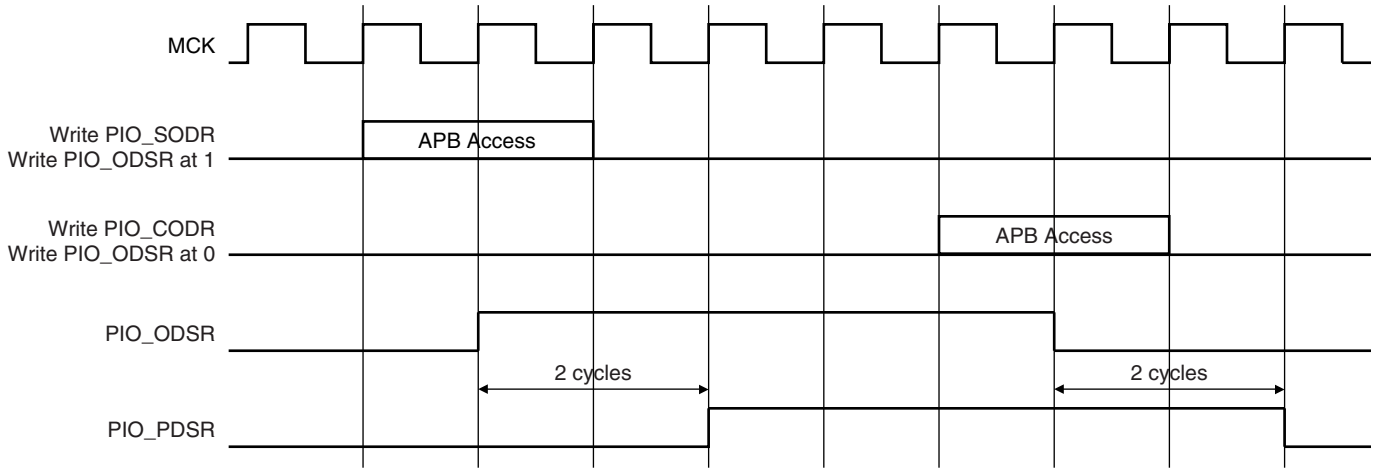
The Multi Drive feature is controlled by PIO_MDER (Multi-driver Enable Register) and PIO_MDDR (Multi-driver Disable Register). The Multi Drive can be selected whether the I/O line is controlled by the PIO controller or assigned to a peripheral function. PIO_MDSR (Multi-driver Status Register) indicates the pins that are configured to support external drivers.

After reset, the Multi Drive feature is disabled on all pins, i.e. PIO_MDSR resets at value 0x0.

31.4.7 Output Line Timings

Figure 31-4 shows how the outputs are driven either by writing PIO_SODR or PIO_CODR, or by directly writing PIO_ODSR. This last case is valid only if the corresponding bit in PIO_OWSR is set. Figure 31-4 also shows when the feedback in PIO_PDSR is available.

Figure 31-4. Output Line Timings



31.4.8 Inputs

The level on each I/O line can be read through PIO_PDSR (Pin Data Status Register). This register indicates the level of the I/O lines regardless of their configuration, whether uniquely as an input or driven by the PIO controller or driven by a peripheral.

Reading the I/O line levels requires the clock of the PIO controller to be enabled, otherwise PIO_PDSR reads the levels present on the I/O line at the time the clock was disabled.

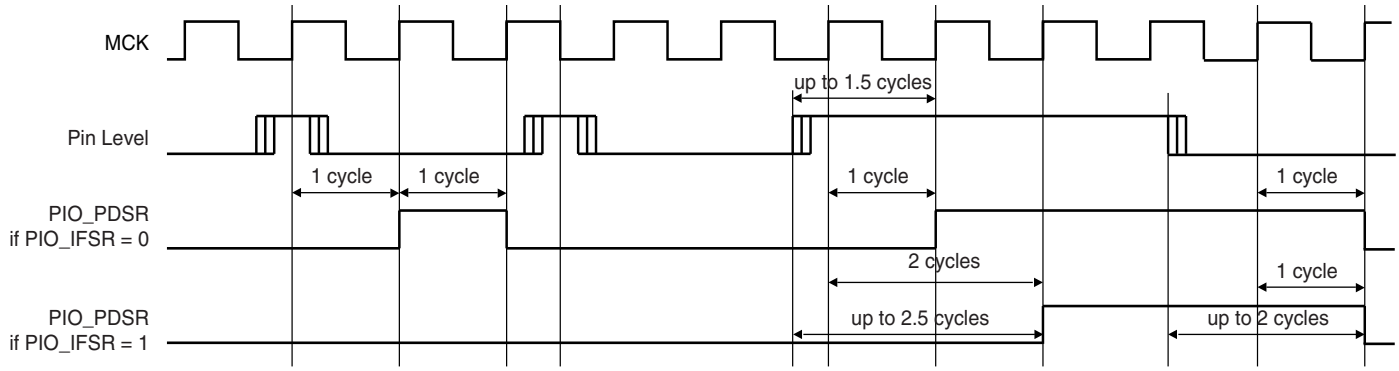
31.4.9 Input Glitch Filtering

Optional input glitch filters are independently programmable on each I/O line. When the glitch filter is enabled, a glitch with a duration of less than 1/2 Master Clock (MCK) cycle is automatically rejected, while a pulse with a duration of 1 Master Clock cycle or more is accepted. For pulse durations between 1/2 Master Clock cycle and 1 Master Clock cycle the pulse may or may not be taken into account, depending on the precise timing of its occurrence. Thus for a pulse to be visible it must exceed 1 Master Clock cycle, whereas for a glitch to be reliably filtered out, its duration must not exceed 1/2 Master Clock cycle. The filter introduces one Master Clock cycle latency if the pin level change occurs before a rising edge. However, this latency does not appear if the pin level change occurs before a falling edge. This is illustrated in Figure 31-5.

The glitch filters are controlled by the register set; PIO_IFER (Input Filter Enable Register), PIO_IFDR (Input Filter Disable Register) and PIO_IFSR (Input Filter Status Register). Writing PIO_IFER and PIO_IFDR respectively sets and clears bits in PIO_IFSR. This last register enables the glitch filter on the I/O lines.

When the glitch filter is enabled, it does not modify the behavior of the inputs on the peripherals. It acts only on the value read in PIO_PDSR and on the input change interrupt detection. The glitch filters require that the PIO Controller clock is enabled.

Figure 31-5. Input Glitch Filter Timing



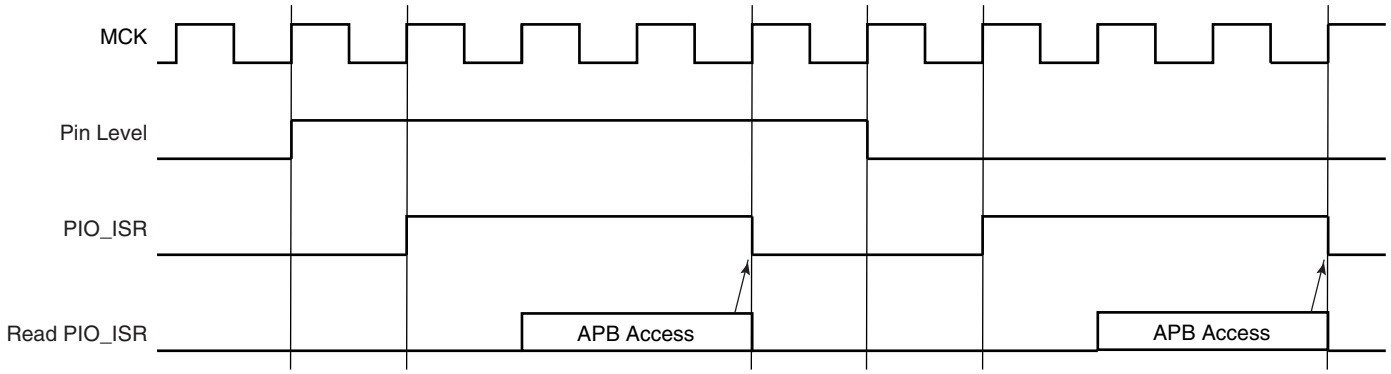
31.4.10 Input Change Interrupt

The PIO Controller can be programmed to generate an interrupt when it detects an input change on an I/O line. The Input Change Interrupt is controlled by writing PIO_IER (Interrupt Enable Register) and PIO_IDR (Interrupt Disable Register), which respectively enable and disable the input change interrupt by setting and clearing the corresponding bit in PIO_IMR (Interrupt Mask Register). As Input change detection is possible only by comparing two successive samplings of the input of the I/O line, the PIO Controller clock must be enabled. The Input Change Interrupt is available, regardless of the configuration of the I/O line, i.e. configured as an input only, controlled by the PIO Controller or assigned to a peripheral function.

When an input change is detected on an I/O line, the corresponding bit in PIO_ISR (Interrupt Status Register) is set. If the corresponding bit in PIO_IMR is set, the PIO Controller interrupt line is asserted. The interrupt signals of the thirty-two channels are ORed-wired together to generate a single interrupt signal to the Advanced Interrupt Controller.

When the software reads PIO_ISR, all the interrupts are automatically cleared. This signifies that all the interrupts that are pending when PIO_ISR is read must be handled.

Figure 31-6. Input Change Interrupt Timings



31.5 I/O Lines Programming Example

The programing example as shown in Table 31-1 below is used to define the following configuration.

- 4-bit output port on I/O lines 0 to 3, (should be written in a single write operation), open-drain, with pull-up resistor
- Four output signals on I/O lines 4 to 7 (to drive LEDs for example), driven high and low, no pull-up resistor
- Four input signals on I/O lines 8 to 11 (to read push-button states for example), with pull-up resistors, glitch filters and input change interrupts
- Four input signals on I/O line 12 to 15 to read an external device status (polled, thus no input change interrupt), no pull-up resistor, no glitch filter
- I/O lines 16 to 19 assigned to peripheral A functions with pull-up resistor
- I/O lines 20 to 23 assigned to peripheral B functions, no pull-up resistor
- I/O line 24 to 27 assigned to peripheral A with Input Change Interrupt and pull-up resistor

Table 31-1. Programming Example

Register	Value to be Written
PIO_PER	0x0000 FFFF
PIO_PDR	0x0FFF 0000
PIO_OER	0x0000 00FF
PIO_ODR	0x0FFF FF00
PIO_IFER	0x0000 0F00
PIO_IFDR	0x0FFF F0FF
PIO_SODR	0x0000 0000
PIO_CODR	0x0FFF FFFF
PIO_IER	0x0F00 0F00
PIO_IDR	0x00FF F0FF
PIO_MDER	0x0000 000F
PIO_MDDR	0x0FFF FFF0
PIO_PUDR	0x00F0 00F0
PIO_PUER	0x0F0F FF0F
PIO_ASR	0x0F0F 0000
PIO_BSR	0x00F0 0000
PIO_OWER	0x0000 000F
PIO_OWDR	0x0FFF FFF0

31.6 Parallel Input/Output 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 31-2. 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 31-2. 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.

31.6.1 PIO Controller PIO Enable Register

Name: PIO_PER

Addresses: 0xFFFFF200 (PIOA), 0xFFFFF400 (PIOB), 0xFFFFF600 (PIOC), 0xFFFFF800 (PIOD), 0xFFFFFA00 (PIOE)

Access Type: Write-only

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

- **P0-P31: PIO Enable**

0 = No effect.

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

31.6.2 PIO Controller PIO Disable Register

Name: PIO_PDR

Addresses: 0xFFFFF204 (PIOA), 0xFFFFF404 (PIOB), 0xFFFFF604 (PIOC), 0xFFFFF804 (PIOD), 0xFFFFFA04 (PIOE)

Access Type: Write-only

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

- **P0-P31: PIO Disable**

0 = No effect.

1 = Disables the PIO from controlling the corresponding pin (enables peripheral control of the pin).

31.6.3 PIO Controller PIO Status Register

Name: PIO_PSR

Addresses: 0xFFFFF208 (PIOA), 0xFFFFF408 (PIOB), 0xFFFFF608 (PIOC), 0xFFFFF808 (PIOD), 0xFFFFFA08 (PIOE)

Access Type: Read-only

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

- **P0-P31: PIO Status**

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

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

31.6.4 PIO Controller Output Enable Register

Name: PIO_OER

Addresses: 0xFFFFF210 (PIOA), 0xFFFFF410 (PIOB), 0xFFFFF610 (PIOC), 0xFFFFF810 (PIOD), 0xFFFFFA10 (PIOE)

Access Type: Write-only

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

- **P0-P31: Output Enable**

0 = No effect.

1 = Enables the output on the I/O line.

31.6.5 PIO Controller Output Disable Register

Name: PIO_ODR

Addresses: 0xFFFFF214 (PIOA), 0xFFFFF414 (PIOB), 0xFFFFF614 (PIOC), 0xFFFFF814 (PIOD), 0xFFFFFA14 (PIOE)

Access Type: Write-only

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

- **P0-P31: Output Disable**

0 = No effect.

1 = Disables the output on the I/O line.

31.6.6 PIO Controller Output Status Register

Name: PIO_OSR

Addresses: 0xFFFFF218 (PIOA), 0xFFFFF418 (PIOB), 0xFFFFF618 (PIOC), 0xFFFFF818 (PIOD), 0xFFFFFA18 (PIOE)

Access Type: Read-only

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

- **P0-P31: Output Status**

0 = The I/O line is a pure input.

1 = The I/O line is enabled in output.

31.6.7 PIO Controller Input Filter Enable Register

Name: PIO_IFER

Addresses: 0xFFFFF220 (PIOA), 0xFFFFF420 (PIOB), 0xFFFFF620 (PIOC), 0xFFFFF820 (PIOD), 0xFFFFFA20 (PIOE)

Access Type: Write-only

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

- **P0-P31: Input Filter Enable**

0 = No effect.

1 = Enables the input glitch filter on the I/O line.

31.6.8 PIO Controller Input Filter Disable Register

Name: PIO_IFDR

Addresses: 0xFFFFF224 (PIOA), 0xFFFFF424 (PIOB), 0xFFFFF624 (PIOC), 0xFFFFF824 (PIOD), 0xFFFFFA24 (PIOE)

Access Type: Write-only

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

- **P0-P31: Input Filter Disable**

0 = No effect.

1 = Disables the input glitch filter on the I/O line.

31.6.9 PIO Controller Input Filter Status Register

Name: PIO_IFSR

Addresses: 0xFFFFF228 (PIOA), 0xFFFFF428 (PIOB), 0xFFFFF628 (PIOC), 0xFFFFF828 (PIOD), 0xFFFFFA28 (PIOE)

Access Type: Read-only

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

- **P0-P31: Input Filer Status**

0 = The input glitch filter is disabled on the I/O line.

1 = The input glitch filter is enabled on the I/O line.

31.6.10 PIO Controller Set Output Data Register

Name: PIO_SODR

Addresses: 0xFFFFF230 (PIOA), 0xFFFFF430 (PIOB), 0xFFFFF630 (PIOC), 0xFFFFF830 (PIOD), 0xFFFFFA30 (PIOE)

Access Type: Write-only

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

- **P0-P31: Set Output Data**

0 = No effect.

1 = Sets the data to be driven on the I/O line.

31.6.11 PIO Controller Clear Output Data Register

Name: PIO_CODR

Addresses: 0xFFFFF234 (PIOA), 0xFFFFF434 (PIOB), 0xFFFFF634 (PIOC), 0xFFFFF834 (PIOD), 0xFFFFFA34 (PIOE)

Access Type: Write-only

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

- **P0-P31: Set Output Data**

0 = No effect.

1 = Clears the data to be driven on the I/O line.

31.6.12 PIO Controller Output Data Status Register

Name: PIO_ODSR

Addresses: 0xFFFFF238 (PIOA), 0xFFFFF438 (PIOB), 0xFFFFF638 (PIOC), 0xFFFFF838 (PIOD), 0xFFFFFA38 (PIOE)

Access Type: Read-only or Read-write

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

- **P0-P31: Output Data Status**

0 = The data to be driven on the I/O line is 0.

1 = The data to be driven on the I/O line is 1.

31.6.13 PIO Controller Pin Data Status Register

Name: PIO_PDSR

Addresses: 0xFFFFF23C (PIOA), 0xFFFFF43C (PIOB), 0xFFFFF63C (PIOC), 0xFFFFF83C (PIOD), 0xFFFFFA3C (PIOE)

Access Type: Read-only

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

- **P0-P31: Output Data Status**

0 = The I/O line is at level 0.

1 = The I/O line is at level 1.

31.6.14 PIO Controller Interrupt Enable Register

Name: PIO_IER

Addresses: 0xFFFFF240 (PIOA), 0xFFFFF440 (PIOB), 0xFFFFF640 (PIOC), 0xFFFFF840 (PIOD), 0xFFFFFA40 (PIOE)

Access Type: Write-only

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

- **P0-P31: Input Change Interrupt Enable**

0 = No effect.

1 = Enables the Input Change Interrupt on the I/O line.

31.6.15 PIO Controller Interrupt Disable Register

Name: PIO_IDR

Addresses: 0xFFFFF244 (PIOA), 0xFFFFF444 (PIOB), 0xFFFFF644 (PIOC), 0xFFFFF844 (PIOD), 0xFFFFFA44 (PIOE)

Access Type: Write-only

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

- **P0-P31: Input Change Interrupt Disable**

0 = No effect.

1 = Disables the Input Change Interrupt on the I/O line.

31.6.16 PIO Controller Interrupt Mask Register

Name: PIO_IMR

Addresses: 0xFFFFF248 (PIOA), 0xFFFFF448 (PIOB), 0xFFFFF648 (PIOC), 0xFFFFF848 (PIOD), 0xFFFFFA48 (PIOE)

Access Type: Read-only

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

- **P0-P31: Input Change Interrupt Mask**

0 = Input Change Interrupt is disabled on the I/O line.

1 = Input Change Interrupt is enabled on the I/O line.

31.6.17 PIO Controller Interrupt Status Register

Name: PIO_ISR

Addresses: 0xFFFFF24C (PIOA), 0xFFFFF44C (PIOB), 0xFFFFF64C (PIOC), 0xFFFFF84C (PIOD), 0xFFFFFA4C (PIOE)

Access Type: Read-only

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

- **P0-P31: Input Change Interrupt Status**

0 = No Input Change has been detected on the I/O line since PIO_ISR was last read or since reset.

1 = At least one Input Change has been detected on the I/O line since PIO_ISR was last read or since reset.

31.6.18 PIO Multi-driver Enable Register

Name: PIO_MDER

Addresses: 0xFFFFF250 (PIOA), 0xFFFFF450 (PIOB), 0xFFFFF650 (PIOC), 0xFFFFF850 (PIOD), 0xFFFFFA50 (PIOE)

Access Type: Write-only

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

- **P0-P31: Multi Drive Enable.**

0 = No effect.

1 = Enables Multi Drive on the I/O line.

31.6.19 PIO Multi-driver Disable Register

Name: PIO_MDDR

Addresses: 0xFFFFF254 (PIOA), 0xFFFFF454 (PIOB), 0xFFFFF654 (PIOC), 0xFFFFF854 (PIOD), 0xFFFFFA54 (PIOE)

Access Type: Write-only

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

- **P0-P31: Multi Drive Disable.**

0 = No effect.

1 = Disables Multi Drive on the I/O line.

31.6.20 PIO Multi-driver Status Register

Name: PIO_MDSR

Addresses: 0xFFFFF258 (PIOA), 0xFFFFF458 (PIOB), 0xFFFFF658 (PIOC), 0xFFFFF858 (PIOD), 0xFFFFFA58 (PIOE)

Access Type: Read-only

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

- **P0-P31: Multi Drive Status.**

0 = The Multi Drive is disabled on the I/O line. The pin is driven at high and low level.

1 = The Multi Drive is enabled on the I/O line. The pin is driven at low level only.

31.6.21 PIO Pull Up Disable Register

Name: PIO_PUDR

Addresses: 0xFFFFF260 (PIOA), 0xFFFFF460 (PIOB), 0xFFFFF660 (PIOC), 0xFFFFF860 (PIOD), 0xFFFFFA60 (PIOE)

Access Type: Write-only

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

- **P0-P31: Pull Up Disable.**

0 = No effect.

1 = Disables the pull up resistor on the I/O line.

31.6.22 PIO Pull Up Enable Register

Name: PIO_PUER

Addresses: 0xFFFFF264 (PIOA), 0xFFFFF464 (PIOB), 0xFFFFF664 (PIOC), 0xFFFFF864 (PIOD), 0xFFFFFA64 (PIOE)

Access Type: Write-only

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

- **P0-P31: Pull Up Enable.**

0 = No effect.

1 = Enables the pull up resistor on the I/O line.

31.6.23 PIO Pull Up Status Register

Name: PIO_PUSR

Addresses: 0xFFFFF268 (PIOA), 0xFFFFF468 (PIOB), 0xFFFFF668 (PIOC), 0xFFFFF868 (PIOD), 0xFFFFFA68 (PIOE)

Access Type: Read-only

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

- **P0-P31: Pull Up Status.**

0 = Pull Up resistor is enabled on the I/O line.

1 = Pull Up resistor is disabled on the I/O line.

31.6.24 PIO Peripheral A Select Register

Name: PIO_ASR

Addresses: 0xFFFFF270 (PIOA), 0xFFFFF470 (PIOB), 0xFFFFF670 (PIOC), 0xFFFFF870 (PIOD), 0xFFFFFA70 (PIOE)

Access Type: Write-only

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

- **P0-P31: Peripheral A Select.**

0 = No effect.

1 = Assigns the I/O line to the Peripheral A function.

31.6.25 PIO Peripheral B Select Register

Name: PIO_BSR

Addresses: 0xFFFFF274 (PIOA), 0xFFFFF474 (PIOB), 0xFFFFF674 (PIOC), 0xFFFFF874 (PIOD), 0xFFFFFA74 (PIOE)

Access Type: Write-only

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

- **P0-P31: Peripheral B Select.**

0 = No effect.

1 = Assigns the I/O line to the peripheral B function.

31.6.26 PIO Peripheral A B Status Register

Name: PIO_ABSR

Addresses: 0xFFFFF278 (PIOA), 0xFFFFF478 (PIOB), 0xFFFFF678 (PIOC), 0xFFFFF878 (PIOD), 0xFFFFFA78 (PIOE)

Access Type: Read-only

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

- **P0-P31: Peripheral A B Status.**

0 = The I/O line is assigned to the Peripheral A.

1 = The I/O line is assigned to the Peripheral B.

31.6.27 PIO Output Write Enable Register

Name: PIO_OWER

Addresses: 0xFFFFF2A0 (PIOA), 0xFFFFF4A0 (PIOB), 0xFFFFF6A0 (PIOC), 0xFFFFF8A0 (PIOD), 0xFFFFFAA0 (PIOE)

Access Type: Write-only

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

- **P0-P31: Output Write Enable.**

0 = No effect.

1 = Enables writing PIO_ODSR for the I/O line.

31.6.28 PIO Output Write Disable Register

Name: PIO_OWDR

Addresses: 0xFFFFF2A4 (PIOA), 0xFFFFF4A4 (PIOB), 0xFFFFF6A4 (PIOC), 0xFFFFF8A4 (PIOD), 0xFFFFFAA4 (PIOE)

Access Type: Write-only

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

- **P0-P31: Output Write Disable.**

0 = No effect.

1 = Disables writing PIO_ODSR for the I/O line.

31.6.29 PIO Output Write Status Register

Name: PIO_OWSR

Addresses: 0xFFFFF2A8 (PIOA), 0xFFFFF4A8 (PIOB), 0xFFFFF6A8 (PIOC), 0xFFFFF8A8 (PIOD), 0xFFFFFAA8 (PIOE)

Access Type: Read-only

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

- **P0-P31: Output Write Status.**

0 = Writing PIO_ODSR does not affect the I/O line.

1 = Writing PIO_ODSR affects the I/O line.

32. Serial Peripheral Interface (SPI)

32.1 Overview

The Serial Peripheral Interface (SPI) circuit is a synchronous serial data link that provides communication with external devices in Master or Slave Mode. It also enables communication between processors if an external processor is connected to the system.

The Serial Peripheral Interface is essentially a shift register that serially transmits data bits to other SPIs. During a data transfer, one SPI system acts as the “master”' which controls the data flow, while the other devices act as “slaves'' which have data shifted into and out by the master. Different CPUs can take turn being masters (Multiple Master Protocol opposite to Single Master Protocol where one CPU is always the master while all of the others are always slaves) and one master may simultaneously shift data into multiple slaves. However, only one slave may drive its output to write data back to the master at any given time.

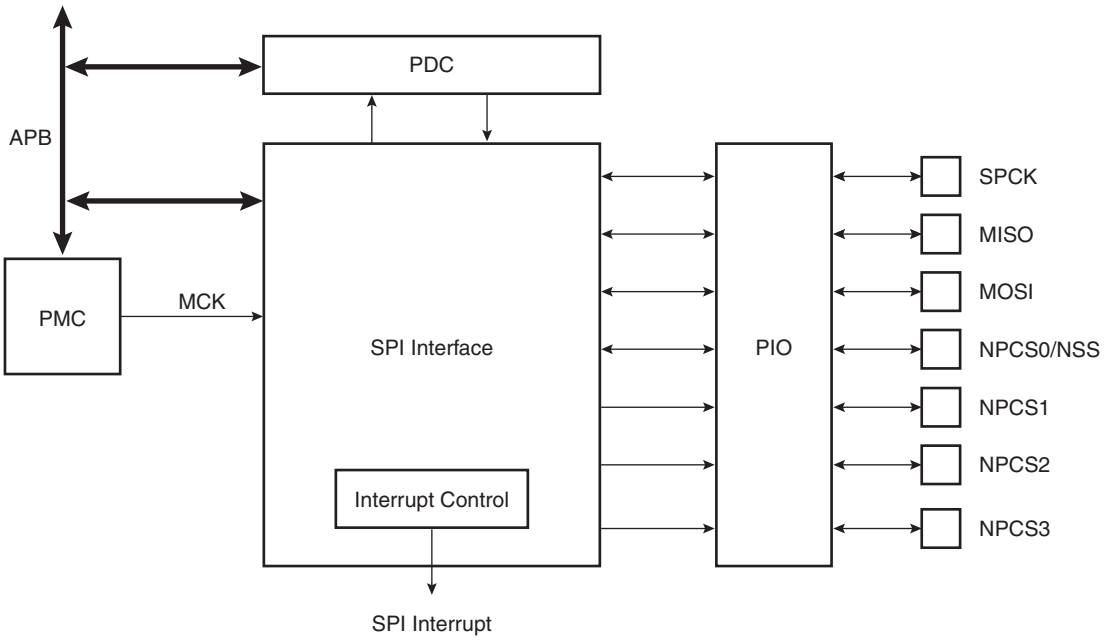
A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

The SPI system consists of two data lines and two control lines:

- Master Out Slave In (MOSI): This data line supplies the output data from the master shifted into the input(s) of the slave(s).
- Master In Slave Out (MISO): This data line supplies the output data from a slave to the input of the master. There may be no more than one slave transmitting data during any particular transfer.
- Serial Clock (SPCK): This control line is driven by the master and regulates the flow of the data bits. The master may transmit data at a variety of baud rates; the SPCK line cycles once for each bit that is transmitted.
- Slave Select (NSS): This control line allows slaves to be turned on and off by hardware.

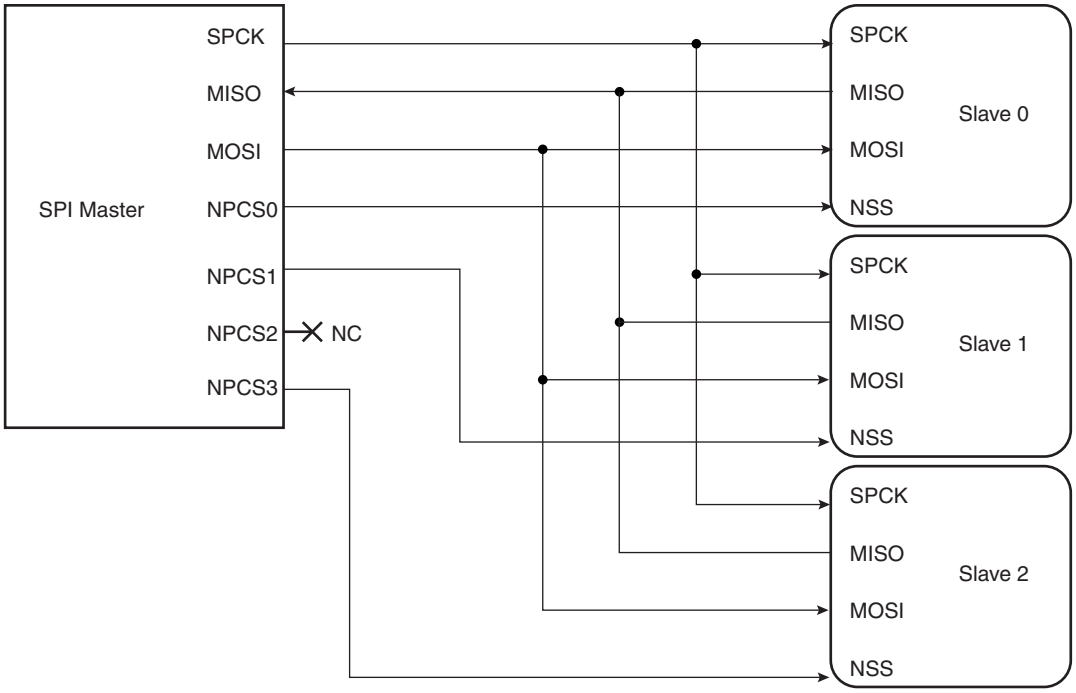
32.2 Block Diagram

Figure 32-1. Block Diagram



32.3 Application Block Diagram

Figure 32-2. Application Block Diagram: Single Master/Multiple Slave Implementation



32.4 Signal Description

Table 32-1. Signal Description

Pin Name	Pin Description	Type	
		Master	Slave
MISO	Master In Slave Out	Input	Output
MOSI	Master Out Slave In	Output	Input
SPCK	Serial Clock	Output	Input
NPCS1-NPCS3	Peripheral Chip Selects	Output	Unused
NPCS0/NSS	Peripheral Chip Select/Slave Select	Output	Input

32.5 Product Dependencies

32.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the SPI pins to their peripheral functions.

32.5.2 Power Management

The SPI may be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the SPI clock.

32.5.3 Interrupt

The SPI interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling the SPI interrupt requires programming the AIC before configuring the SPI.

32.6 Functional Description

32.6.1 Modes of Operation

The SPI operates in Master Mode or in Slave Mode.

Operation in Master Mode is programmed by writing at 1 the MSTR bit in the Mode Register. The pins NPCS0 to NPCS3 are all configured as outputs, the SPCK pin is driven, the MISO line is wired on the receiver input and the MOSI line driven as an output by the transmitter.

If the MSTR bit is written at 0, the SPI operates in Slave Mode. The MISO line is driven by the transmitter output, the MOSI line is wired on the receiver input, the SPCK pin is driven by the transmitter to synchronize the receiver. The NPCS0 pin becomes an input, and is used as a Slave Select signal (NSS). The pins NPCS1 to NPCS3 are not driven and can be used for other purposes.

The data transfers are identically programmable for both modes of operations. The baud rate generator is activated only in Master Mode.

32.6.2 Data Transfer

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the Chip Select Register. The clock phase is programmed with the NCPHA bit. These two parameters determine the edges of the clock signal on which data is driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations that are incompatible with one another. Thus, a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are used and fixed in different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

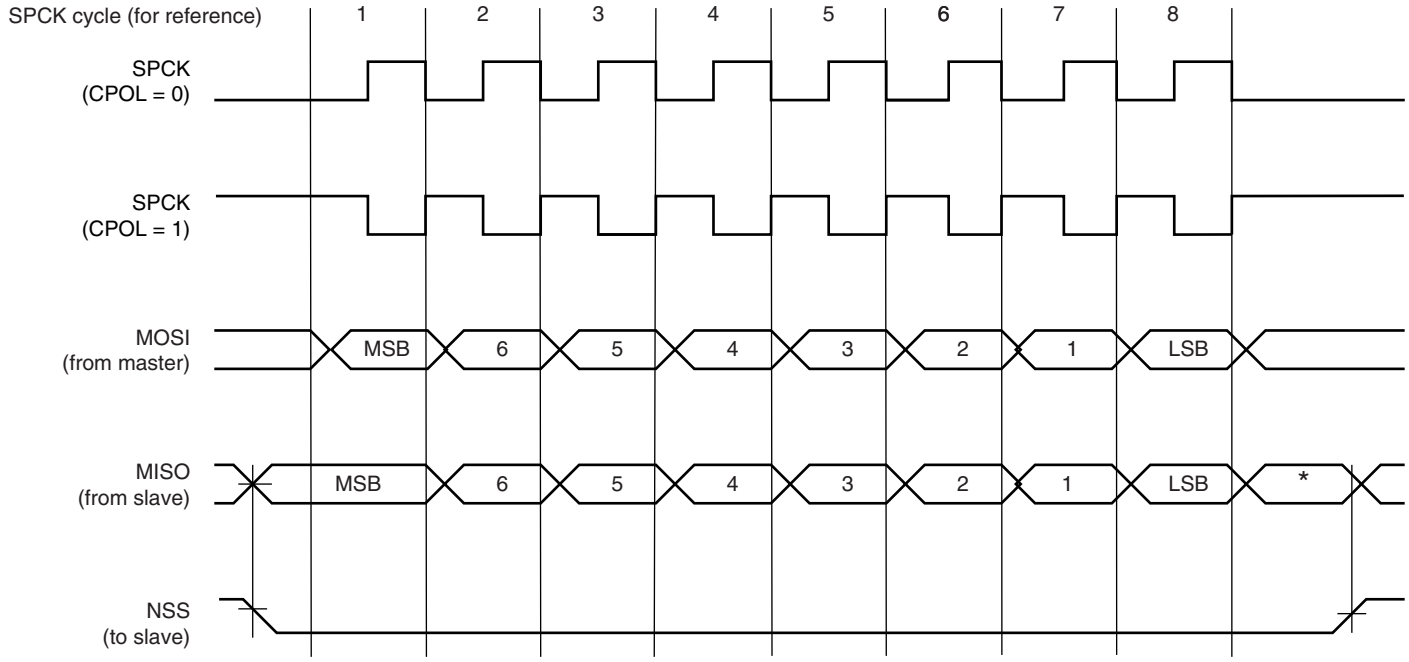
Table 32-2 shows the four modes and corresponding parameter settings.

Table 32-2. SPI Bus Protocol Mode

SPI Mode	CPOL	NCPHA
0	0	1
1	0	0
2	1	1
3	1	0

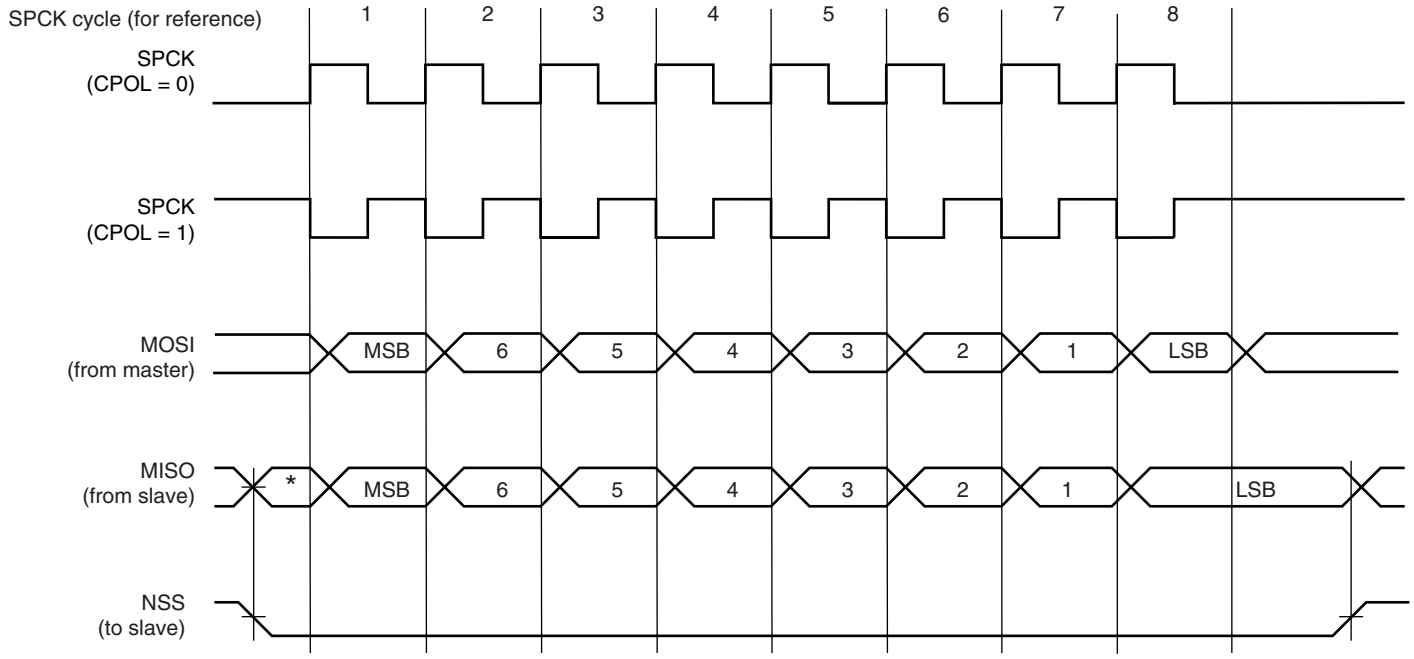
Figure 32-3 and Figure 32-4 show examples of data transfers.

Figure 32-3. SPI Transfer Format (NCPHA = 1, 8 bits per transfer)



* Not defined, but normally MSB of previous character received.

Figure 32-4. SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



* Not defined but normally LSB of previous character transmitted.

32.6.3 Master Mode Operations

When configured in Master Mode, the SPI operates on the clock generated by the internal programmable baud rate generator. It fully controls the data transfers to and from the slave(s)

connected to the SPI bus. The SPI drives the chip select line to the slave and the serial clock signal (SPCK).

The SPI features two holding registers, the Transmit Data Register and the Receive Data Register, and a single Shift Register. The holding registers maintain the data flow at a constant rate.

After enabling the SPI, a data transfer begins when the processor writes to the SPI_TDR (Transmit Data Register). The written data is immediately transferred in the Shift Register and transfer on the SPI bus starts. While the data in the Shift Register is shifted on the MOSI line, the MISO line is sampled and shifted in the Shift Register. Transmission cannot occur without reception.

Before writing the TDR, the PCS field must be set in order to select a slave.

If new data is written in SPI_TDR during the transfer, it stays in it until the current transfer is completed. Then, the received data is transferred from the Shift Register to SPI_RDR, the data in SPI_TDR is loaded in the Shift Register and a new transfer starts.

The transfer of a data written in SPI_TDR in the Shift Register is indicated by the TDRE bit (Transmit Data Register Empty) in the Status Register (SPI_SR). When new data is written in SPI_TDR, this bit is cleared. The TDRE bit is used to trigger the Transmit PDC channel.

The end of transfer is indicated by the TXEMPTY flag in the SPI_SR register. If a transfer delay (DLYBCT) is greater than 0 for the last transfer, TXEMPTY is set after the completion of said delay. The master clock (MCK) can be switched off at this time.

The transfer of received data from the Shift Register in SPI_RDR is indicated by the RDRF bit (Receive Data Register Full) in the Status Register (SPI_SR). When the received data is read, the RDRF bit is cleared.

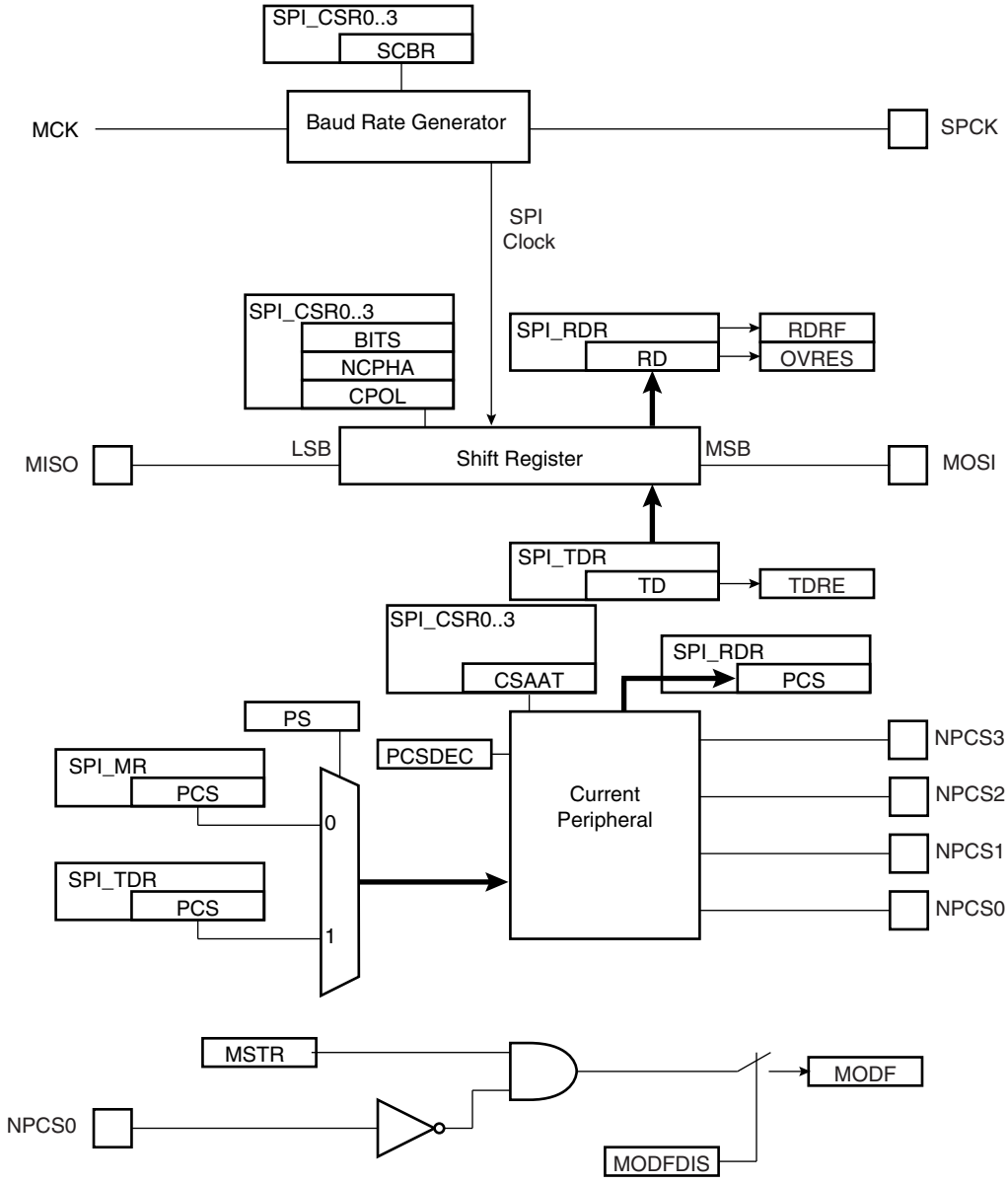
If the SPI_RDR (Receive Data Register) has not been read before new data is received, the Overrun Error bit (OVRES) in SPI_SR is set. As long as this flag is set, data is loaded in SPI_RDR. The user has to read the status register to clear the OVRES bit.

Figure 32-5, shows a block diagram of the SPI when operating in Master

Mode. Figure 32-6 on page 458 shows a flow chart describing how transfers are handled.

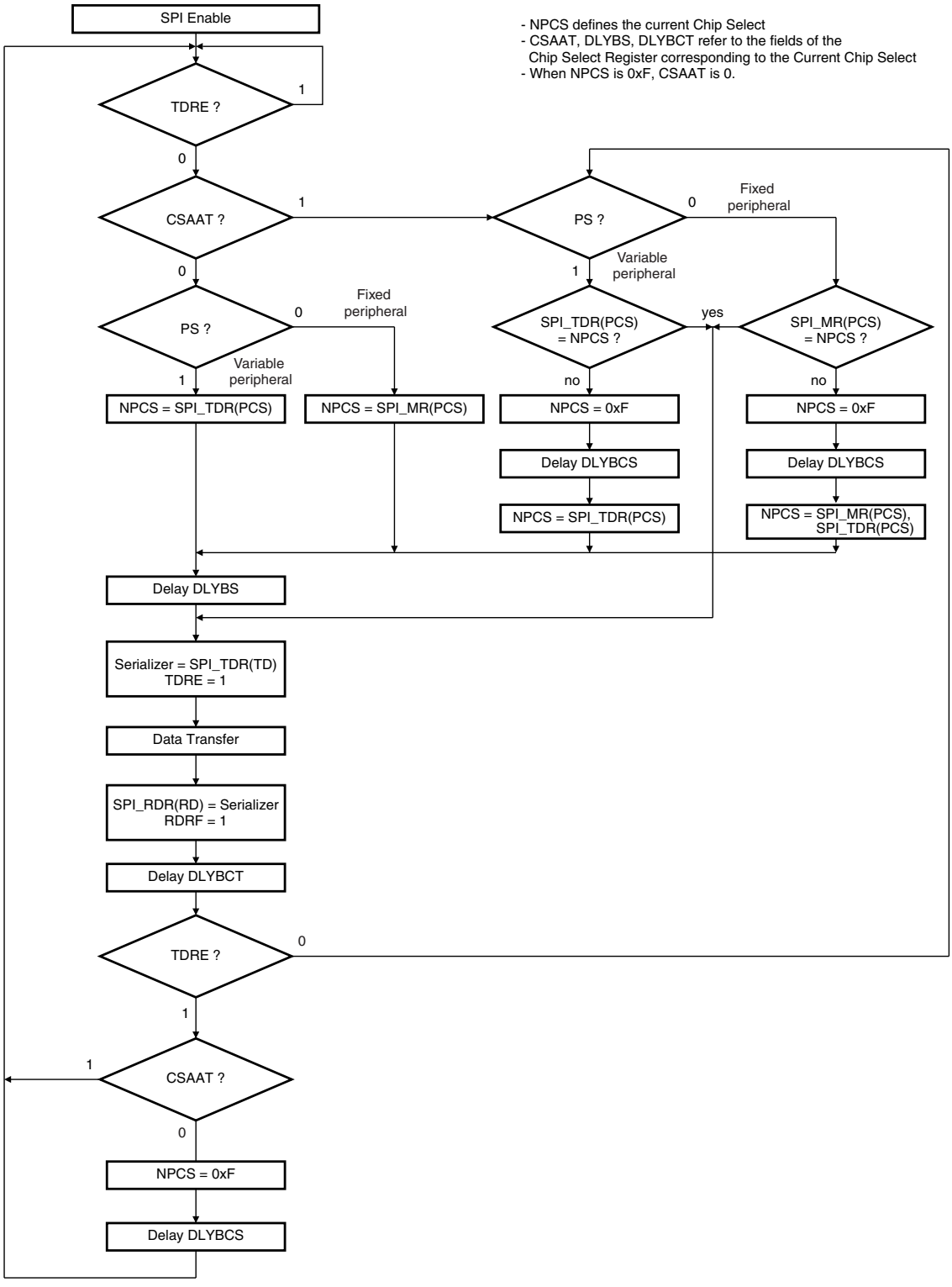
32.6.3.1 Master Mode Block Diagram

Figure 32-5. Master Mode Block Diagram



32.6.3.2 *Master Mode Flow Diagram*

Figure 32-6. Master Mode Flow Diagram



32.6.3.3 Clock Generation

The SPI Baud rate clock is generated by dividing the Master Clock (MCK), by a value between 1 and 255.

This allows a maximum operating baud rate at up to Master Clock and a minimum operating baud rate of MCK divided by 255.

Programming the SCBR field at 0 is forbidden. Triggering a transfer while SCBR is at 0 can lead to unpredictable results.

At reset, SCBR is 0 and the user has to program it at a valid value before performing the first transfer.

The divisor can be defined independently for each chip select, as it has to be programmed in the SCBR field of the Chip Select Registers. This allows the SPI to automatically adapt the baud rate for each interfaced peripheral without reprogramming.

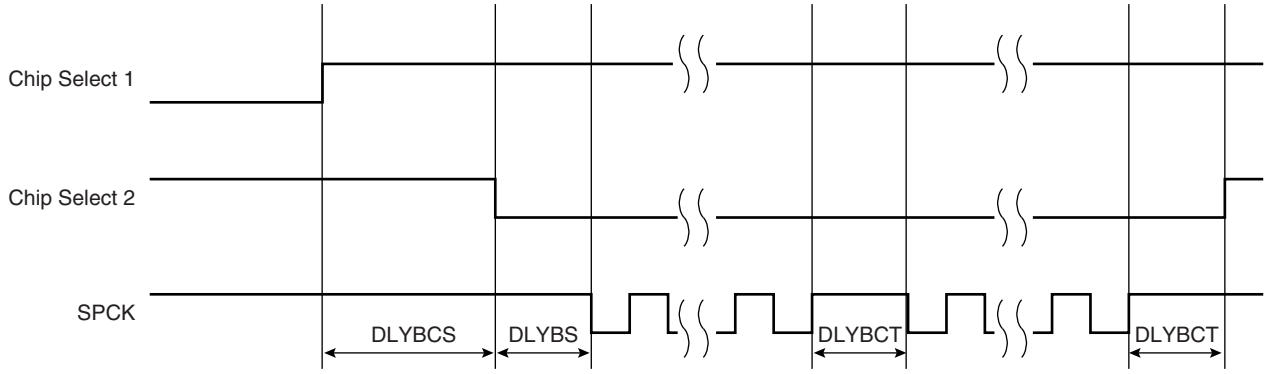
32.6.3.4 Transfer Delays

Figure 32-7 shows a chip select transfer change and consecutive transfers on the same chip select. Three delays can be programmed to modify the transfer waveforms:

- The delay between chip selects, programmable only once for all the chip selects by writing the DLYBCS field in the Mode Register. Allows insertion of a delay between release of one chip select and before assertion of a new one.
- The delay before SPCK, independently programmable for each chip select by writing the field DLYBS. Allows the start of SPCK to be delayed after the chip select has been asserted.
- The delay between consecutive transfers, independently programmable for each chip select by writing the DLYBCT field. Allows insertion of a delay between two transfers occurring on the same chip select

These delays allow the SPI to be adapted to the interfaced peripherals and their speed and bus release time.

Figure 32-7. Programmable Delays



32.6.3.5 Peripheral Selection

The serial peripherals are selected through the assertion of the NPCS0 to NPCS3 signals. By default, all the NPCS signals are high before and after each transfer.

The peripheral selection can be performed in two different ways:

- Fixed Peripheral Select: SPI exchanges data with only one peripheral

- Variable Peripheral Select: Data can be exchanged with more than one peripheral

Fixed Peripheral Select is activated by writing the PS bit to zero in SPI_MR (Mode Register). In this case, the current peripheral is defined by the PCS field in SPI_MR and the PCS field in the SPI_TDR has no effect.

Variable Peripheral Select is activated by setting PS bit to one. The PCS field in SPI_TDR is used to select the current peripheral. This means that the peripheral selection can be defined for each new data.

The Fixed Peripheral Selection allows buffer transfers with a single peripheral. Using the PDC is an optimal means, as the size of the data transfer between the memory and the SPI is either 8 bits or 16 bits. However, changing the peripheral selection requires the Mode Register to be reprogrammed.

The Variable Peripheral Selection allows buffer transfers with multiple peripherals without reprogramming the Mode Register. Data written in SPI_TDR is 32 bits wide and defines the real data to be transmitted and the peripheral it is destined to. Using the PDC in this mode requires 32-bit wide buffers, with the data in the 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.

32.6.3.6 Peripheral Chip Select Decoding

The user can program the SPI to operate with up to 15 peripherals by decoding the four Chip Select lines, NPCS0 to NPCS3 with an external logic. This can be enabled by writing the 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.

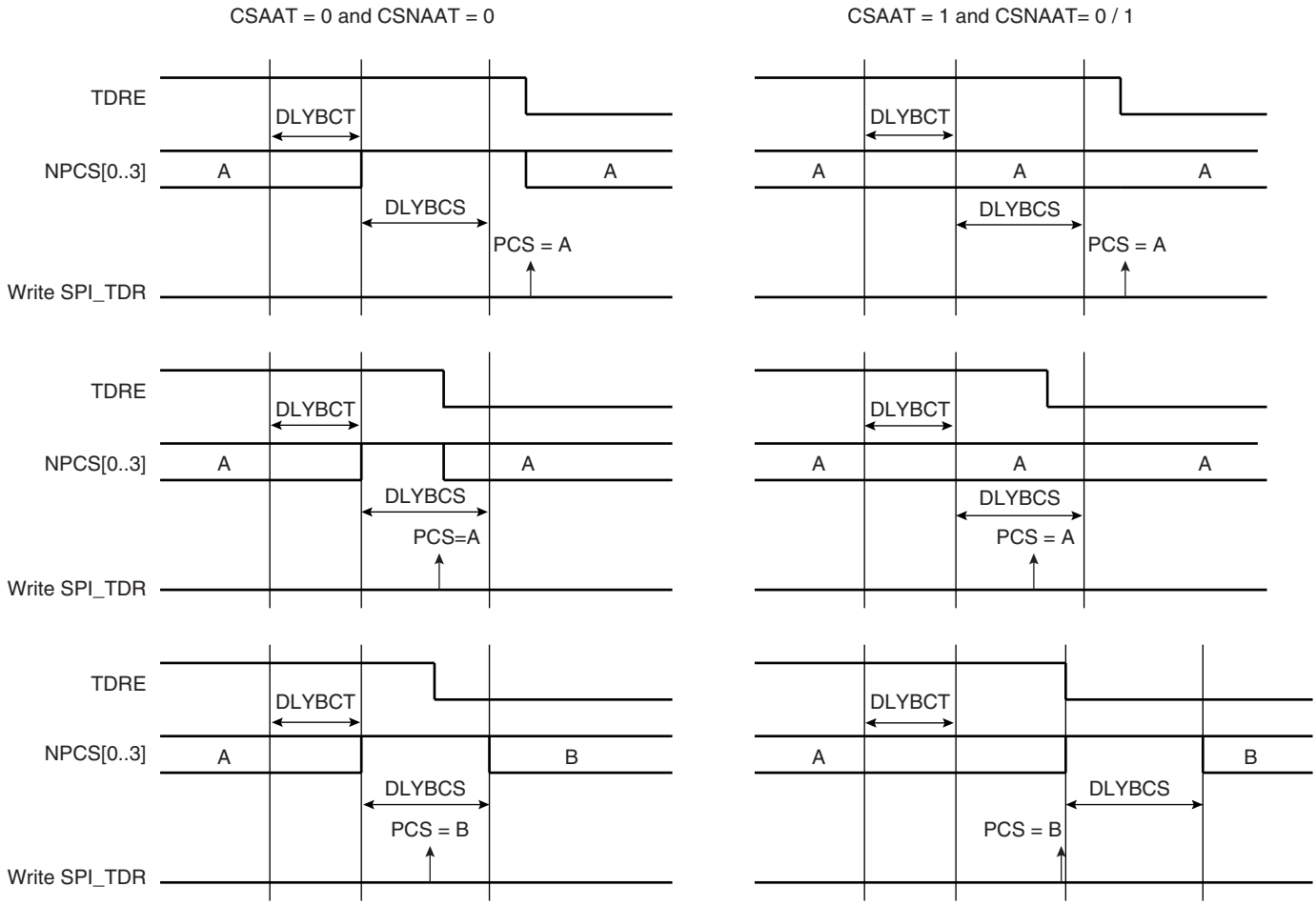
The SPI has only four Chip Select Registers, not 15. As a result, when decoding is activated, each chip select defines the characteristics of up to four peripherals. As an example, SPI_CRS0 defines the characteristics of the externally decoded peripherals 0 to 3, corresponding to the PCS values 0x0 to 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.

32.6.3.7 Peripheral Deselection

When operating normally, as soon as the transfer of the last data written in SPI_TDR is completed, the NPCS lines all rise. This might lead to runtime error if the processor is too long in responding to an interrupt, and thus might lead to difficulties for interfacing with some serial peripherals requiring the chip select line to remain active during a full set of transfers.

To facilitate interfacing with such devices, the Chip Select Register can be programmed with the CSAAT bit (Chip Select Active After Transfer) at 1. This allows the chip select lines to remain in their current state (low = active) until transfer to another peripheral is required.

Figure 32-8. Peripheral Deselection



32.6.3.8 Mode Fault Detection

A mode fault is detected when the SPI is programmed in Master Mode and a low level is driven by an external master on the NPCS0/NSS signal. NPCS0, MOSI, MISO and SPCK must be configured in open drain through the PIO controller, so that external pull up resistors are needed to guarantee high level.

When a mode fault is detected, the MODF bit in the SPI_SR is set until the SPI_SR is read and the SPI is automatically disabled until re-enabled by writing the SPIEN bit in the SPI_CR (Control Register) at 1.

By default, the Mode Fault detection circuitry is enabled. The user can disable Mode Fault detection by setting the MODFDIS bit in the SPI Mode Register (SPI_MR).

32.6.4 SPI Slave Mode

When operating in Slave Mode, the SPI processes data bits on the clock provided on the SPI clock pin (SPCK).

The SPI waits for NSS to go active before receiving the serial clock from an external master. When NSS falls, the clock is validated on the serializer, which processes the number of bits defined by the BITS field of the Chip Select Register 0 (SPI_CSR0). These bits are processed following a phase and a polarity defined respectively by the NCPHA and CPOL bits of the

SPI_CSR0. Note that BITS, CPOL and NCPHA of the other Chip Select Registers have no effect when the SPI is programmed in Slave Mode.

The bits are shifted out on the MISO line and sampled on the MOSI line.

(For more information on BITS field, see also, the (Note:) below the register table; Section 32.7.9 “SPI Chip Select Register” on page 474.)

When all the bits are processed, the received data is transferred in the Receive Data Register and the RDRF bit rises. If the SPI_RDR (Receive Data Register) has not been read before new data is received, the Overrun Error bit (OVRES) in SPI_SR is set. As long as this flag is set, data is loaded in SPI_RDR. The user has to read the status register to clear the OVRES bit.

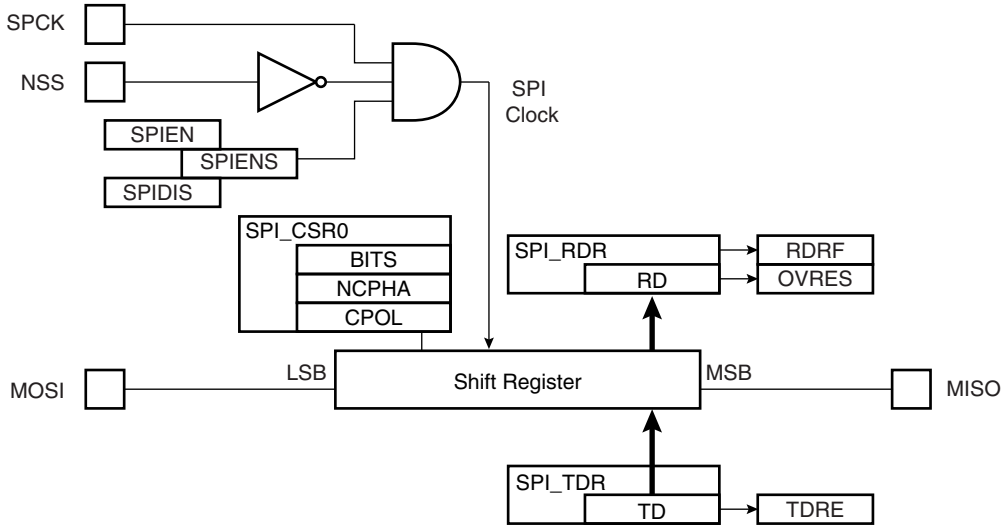
When a transfer starts, the data shifted out is the data present in the Shift Register. If no data has been written in the Transmit Data Register (SPI_TDR), the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift Register resets at 0.

When a first data is written in SPI_TDR, it is transferred immediately in the Shift Register and the TDRE bit rises. If new data is written, it remains in SPI_TDR until a transfer occurs, i.e. NSS falls and there is a valid clock on the SPCK pin. When the transfer occurs, the last data written in SPI_TDR is transferred in the Shift Register and the TDRE bit rises. This enables frequent updates of critical variables with single transfers.

Then, a new data is loaded in the Shift Register from the Transmit Data Register. In case no character is ready to be transmitted, i.e. no character has been written in SPI_TDR since the last load from SPI_TDR to the Shift Register, the Shift Register is not modified and the last received character is retransmitted.

Figure 32-9 shows a block diagram of the SPI when operating in Slave Mode.

Figure 32-9. Slave Mode Functional Bloc Diagram



32.7 Serial Peripheral Interface (SPI) User Interface

Table 32-3. 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	–	–	–
0x004C - 0x00FC	Reserved	–	–	–
0x100 - 0x124	Reserved for the PDC			

32.7.1 SPI Control Register

Name: SPI_CR

Addresses: 0xFFFA4000 (0), 0xFFFA8000 (1)

Access Type: Write-only

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

- **SPIEN: SPI Enable**

0 = No effect.

1 = Enables the SPI to transfer and receive data.

- **SPIDIS: SPI Disable**

0 = No effect.

1 = Disables the SPI.

As soon as SPIDIS is set, SPI finishes its transfer.

All pins are set in input mode and no data is received or transmitted.

If a transfer is in progress, the transfer is finished before the SPI is disabled.

If both SPIEN and SPIDIS are equal to one when the control register is written, the SPI is disabled.

- **SWRST: SPI Software Reset**

0 = No effect.

1 = Reset the SPI. A software-triggered hardware reset of the SPI interface is performed.

The SPI is in slave mode after software reset.

PDC channels are not affected by software reset.

- **LASTXFER: Last Transfer**

0 = No effect.

1 = The current NPCS will be deasserted after the character written in TD has been transferred. When CSAAT is set, this allows to close the communication with the current serial peripheral by raising the corresponding NPCS line as soon as TD transfer has completed.

32.7.2 SPI Mode Register

Name: SPI_MR

Addresses: 0xFFFA4004 (0), 0xFFFA8004 (1)

Access Type: Read/Write

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

- **MSTR: Master/Slave Mode**

0 = SPI is in Slave mode.

1 = SPI is in Master mode.

- **PS: Peripheral Select**

0 = Fixed Peripheral Select.

1 = Variable Peripheral Select.

- **PCSDEC: Chip Select Decode**

0 = The chip selects are directly connected to a peripheral device.

1 = The four chip select lines are connected to a 4- to 16-bit decoder.

When PCSDEC equals one, up to 15 Chip Select signals can be generated with the four lines using an external 4- to 16-bit decoder. The Chip Select Registers define the characteristics of the 15 chip selects according to the following rules:

SPI_CSR0 defines peripheral chip select signals 0 to 3.

SPI_CSR1 defines peripheral chip select signals 4 to 7.

SPI_CSR2 defines peripheral chip select signals 8 to 11.

SPI_CSR3 defines peripheral chip select signals 12 to 14.

- **MODFDIS: Mode Fault Detection**

0 = Mode fault detection is enabled.

1 = Mode fault detection is disabled.

- **LLB: Local Loopback Enable**

0 = Local loopback path disabled.

1 = Local loopback path enabled

LLB controls the local loopback on the data serializer for testing in Master Mode only. (MISO is internally connected on MOSI.)

- **PCS: Peripheral Chip Select**

This field is only used if Fixed Peripheral Select is active (PS = 0).

If PCSDEC = 0:

PCS = 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:

$$\text{Delay Between Chip Selects} = \frac{DLYBCS}{MCK}$$

32.7.3 SPI Receive Data Register

Name: SPI_RDR

Addresses: 0xFFFA4008 (0), 0xFFFA8008 (1)

Access Type: Read-only

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

- **RD: Receive Data**

Data received by the SPI Interface is stored in this register right-justified. Unused bits read zero.

- **PCS: Peripheral Chip Select**

In Master Mode only, these bits indicate the value on the NPCS pins at the end of a transfer. Otherwise, these bits read zero.

32.7.4 SPI Transmit Data Register

Name: SPI_TDR

Addresses: 0xFFFA400C (0), 0xFFFA800C (1)

Access Type: Write-only

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

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

15	14	13	12	11	10	9	8
TD							

7	6	5	4	3	2	1	0
TD							

- **TD: Transmit Data**

Data to be transmitted by the SPI Interface is stored in this register. Information to be transmitted must be written to the transmit data register in a right-justified format.

- **PCS: Peripheral Chip Select**

This field is only used if Variable Peripheral Select is active (PS = 1).

If PCSDEC = 0:

PCS = 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).

32.7.5 SPI Status Register

Name: SPI_SR

Addresses: 0xFFFA4010 (0), 0xFFFA8010 (1)

Access Type: Read-only

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

- **RDRF: Receive Data Register Full**

0 = No data has been received since the last read of SPI_RDR

1 = Data has been received and the received data has been transferred from the serializer to SPI_RDR since the last read of SPI_RDR.

- **TDRE: Transmit Data Register Empty**

0 = Data has been written to SPI_TDR and not yet transferred to the serializer.

1 = The last data written in the Transmit Data Register has been transferred to the serializer.

TDRE equals zero when the SPI is disabled or at reset. The SPI enable command sets this bit to one.

- **MODF: Mode Fault Error**

0 = No Mode Fault has been detected since the last read of SPI_SR.

1 = A Mode Fault occurred since the last read of the SPI_SR.

- **OVRES: Overrun Error Status**

0 = No overrun has been detected since the last read of SPI_SR.

1 = An overrun has occurred since the last read of SPI_SR.

An overrun occurs when SPI_RDR is loaded at least twice from the serializer since the last read of the SPI_RDR.

- **ENDRX: End of RX buffer**

0 = The Receive Counter Register has not reached 0 since the last write in SPI_RCR(1) or SPI_RNCR(1).

1 = The Receive Counter Register has reached 0 since the last write in SPI_RCR(1) or SPI_RNCR(1).

- **ENDTX: End of TX buffer**

0 = The Transmit Counter Register has not reached 0 since the last write in SPI_TCR(1) or SPI_TNCR(1).

1 = The Transmit Counter Register has reached 0 since the last write in SPI_TCR(1) or SPI_TNCR(1).

- **RXBUFF: RX Buffer Full**

0 = SPI_RCR(1) or SPI_RNCR(1) has a value other than 0.

1 = Both SPI_RCR(1) and SPI_RNCR(1) have a value of 0.

- **TXBUFE: TX Buffer Empty**

0 = SPI_TCR[1] or SPI_TNCR[1] has a value other than 0.

1 = Both SPI_TCR[1] and SPI_TNCR[1] have a value of 0.

- **NSSR: NSS Rising**

0 = No rising edge detected on NSS pin since last read.

1 = A rising edge occurred on NSS pin since last read.

- **TXEMPTY: Transmission Registers Empty**

0 = As soon as data is written in SPI_TDR.

1 = SPI_TDR and internal shifter are empty. If a transfer delay has been defined, TXEMPTY is set after the completion of such delay.

- **SPIENS: SPI Enable Status**

0 = SPI is disabled.

1 = SPI is enabled.

Note: 1. SPI_RCR, SPI_RNCR, SPI_TCR, SPI_TNCR are physically located in the PDC.

32.7.6 SPI Interrupt Enable Register

Name: SPI_IER

Addresses: 0xFFFA4014 (0), 0xFFFA8014 (1)

Access Type: Write-only

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

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**

32.7.7 SPI Interrupt Disable Register

Name: SPI_IDR

Addresses: 0xFFFA4018 (0), 0xFFFA8018 (1)

Access Type: Write-only

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

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**

32.7.8 SPI Interrupt Mask Register

Name: SPI_IMR

Addresses: 0xFFFA401C (0), 0xFFFA801C (1)

Access Type: Read-only

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

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**

32.7.9 SPI Chip Select Register

Name: SPI_CSR0... SPI_CSR3

Addresses: 0xFFFA4030 (0), 0xFFFA8030 (1)

Access Type: Read/Write

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

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 32.7.9 “SPI Chip Select Register” on page 474.)

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:

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{MCK}$$

33. Two-wire Interface (TWI)

33.1 Description

The Atmel Two-wire Interface (TWI) interconnects components on a unique two-wire bus, made up of one clock line and one data line with speeds of up to 400 Kbits per second, based on a byte-oriented transfer format. It can be used with any Atmel Two-wire Interface bus Serial EEPROM and 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. Arbitration of the bus is performed internally and puts the TWI in slave mode automatically if the bus arbitration is lost.

A configurable baud rate generator permits the output data rate to be adapted to a wide range of core clock frequencies.

Below, Table 33-1 lists the compatibility level of the Atmel Two-wire Interface in Master Mode and a full I2C compatible device.

Table 33-1. Atmel TWI compatibility with i2C Standard

I2C Standard	Atmel TWI
Standard Mode Speed (100 KHz)	Supported
Fast Mode Speed (400 KHz)	Supported
7 or 10 bits Slave Addressing	Supported
START BYTE(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

Note: 1. START + b000000001 + Ack + Sr

33.2 Embedded Characteristics

- Master, MultiMaster and Slave modes supported
- General Call supported in Slave mode

33.3 List of Abbreviations

Table 33-2. Abbreviations

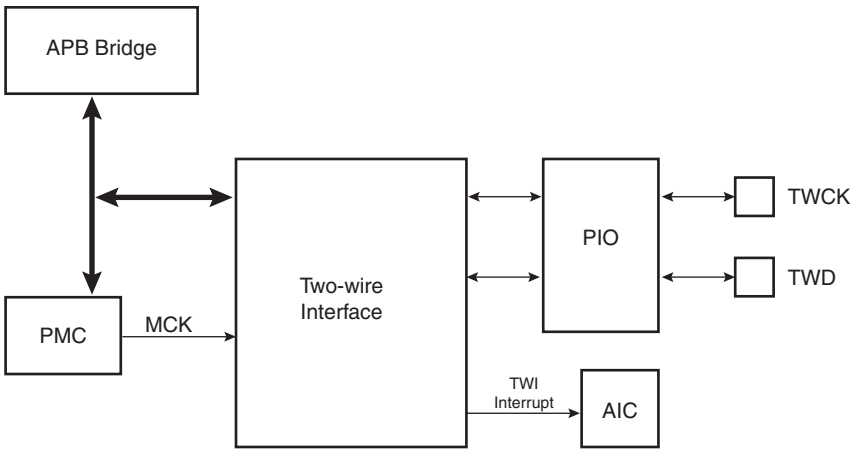
Abbreviation	Description
TWI	Two-wire Interface
A	Acknowledge
NA	Non Acknowledge
P	Stop
S	Start
Sr	Repeated Start

Table 33-2. Abbreviations

Abbreviation	Description
SADR	Slave Address
ADR	Any address except SADR
R	Read
W	Write

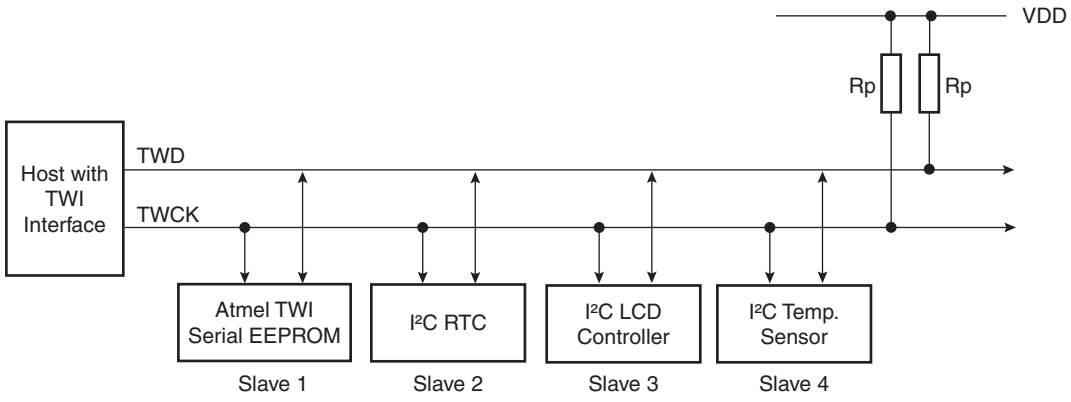
33.4 Block Diagram

Figure 33-1. Block Diagram



33.5 Application Block Diagram

Figure 33-2. Application Block Diagram



Rp: Pull up value as given by the I²C Standard

33.5.1 I/O Lines Description

Table 33-3. I/O Lines Description

Pin Name	Pin Description	Type
TWD	Two-wire Serial Data	Input/Output
TWCK	Two-wire Serial Clock	Input/Output

33.6 Product Dependencies

33.6.1 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 33-2 on page 478). When the bus is free, both lines are high. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function.

TWD and TWCK pins may be multiplexed with PIO lines. To enable the TWI, the programmer must perform the following steps:

- Program the PIO controller to:
  - Dedicate TWD and TWCK as peripheral lines.
  - Define TWD and TWCK as open-drain.

33.6.2 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.

33.6.3 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.

33.7 Functional Description

33.7.1 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 33-4).

Each transfer begins with a START condition and terminates with a STOP condition (see Figure 33-3).

- A high-to-low transition on the TWD line while TWCK is high defines the START condition.
- A low-to-high transition on the TWD line while TWCK is high defines a STOP condition.

Figure 33-3. START and STOP Conditions

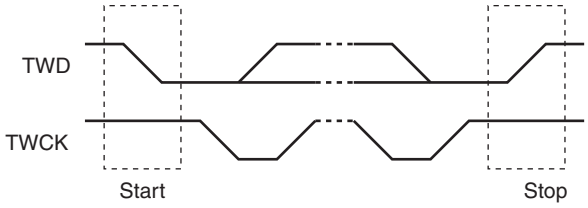
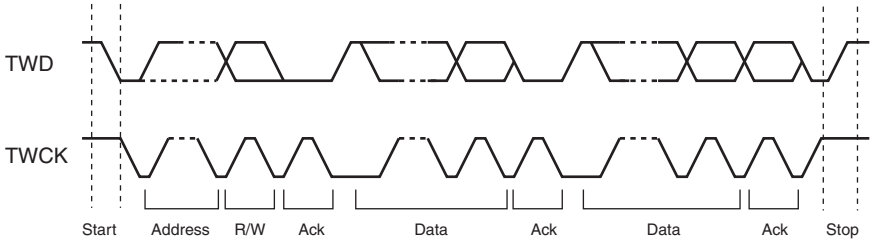


Figure 33-4. Transfer Format



33.7.2 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.

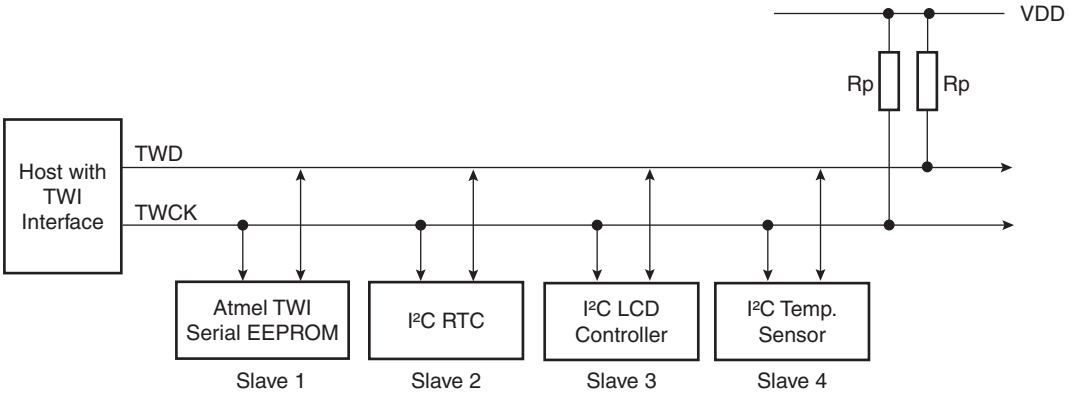
33.8 Master Mode

33.8.1 Definition

The Master is the device which starts a transfer, generates a clock and stops it.

33.8.2 Application Block Diagram

Figure 33-5. Master Mode Typical Application Block Diagram



Rp: Pull up value as given by the I²C Standard

33.8.3 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.

33.8.4 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. When no more data is written into the TWI_THR, the master generates a stop condition to end the transfer. The end of the complete transfer is marked by the TWI_TXCOMP bit set to one. See Figure 33-6, Figure 33-7, and Figure 33-8.

Figure 33-6. Master Write with One Data Byte

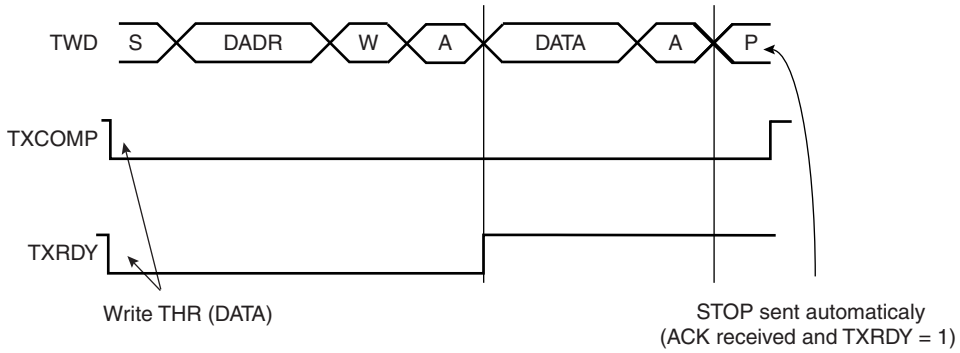


Figure 33-7. Master Write with Multiple Data Byte

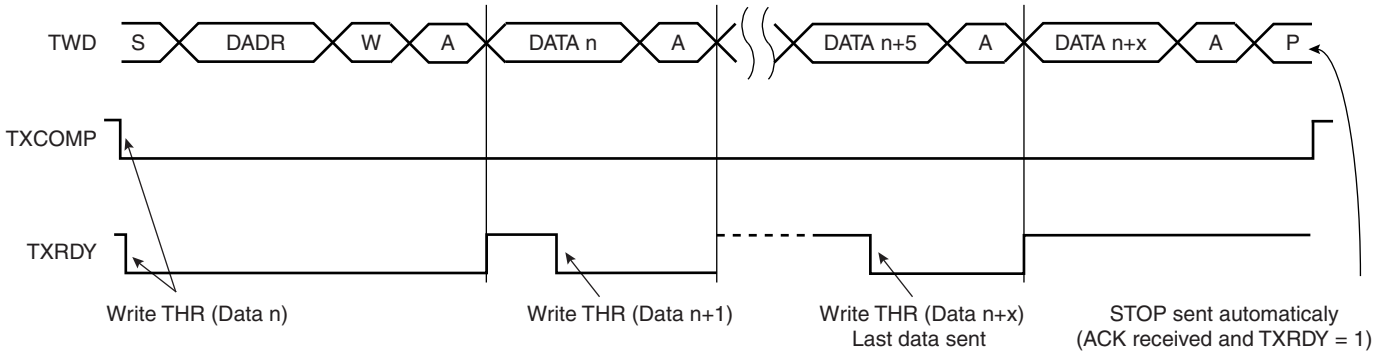
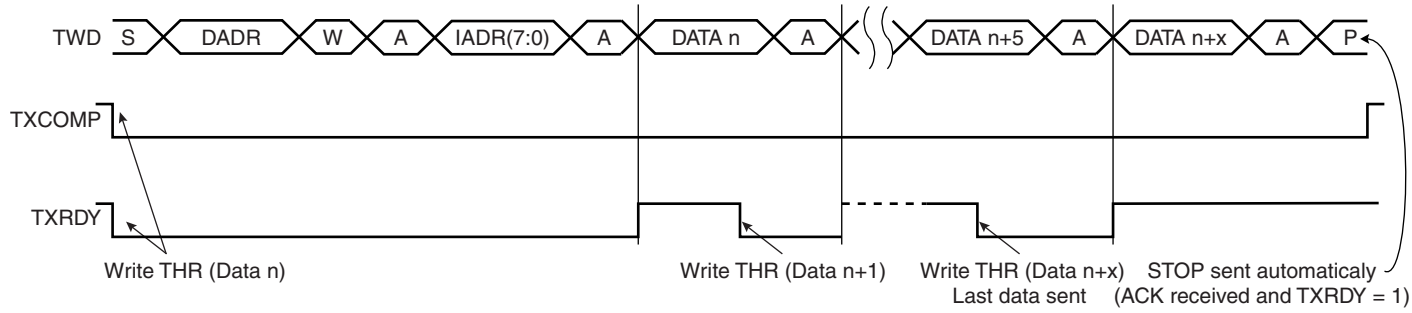


Figure 33-8. Master Write with One Byte Internal Address and Multiple Data Bytes



33.8.5 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 33-9. When the RXRDY bit is set in the status register, a character has been received in the receive-holding register (TWI_RHR). The RXRDY bit is reset when reading the TWI_RHR.

When a single data byte read is performed, with or without internal address **(IADR)**, the START and STOP bits must be set at the same time. See Figure 33-9. When a multiple data byte read is performed, with or without internal address **(IADR)**, the STOP bit must be set after the next-to-last data received. See Figure 33-10. For Internal Address usage see Section 33.8.6.

Figure 33-9. Master Read with One Data Byte

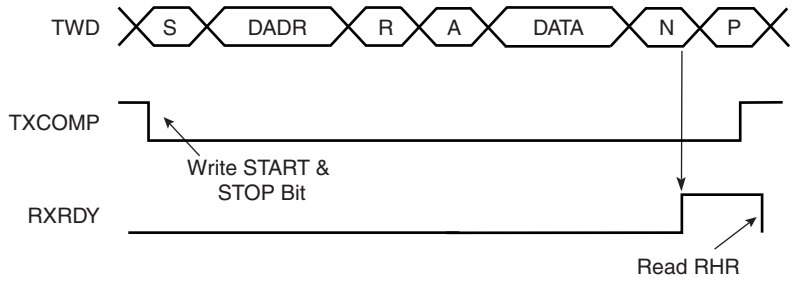
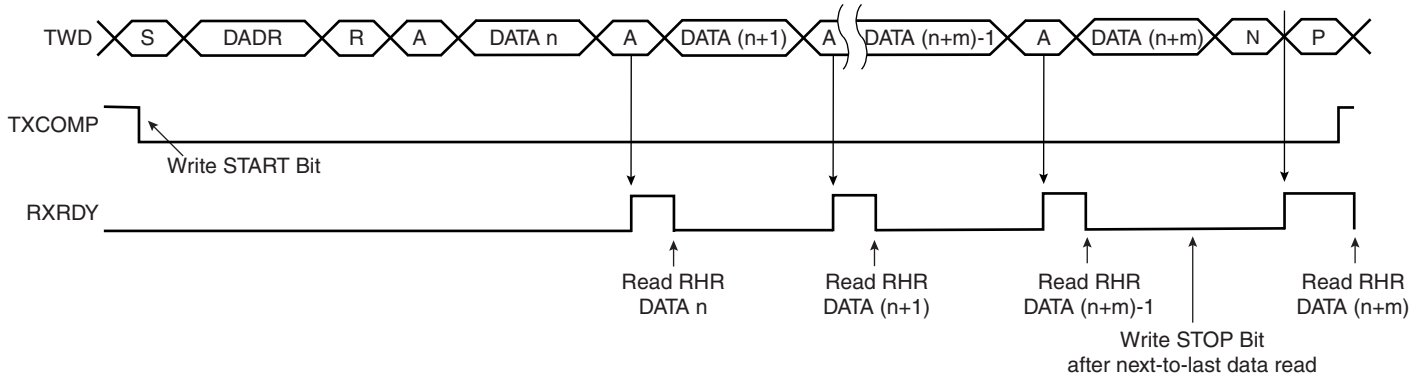


Figure 33-10. Master Read with Multiple Data Bytes



33.8.6 Internal Address

The TWI interface can perform various transfer formats: Transfers with 7-bit slave address devices and 10-bit slave address devices.

33.8.6.1 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 33-12. See Figure 33-11 and Figure 33-13 for Master Write operation with internal address.

The three internal address bytes are configurable through the Master Mode register (TWI_MMR).

If the slave device supports only a 7-bit address, i.e. no internal address, **IADRSZ** must be set to 0.

In the figures below the following abbreviations are used:

- S Start
- Sr Repeated Start
- P Stop
- W Write
- R Read
- A Acknowledge
- N Not Acknowledge
- DADR Device Address
- IADR Internal Address

Figure 33-11. Master Write with One, Two or Three Bytes Internal Address and One Data Byte

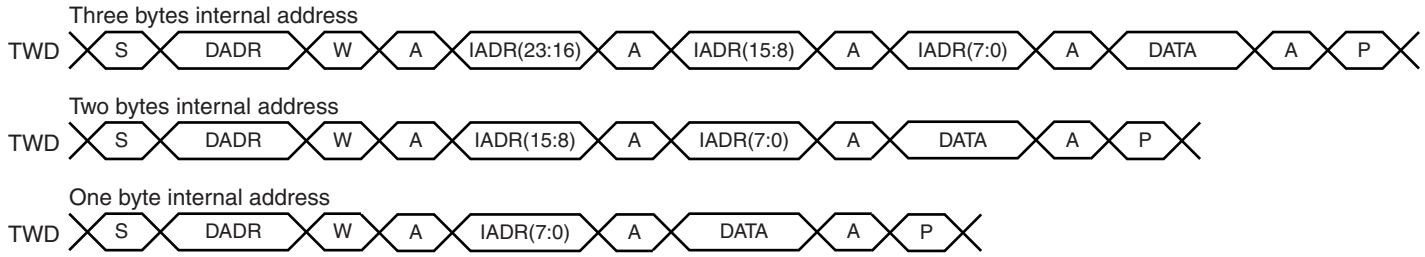
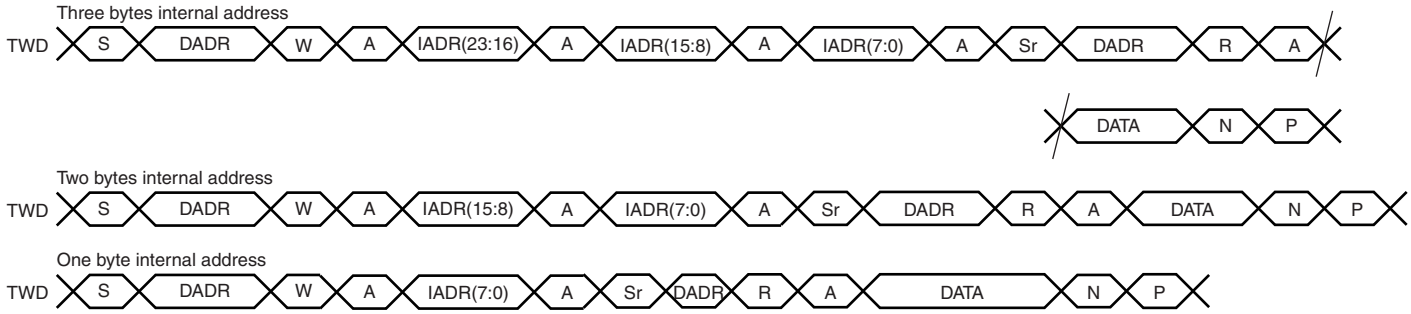


Figure 33-12. Master Read with One, Two or Three Bytes Internal Address and One Data Byte



33.8.6.2 10-bit Slave Addressing

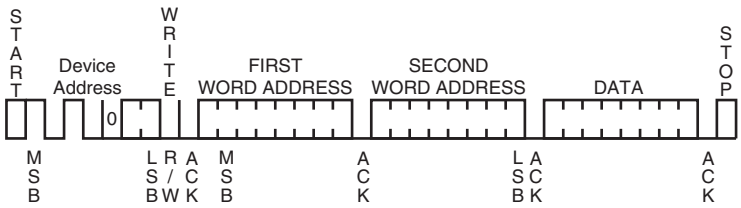
For a slave address higher than 7 bits, the user must configure the address size **(IADRSZ)** and set the other slave address bits in the internal address register (TWI_IADR). The two remaining Internal address bytes, IADR[15:8] and IADR[23:16] can be used the same as in 7-bit Slave Addressing.

Example: Address a 10-bit device (10-bit device address is b1 b2 b3 b4 b5 b6 b7 b8 b9 b10)

1. Program IADRSZ = 1,
2. Program DADR with 1 1 1 1 0 b1 b2 (b1 is the MSB of the 10-bit address, b2, etc.)
3. Program TWI_IADR with b3 b4 b5 b6 b7 b8 b9 b10 (b10 is the LSB of the 10-bit address)

Figure 33-13 below shows a byte write to an Atmel AT24LC512 EEPROM. This demonstrates the use of internal addresses to access the device.

Figure 33-13. Internal Address Usage



33.8.7 Read-write Flowcharts

The following flowcharts shown in Figure 33-14, Figure 33-15 on page 487, Figure 33-16 on page 488, Figure 33-17 on page 489, Figure 33-18 on page 490 and Figure 33-19 on page 491 give examples for read and write operations. A polling or interrupt method can be used to check the status bits. The interrupt method requires that the interrupt enable register (TWI_IER) be configured first.

Figure 33-14. TWI Write Operation with Single Data Byte without Internal Address

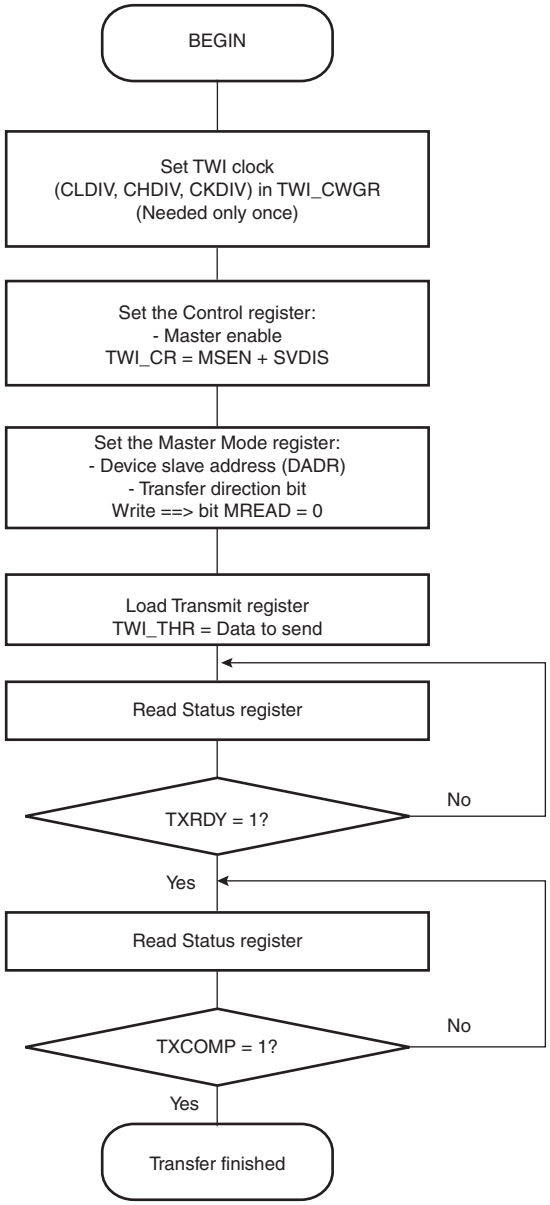


Figure 33-15. TWI Write Operation with Single Data Byte and Internal Address

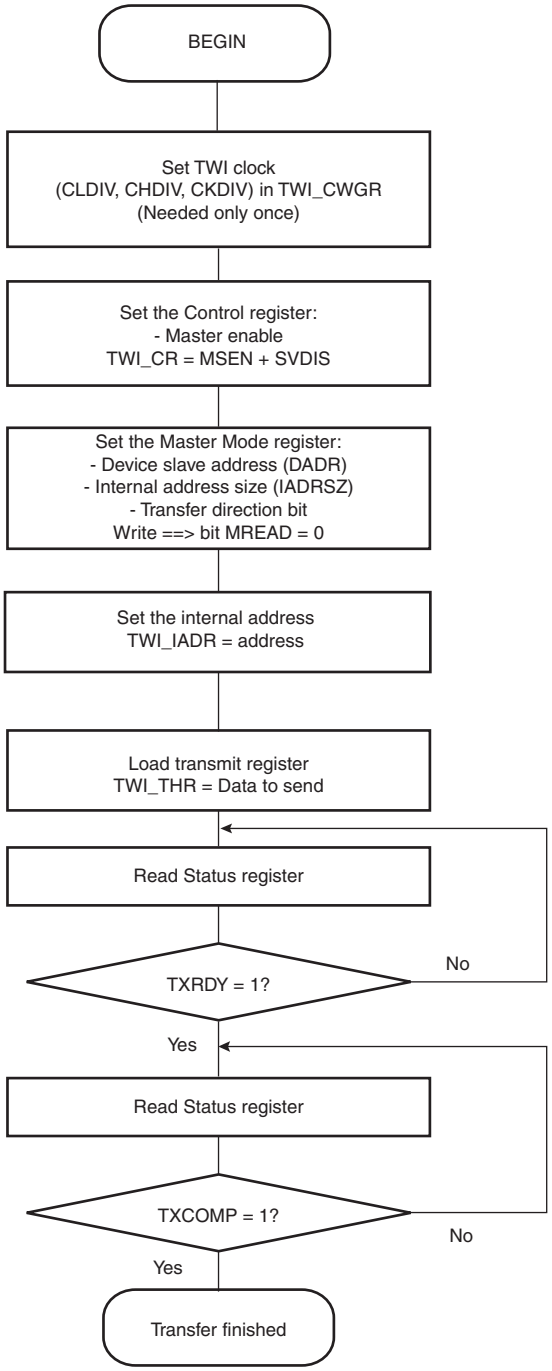


Figure 33-16. TWI Write Operation with Multiple Data Bytes with or without Internal Address

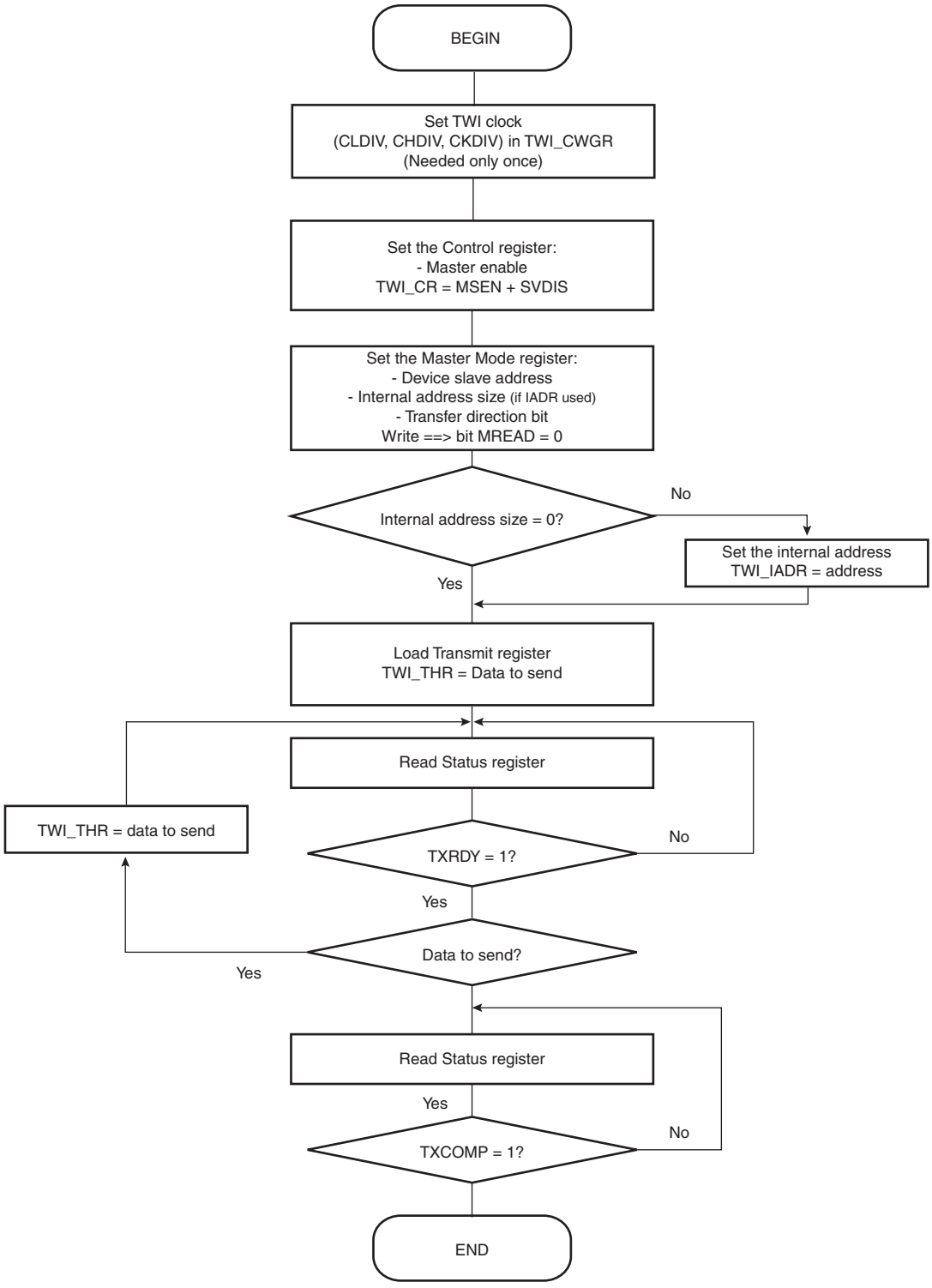


Figure 33-17. TWI Read Operation with Single Data Byte without Internal Address

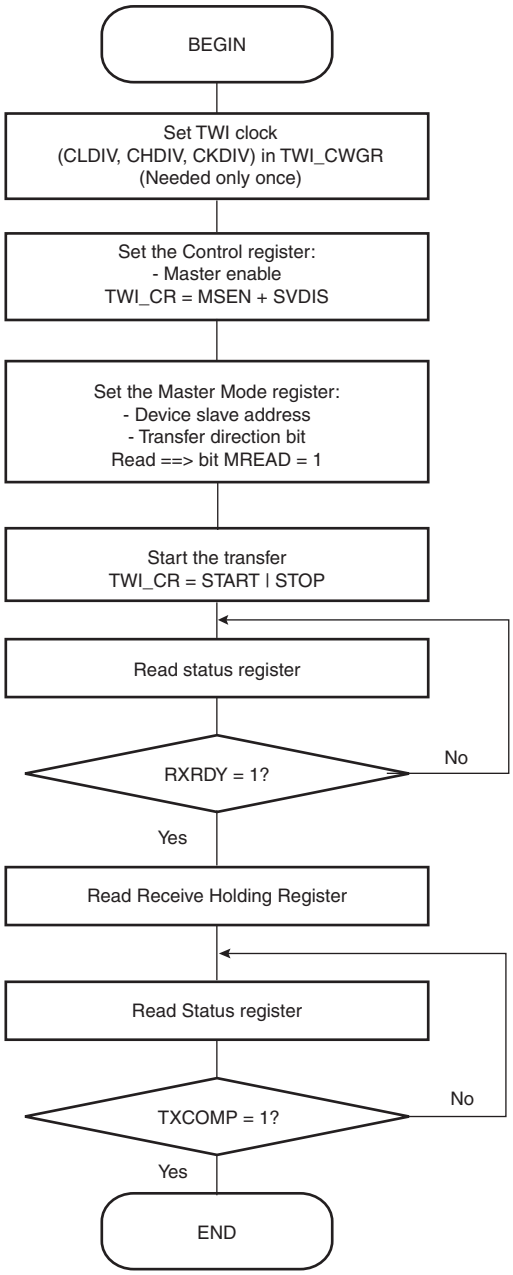


Figure 33-18. TWI Read Operation with Single Data Byte and Internal Address

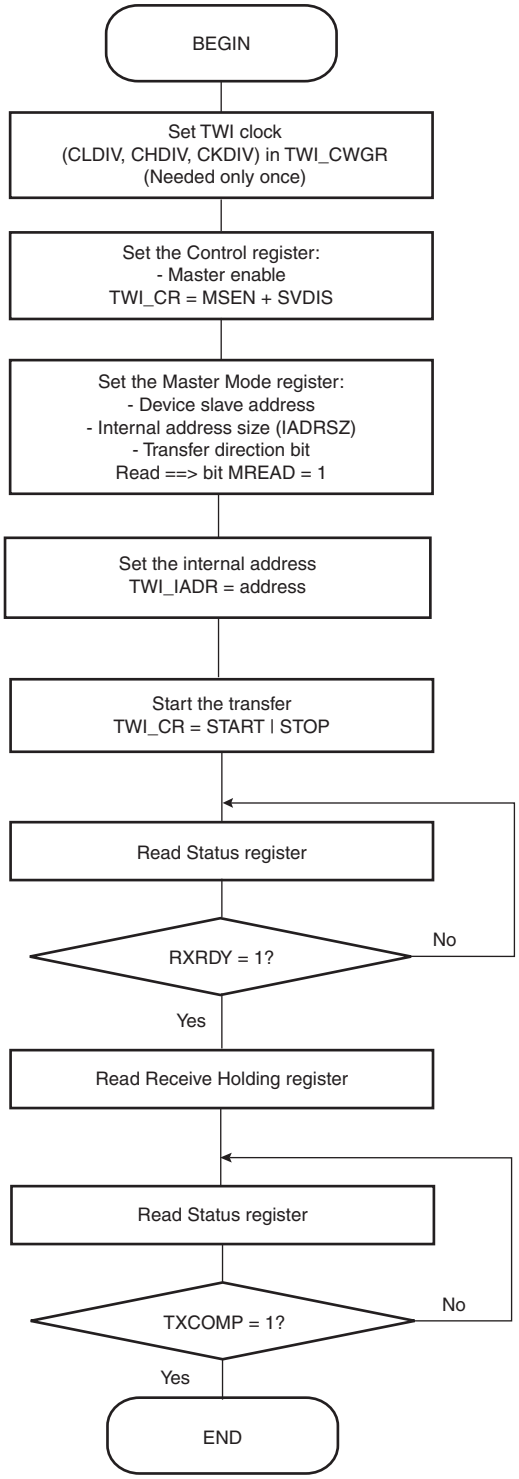
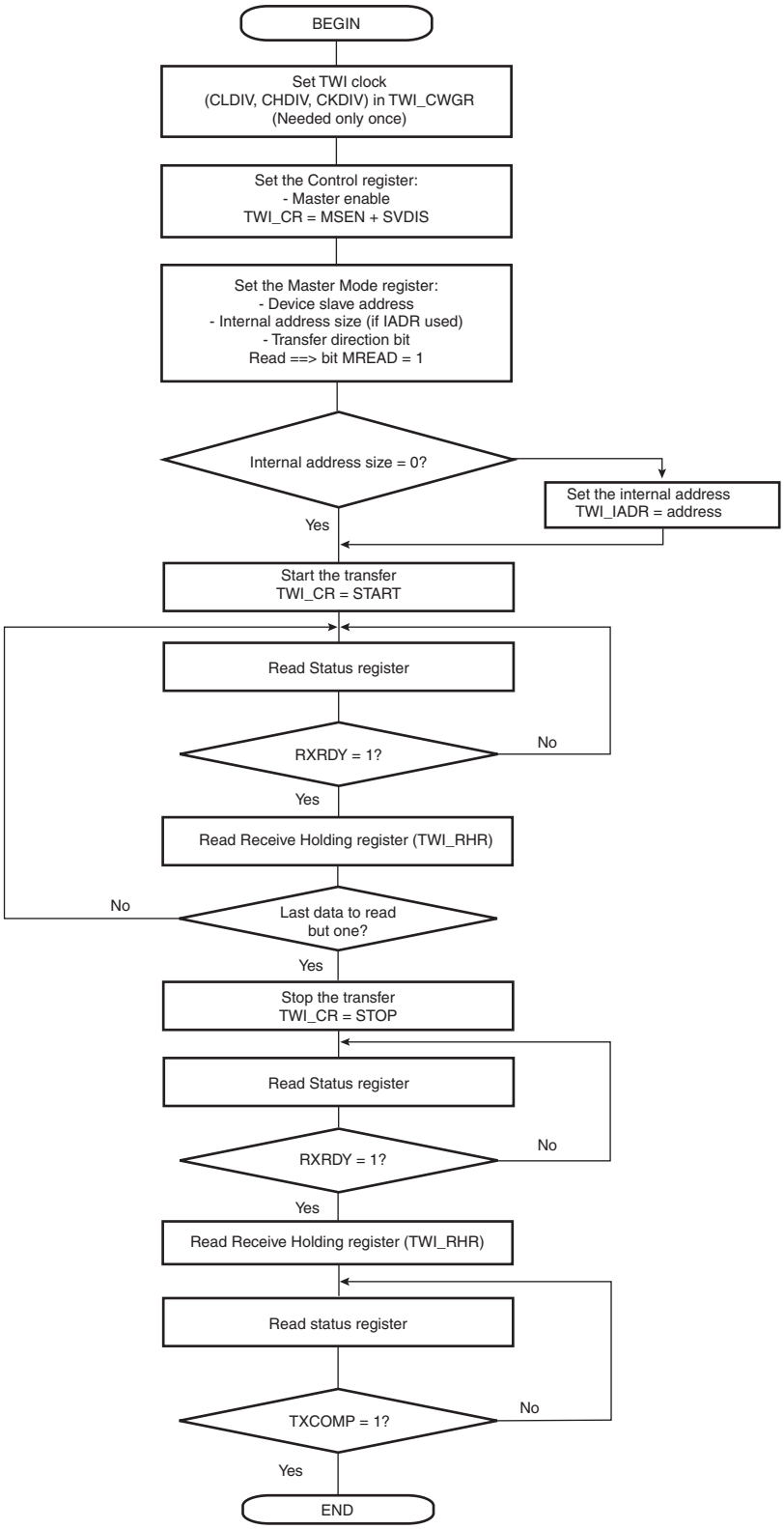


Figure 33-19. TWI Read Operation with Multiple Data Bytes with or without Internal Address



33.9 Multi-master Mode

33.9.1 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 33-21 on page 493.

33.9.2 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.

33.9.2.1 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 33-20 on page 493).

Note: The state of the bus (busy or free) is not indicated in the user interface.

33.9.2.2 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 33-20. Programmer Sends Data While the Bus is Busy

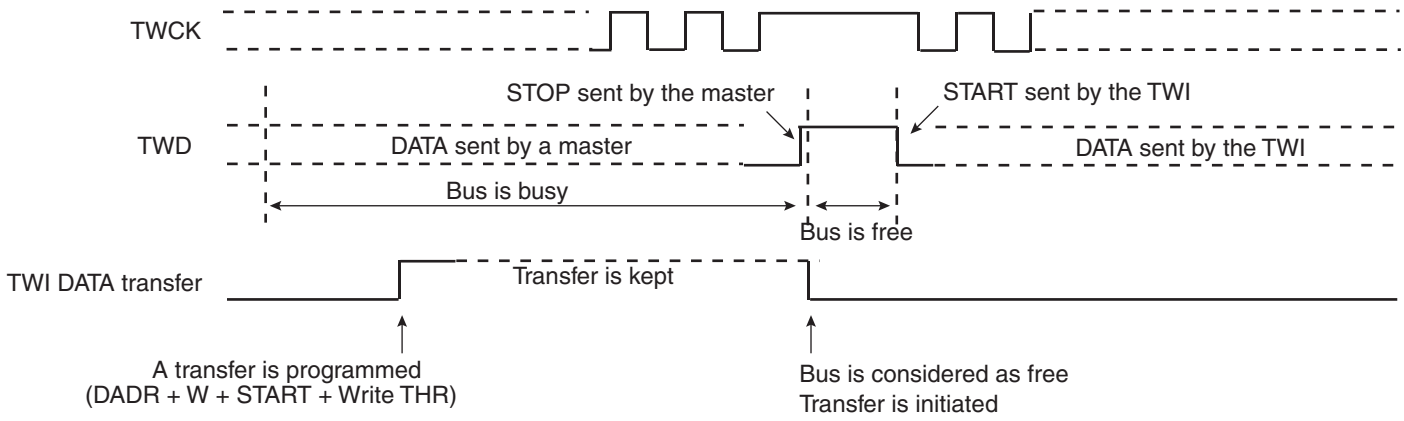
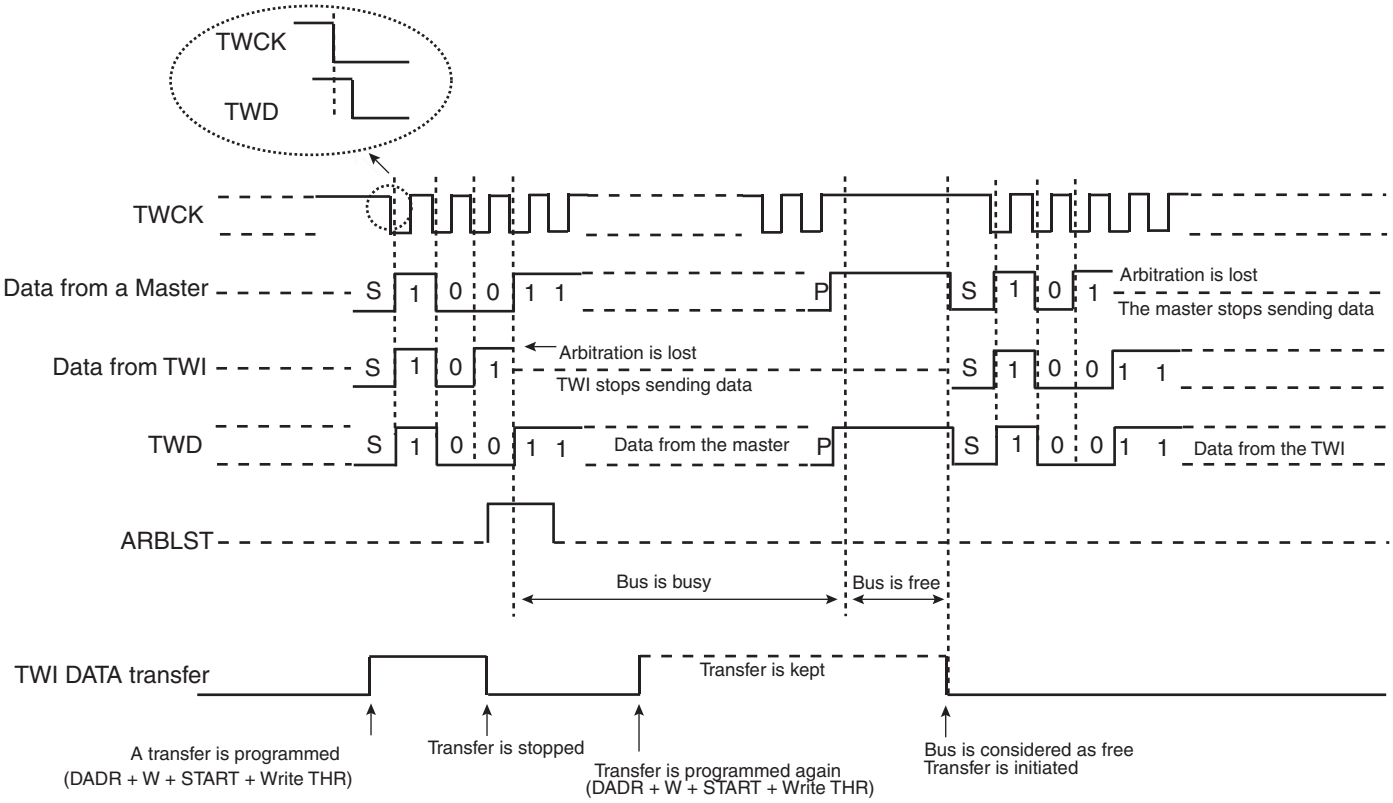
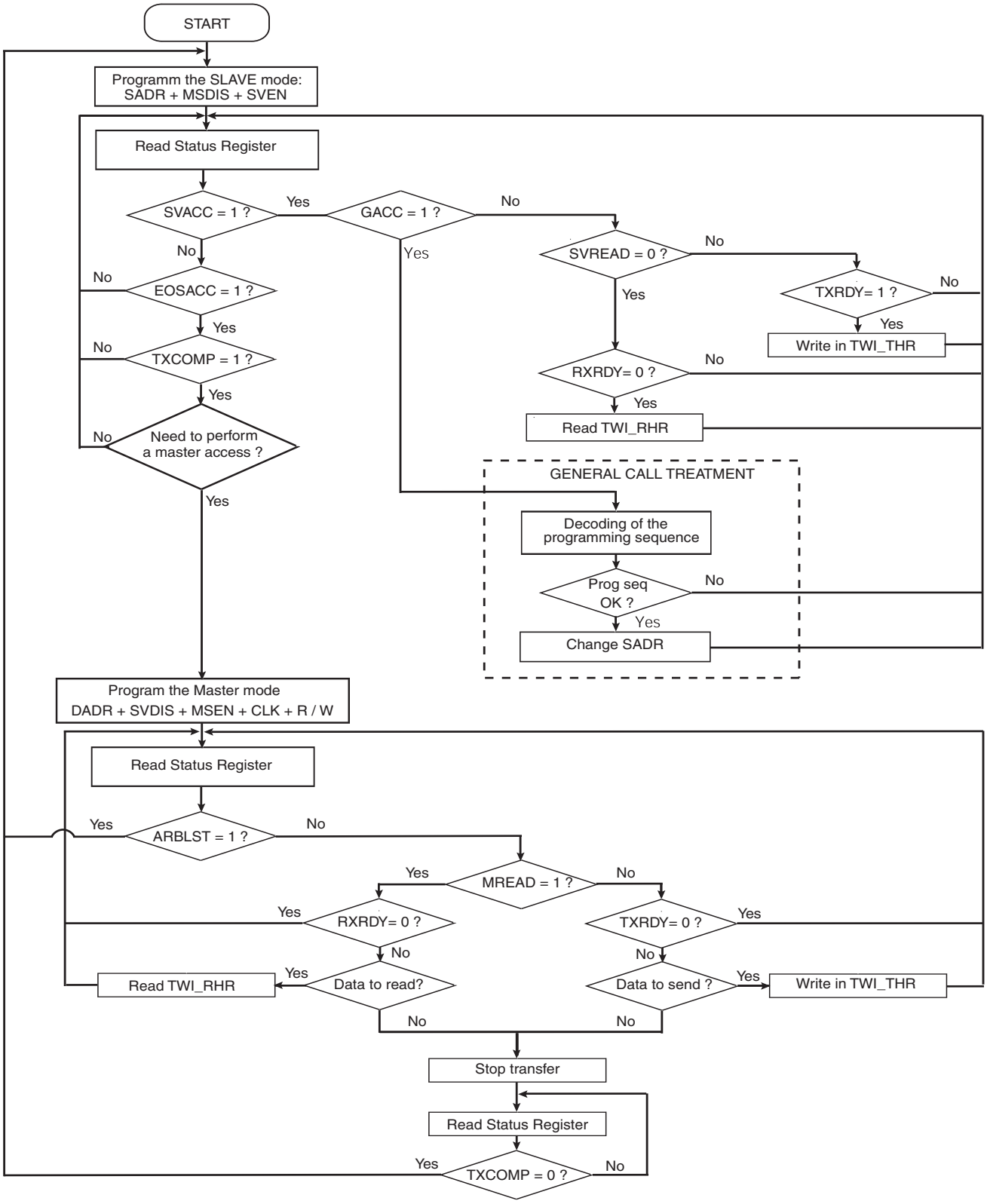


Figure 33-21. Arbitration Cases



The flowchart shown in Figure 33-22 on page 494 gives an example of read and write operations in Multi-master mode.

Figure 33-22. Multi-master Flowchart



33.10 Slave Mode

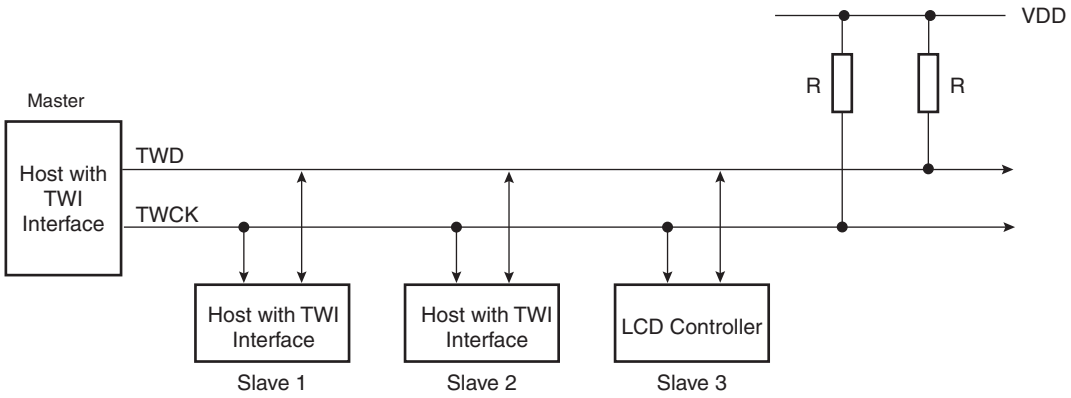
33.10.1 Definition

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).

33.10.2 Application Block Diagram

Figure 33-23. Slave Mode Typical Application Block Diagram



33.10.3 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.

33.10.4 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.

33.10.4.1 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 33-24 on page 497.

33.10.4.2 *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 33-25 on page 497.

33.10.4.3 *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 33-27 on page 499 and Figure 33-28 on page 500.

33.10.4.4 *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 33-26 on page 498.

33.10.4.5

33.10.5 Data Transfer

33.10.5.1 *Read Operation*

The read mode is defined as a data requirement from the master.

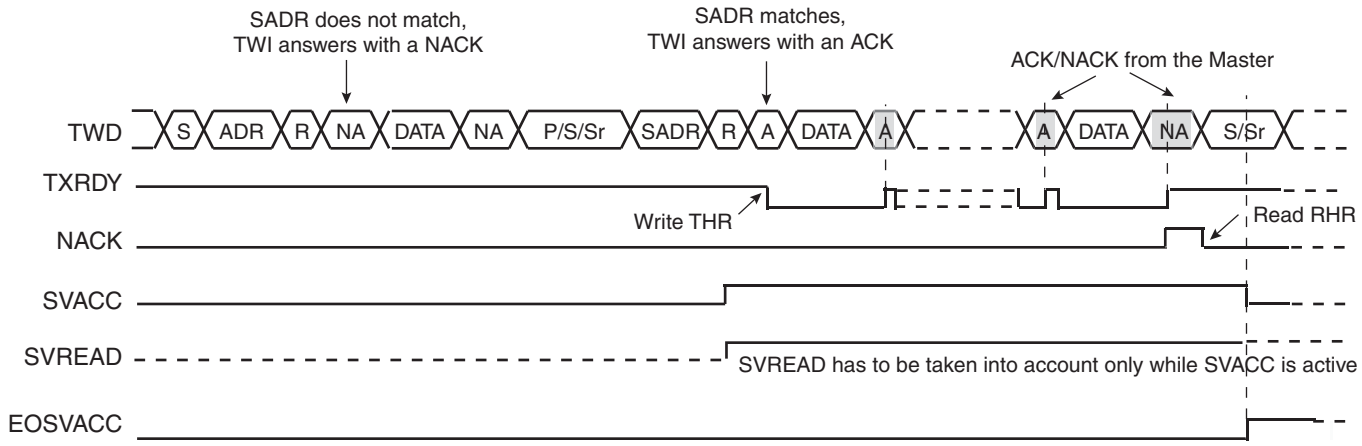
After a START or a REPEATED START condition is detected, the decoding of the address starts. If the slave address (SADR) is decoded, SVACC is set and SVREAD indicates the direction of the transfer.

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 33-24 on page 497 describes the write operation.

Figure 33-24. Read Access Ordered by a MASTER



Notes: 1. When SVACC is low, the state of SVREAD becomes irrelevant.

2. TXRDY is reset when data has been transmitted from TWI_THR to the shift register and set when this data has been acknowledged or non acknowledged.

33.10.5.2 Write Operation

The write mode is defined as a data transmission from the master.

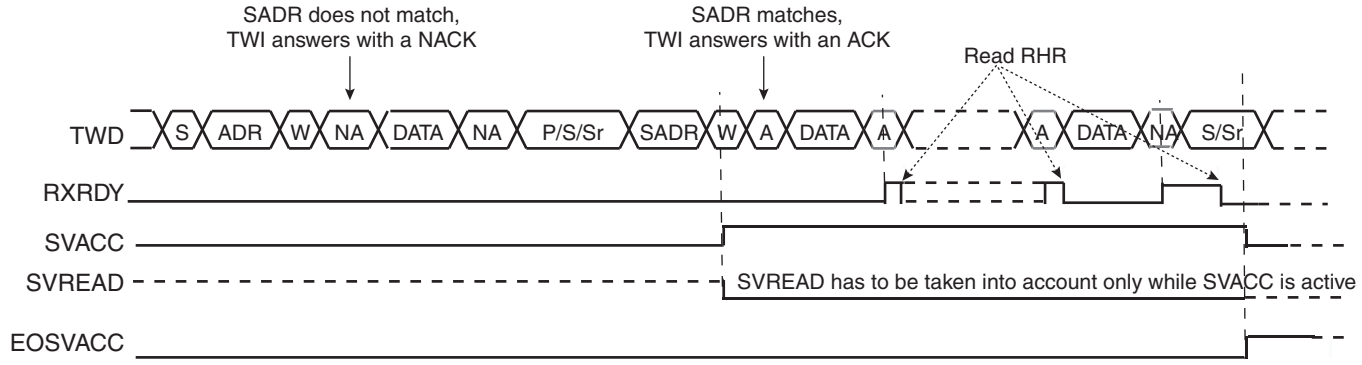
After a START or a REPEATED START, the decoding of the address starts. If the slave address is decoded, SVACC is set and SVREAD indicates the direction of the transfer (SVREAD is low in this case).

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 33-25 on page 497 describes the Write operation.

Figure 33-25. Write Access Ordered by a Master



Notes: 1. When SVACC is low, the state of SVREAD becomes irrelevant.

2. RXRDY is set when data has been transmitted from the shift register to the TWI_RHR and reset when this data is read.

33.10.5.3 General Call

The general call is performed in order to change the address of the slave.

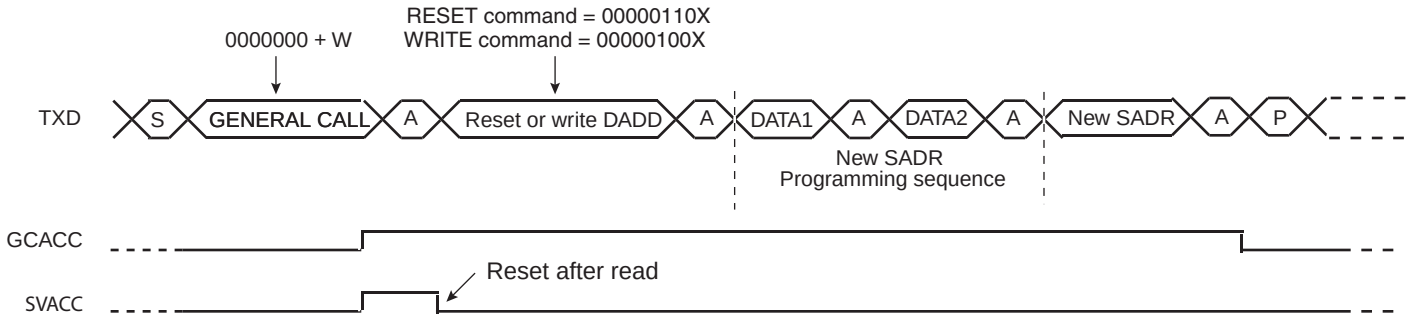
If a GENERAL CALL is detected, GACC is set.

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 33-26 on page 498 describes the General Call access.

Figure 33-26. Master Performs a General Call



Note: This method allows the user to create an own programming sequence by choosing the programming bytes and the number of them. The programming sequence has to be provided to the master.

33.10.5.4 Clock Synchronization

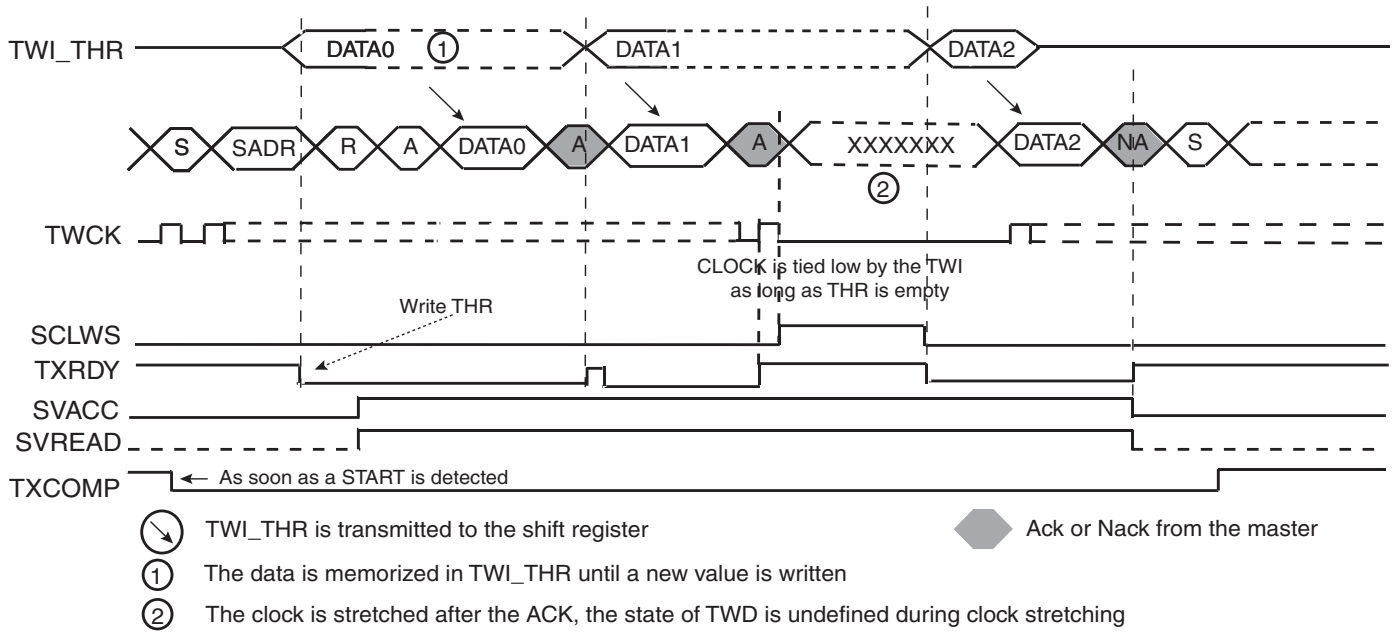
In both read and write modes, it may happen that TWI_THR/TWI_RHR buffer is not filled /emptied before the emission/reception of a new character. In this case, to avoid sending/receiving undesired data, a clock stretching mechanism is implemented.

33.10.5.5 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 33-27 on page 499 describes the clock synchronization in Read mode.

Figure 33-27. Clock Synchronization in Read Mode



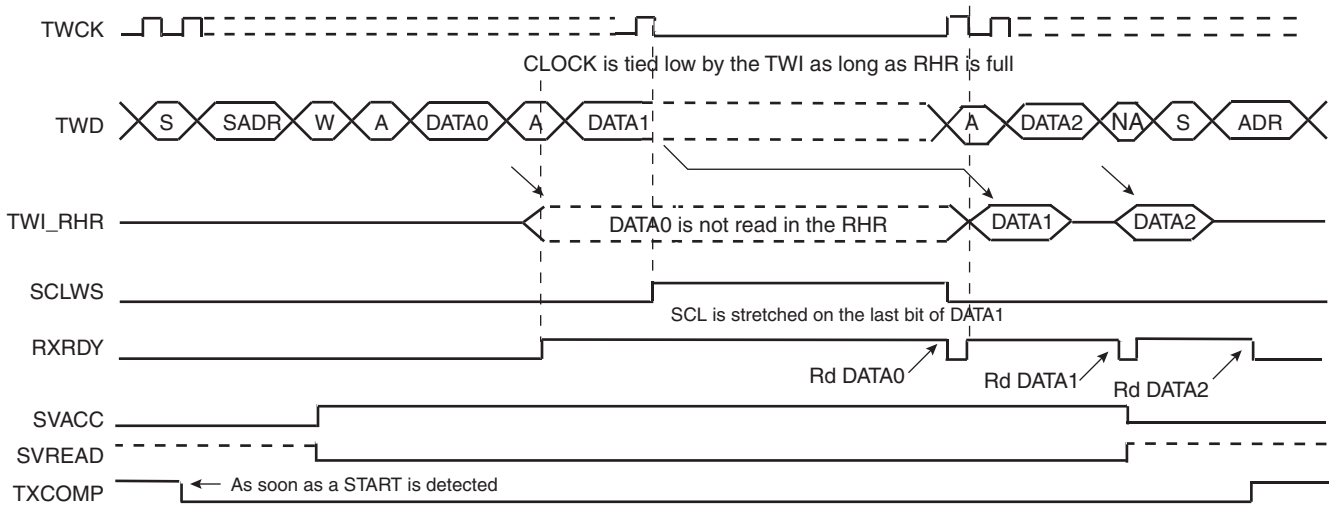
Notes:
1. TXRDY is reset when data has been written in the TWI_TH to the shift register and set when this data has been acknowledged or non acknowledged.
2. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.
3. SCLWS is automatically set when the clock synchronization mechanism is started.

33.10.5.6 *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 33-28 on page 500 describes the clock synchronization in Read mode.

Figure 33-28. Clock Synchronization in Write Mode



Notes:
1. At the end of the read sequence, TXCOMP is set after a STOP or after a REPEATED_START + an address different from SADR.
2. SCLWS is automatically set when the clock synchronization mechanism is started and automatically reset when the mechanism is finished.

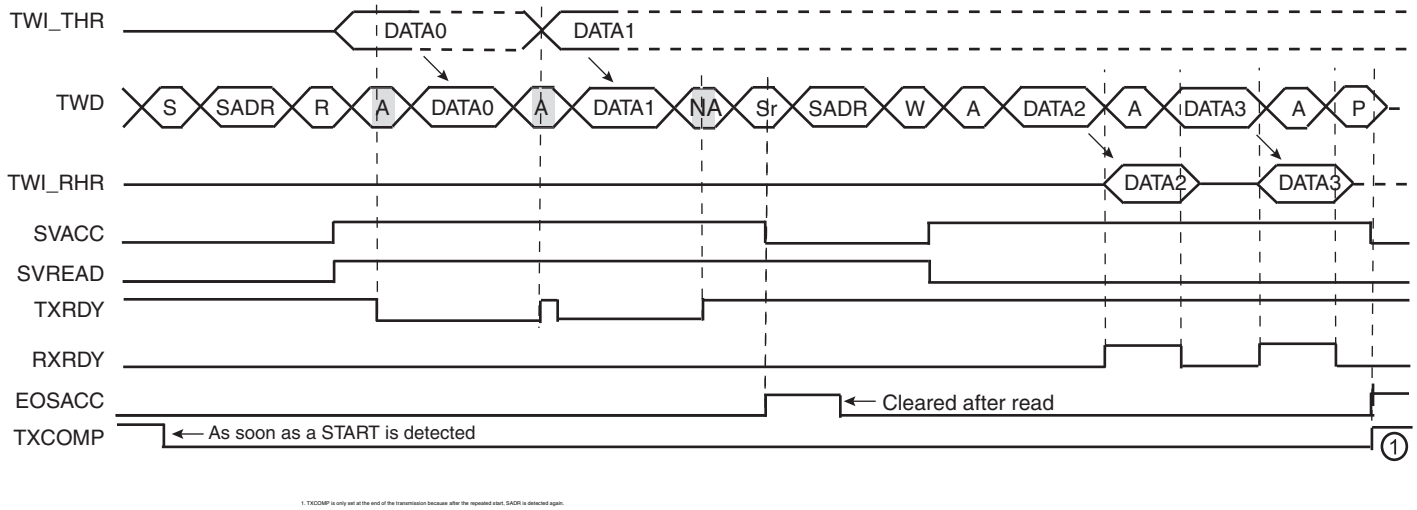
33.10.5.7 Reversal after a Repeated Start

33.10.5.8 Reversal of Read to Write

The master initiates the communication by a read command and finishes it by a write command.

Figure 33-29 on page 501 describes the repeated start + reversal from Read to Write mode.

Figure 33-29. Repeated Start + Reversal from Read to Write Mode

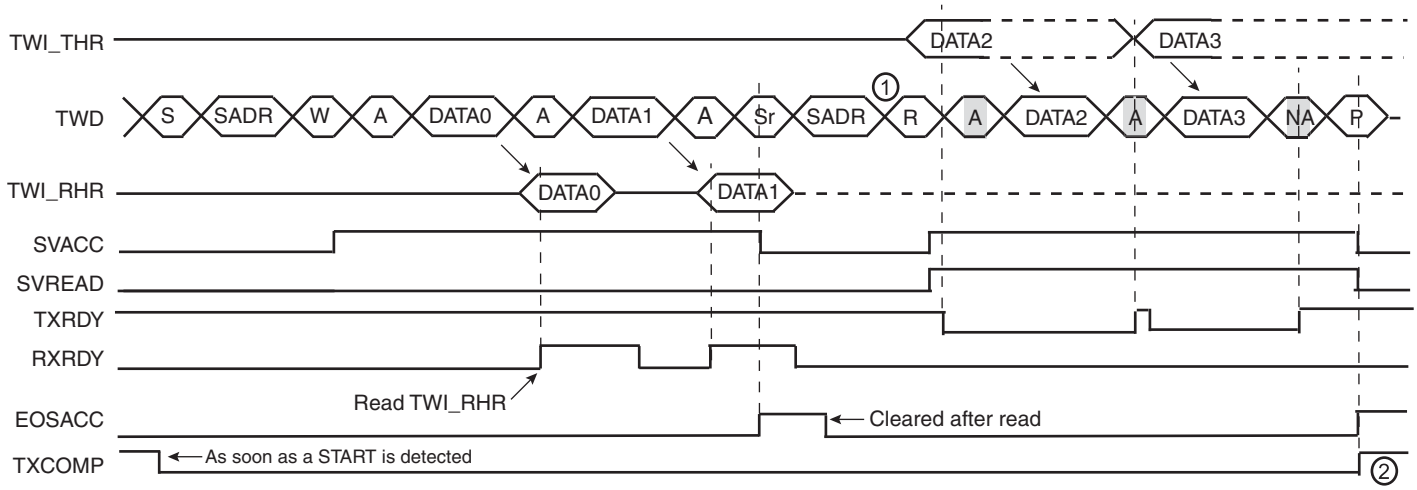


1. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

33.10.5.9 Reversal of Write to Read

The master initiates the communication by a write command and finishes it by a read command.Figure 33-30 on page 501 describes the repeated start + reversal from Write to Read mode.

Figure 33-30. Repeated Start + Reversal from Write to Read Mode

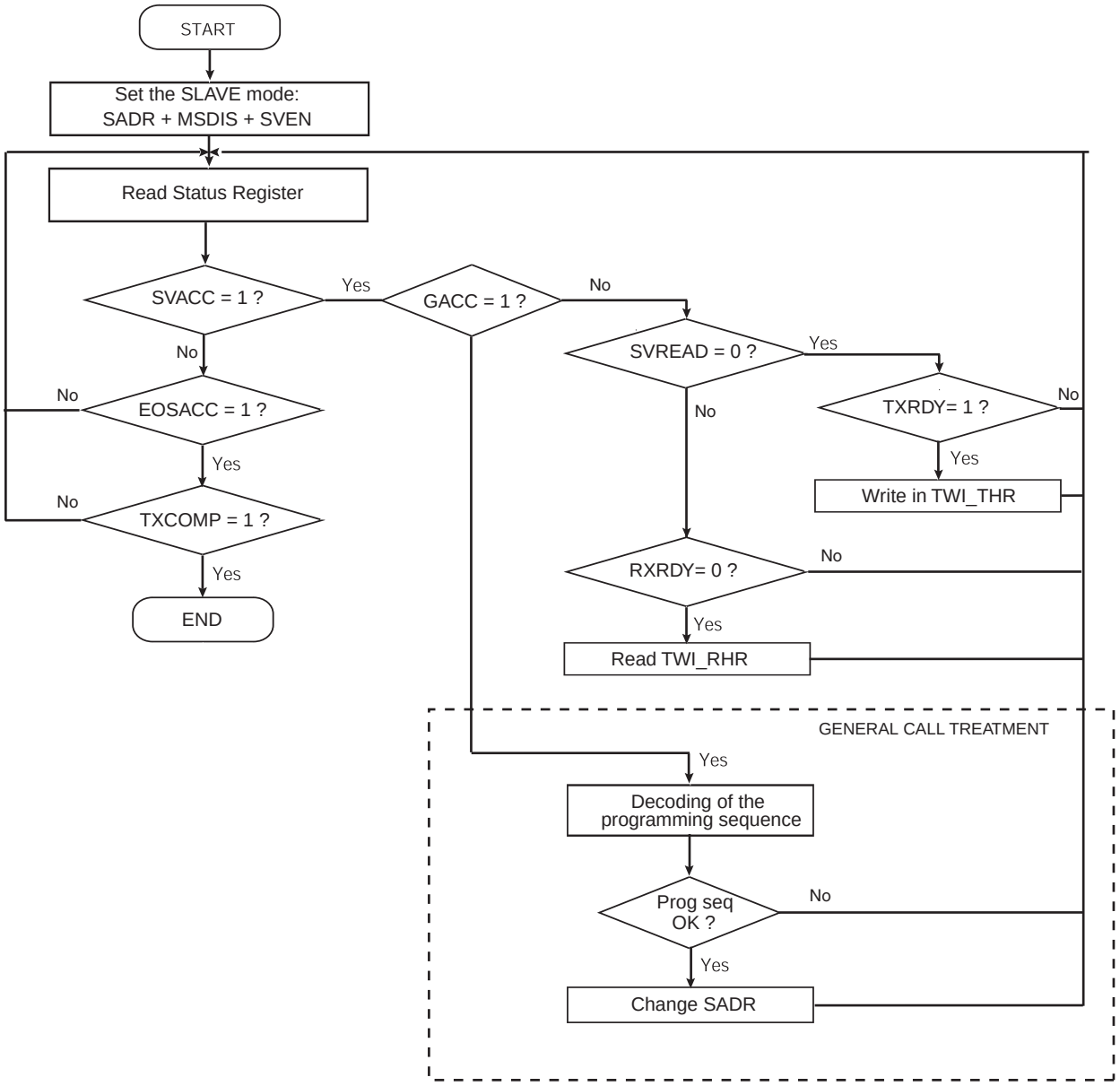


Notes:
1. In this case, if TWI_THR has not been written at the end of the read command, the clock is automatically stretched before the ACK.
2. TXCOMP is only set at the end of the transmission because after the repeated start, SADR is detected again.

33.10.6 Read Write Flowcharts

The flowchart shown in Figure 33-31 on page 502 gives an example of read and write operations in Slave mode. A polling or interrupt method can be used to check the status bits. The interrupt method requires that the interrupt enable register (TWI_IER) be configured first.

Figure 33-31. Read Write Flowchart in Slave Mode



33.11 Two-wire Interface (TWI) User Interface

Table 33-4. 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	–	–	–

33.11.1 TWI Control Register

Name: TWI_CR

Access: Write-only

Reset Value: 0x00000000

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

- **START: Send a START Condition**

0 = No effect.

1 = A frame beginning with a START bit is transmitted according to the features defined in the mode register.

This action is necessary when the TWI peripheral wants to read data from a slave. When configured in Master Mode with a write operation, a frame is sent as soon as the user writes a character in the Transmit Holding Register (TWI_THR).

- **STOP: Send a STOP Condition**

0 = No effect.

1 = STOP Condition is sent just after completing the current byte transmission in master read mode.

  – In single data byte master read, the START and STOP must both be set.
  – In multiple data bytes master read, the STOP must be set after the last data received but one.
  – In master read mode, if a NACK bit is received, the STOP is automatically performed.
  – In multiple data write operation, when both THR and shift register are empty, a STOP condition is automatically sent.

- **MSEN: TWI Master Mode Enabled**

0 = No effect.

1 = If MSDIS = 0, the master mode is enabled.

Note: Switching from Slave to Master mode is only permitted when TXCOMP = 1.

- **MSDIS: TWI Master Mode Disabled**

0 = No effect.

1 = The master mode is disabled, all pending data is transmitted. The shifter and holding characters (if it contains data) are transmitted in case of write operation. In read operation, the character being transferred must be completely received before disabling.

- **SVEN: TWI Slave Mode Enabled**

0 = No effect.

1 = If SVDIS = 0, the slave mode is enabled.

Note: Switching from Master to Slave mode is only permitted when TXCOMP = 1.

- **SVDIS: TWI Slave Mode Disabled**

0 = No effect.

1 = The slave mode is disabled. The shifter and holding characters (if it contains data) are transmitted in case of read operation. In write operation, the character being transferred must be completely received before disabling.

- **SWRST: Software Reset**

0 = No effect.

1 = Equivalent to a system reset.

33.11.2 TWI Master Mode Register

Name: TWI_MMR

Access: Read-write

Reset Value: 0x00000000

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

- **IADRSZ: Internal Device Address Size**

IADRSZ[9:8]		
0	0	No internal device address
0	1	One-byte internal device address
1	0	Two-byte internal device address
1	1	Three-byte internal device address

- **MREAD: Master Read Direction**

0 = Master write direction.

1 = Master read direction.

- **DADR: Device Address**

The device address is used to access slave devices in read or write mode. Those bits are only used in Master mode.

33.11.3 TWI Slave Mode Register

Name: TWI_SMR

Access: Read-write

Reset Value: 0x00000000

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

- **SADR: Slave Address**

The slave device address is used in Slave mode in order to be accessed by master devices in read or write mode.

SADR must be programmed before enabling the Slave mode or after a general call. Writes at other times have no effect.

33.11.4 TWI Internal Address Register

Name: TWI_IADR

Access: Read-write

Reset Value: 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.

33.11.5 TWI Clock Waveform Generator Register

Name: TWI_CWGR

Access: Read-write

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
					CKDIV		

15	14	13	12	11	10	9	8
CHDIV							

7	6	5	4	3	2	1	0
CLDIV							

TWI_CWGR is only used in Master mode.

- **CLDIV: Clock Low Divider**

The SCL low period is defined as follows:

$$T_{low} = ((\text{CLDIV} \times 2^{\text{CKDIV}}) + 4) \times T_{MCK}$$

- **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.

33.11.6 TWI Status Register

Name: TWI_SR

Access: Read-only

Reset Value: 0x0000F009

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
				EOSACC	SCLWS	ARBLST	NACK
7	6	5	4	3	2	1	0
–	OVRE	GACC	SVACC	SVREAD	TXRDY	RXRDY	TXCOMP

- **TXCOMP: Transmission Completed (automatically set / reset)**

TXCOMP used in Master mode:

0 = During the length of the current frame.

1 = When both holding and shifter registers are empty and STOP condition has been sent.

TXCOMP behavior in Master mode can be seen in Figure 33-8 on page 482 and in Figure 33-10 on page 483.

TXCOMP used in Slave mode:

0 = As soon as a Start is detected.

1 = After a Stop or a Repeated Start + an address different from SADR is detected.

TXCOMP behavior in Slave mode can be seen in Figure 33-27 on page 499, Figure 33-28 on page 500, Figure 33-29 on page 501 and Figure 33-30 on page 501.

- **RXRDY: Receive Holding Register Ready (automatically set / reset)**

0 = No character has been received since the last TWI_RHR read operation.

1 = A byte has been received in the TWI_RHR since the last read.

RXRDY behavior in Master mode can be seen in Figure 33-10 on page 483.

RXRDY behavior in Slave mode can be seen in Figure 33-25 on page 497, Figure 33-28 on page 500, Figure 33-29 on page 501 and Figure 33-30 on page 501.

- **TXRDY: Transmit Holding Register Ready (automatically set / reset)**

TXRDY used in Master mode:

0 = The transmit holding register has not been transferred into shift register. Set to 0 when writing into TWI_THR register.

1 = As soon as a data byte is transferred from TWI_THR to internal shifter or if a NACK error is detected, TXRDY is set at the same time as TXCOMP and NACK. TXRDY is also set when MSEN is set (enable TWI).

TXRDY behavior in Master mode can be seen in Figure 33-8 on page 482.

TXRDY used in Slave mode:

0 = As soon as data is written in the TWI_THR, until this data has been transmitted and acknowledged (ACK or NACK).

1 = It indicates that the TWI_THR is empty and that data has been transmitted and acknowledged.

If TXRDY is high and if a NACK has been detected, the transmission will be stopped. Thus when TRDY = NACK = 1, the programmer must not fill TWI_THR to avoid losing it.

TXRDY behavior in Slave mode can be seen in Figure 33-24 on page 497, Figure 33-27 on page 499, Figure 33-29 on page 501 and Figure 33-30 on page 501.

- **SVREAD: Slave Read (automatically set / reset)**

This bit is only used in Slave mode. When SVACC is low (no Slave access has been detected) SVREAD is irrelevant.

0 = Indicates that a write access is performed by a Master.

1 = Indicates that a read access is performed by a Master.

SVREAD behavior can be seen in Figure 33-24 on page 497, Figure 33-25 on page 497, Figure 33-29 on page 501 and Figure 33-30 on page 501.

- **SVACC: Slave Access (automatically set / reset)**

This bit is only used in Slave mode.

0 = TWI is not addressed. SVACC is automatically cleared after a NACK or a STOP condition is detected.

1 = Indicates that the address decoding sequence has matched (A Master has sent SADR). SVACC remains high until a NACK or a STOP condition is detected.

SVACC behavior can be seen in Figure 33-24 on page 497, Figure 33-25 on page 497, Figure 33-29 on page 501 and Figure 33-30 on page 501.

- **GACC: General Call Access (clear on read)**

This bit is only used in Slave mode.

0 = No General Call has been detected.

1 = A General Call has been detected. After the detection of General Call, the programmer decoded the commands that follow and the programming sequence.

GACC behavior can be seen in Figure 33-26 on page 498.

- **OVRE: Overrun Error (clear on read)**

This bit is only used in Master mode.

0 = TWI_RHR has not been loaded while RXRDY was set

1 = TWI_RHR has been loaded while RXRDY was set. Reset by read in TWI_SR when TXCOMP is set.

- **NACK: Not Acknowledged (clear on read)**

NACK used in Master mode:

0 = Each data byte has been correctly received by the far-end side TWI slave component.

1 = A data byte has not been acknowledged by the slave component. Set at the same time as TXCOMP.

NACK used in Slave Read mode:

0 = Each data byte has been correctly received by the Master.

1 = In read mode, a data byte has not been acknowledged by the Master. When NACK is set the programmer must not fill TWI_THR even if TXRDY is set, because it means that the Master will stop the data transfer or re initiate it.

Note that in Slave Write mode all data are acknowledged by the TWI.

- **ARBLST: Arbitration Lost (clear on read)**

This bit is only used in Master mode.

0: Arbitration won.

1: Arbitration lost. Another master of the TWI bus has won the multi-master arbitration. TXCOMP is set at the same time.

- **SCLWS: Clock Wait State (automatically set / reset)**

This bit is only used in Slave mode.

0 = The clock is not stretched.

1 = The clock is stretched. TWI_THR / TWI_RHR buffer is not filled / emptied before the emission / reception of a new character.

SCLWS behavior can be seen in Figure 33-27 on page 499 and Figure 33-28 on page 500.

- **EOSACC: End Of Slave Access (clear on read)**

This bit is only used in Slave mode.

0 = A slave access is being performing.

1 = The Slave Access is finished. End Of Slave Access is automatically set as soon as SVACC is reset.

EOSACC behavior can be seen in Figure 33-29 on page 501 and Figure 33-30 on page 501

33.11.7 TWI Interrupt Enable Register

Name: TWI_IER

Access: Write-only

Reset Value: 0x00000000

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

0 = No effect.

1 = Enables the corresponding interrupt.

33.11.8 TWI Interrupt Disable Register

Name: TWI_IDR

Access: Write-only

Reset Value: 0x00000000

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

0 = No effect.

1 = Disables the corresponding interrupt.

33.11.9 TWI Interrupt Mask Register

Name: TWI_IMR

Access: Read-only

Reset Value: 0x00000000

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

0 = The corresponding interrupt is disabled.

1 = The corresponding interrupt is enabled.

33.11.10 TWI Receive Holding Register

Name: TWI_RHR

Access: Read-only

Reset Value: 0x00000000

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

- **RXDATA: Master or Slave Receive Holding Data**

33.11.11 TWI Transmit Holding Register

Name: TWI_THR

Access: Read-write

Reset Value: 0x00000000

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

- **TXDATA: Master or Slave Transmit Holding Data**

34. Universal Synchronous Asynchronous Receiver Transmitter (USART)

34.1 Overview

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.

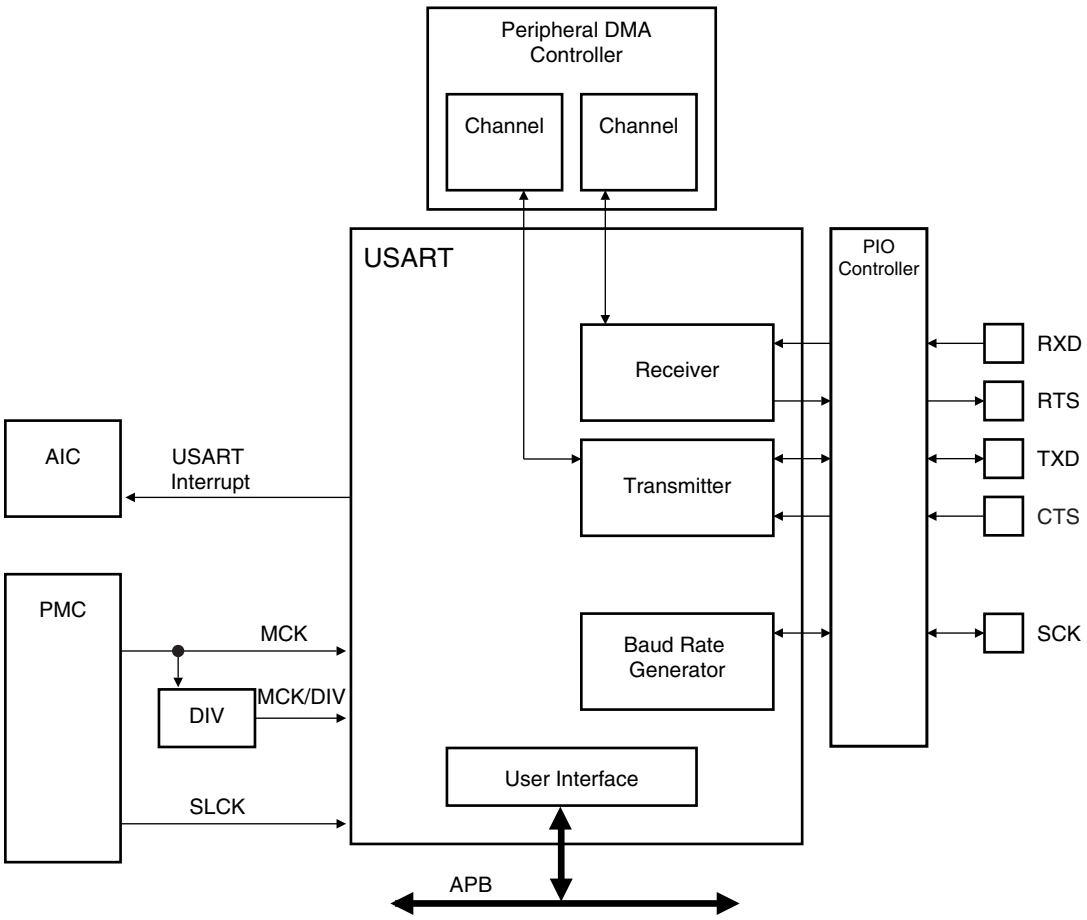
The USART features three test modes: remote loopback, local loopback and automatic echo.

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.

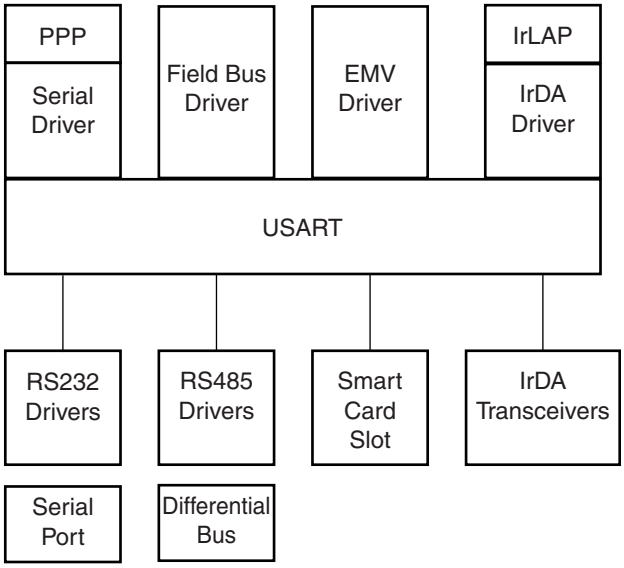
34.2 Block Diagram

Figure 34-1. USART Block Diagram



34.3 Application Block Diagram

Figure 34-2. Application Block Diagram



34.4 I/O Lines Description

Table 34-1. I/O Line Description

Name	Description	Type	Active Level
SCK	Serial Clock	I/O	
TXD	Transmit Serial Data	I/O	
RXD	Receive Serial Data	Input	
CTS	Clear to Send	Input	Low
RTS	Request to Send	Output	Low

34.5 Product Dependencies

34.5.1 I/O Lines

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

To prevent the TXD line from falling when the USART is disabled, the use of an internal pull up is mandatory. If the hardware handshaking feature is used, the internal pull up on TXD must also be enabled.

34.5.2 Power Management

The USART is not continuously clocked. The programmer must first enable the USART Clock in the Power Management Controller (PMC) before using the USART. However, if the application does not require USART operations, the USART clock can be stopped when not needed and be restarted later. In this case, the USART will resume its operations where it left off.

Configuring the USART does not require the USART clock to be enabled.

34.5.3 Interrupt

The USART interrupt line is connected on one of the internal sources of the Advanced Interrupt Controller. Using the USART interrupt requires the AIC to be programmed first. Note that it is not recommended to use the USART interrupt line in edge sensitive mode.

34.6 Functional Description

The USART is capable of managing several types of serial synchronous or asynchronous communications.

It supports the following communication modes:

- 5- to 9-bit full-duplex asynchronous serial communication
  - MSB- or LSB-first
  - 1, 1.5 or 2 stop bits
  - Parity even, odd, marked, space or none
  - By 8 or by 16 over-sampling receiver frequency
  - Optional hardware handshaking
  - Optional 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

34.6.1 Baud Rate Generator

The Baud Rate Generator provides the bit period clock named the Baud Rate Clock to both the receiver and the transmitter.

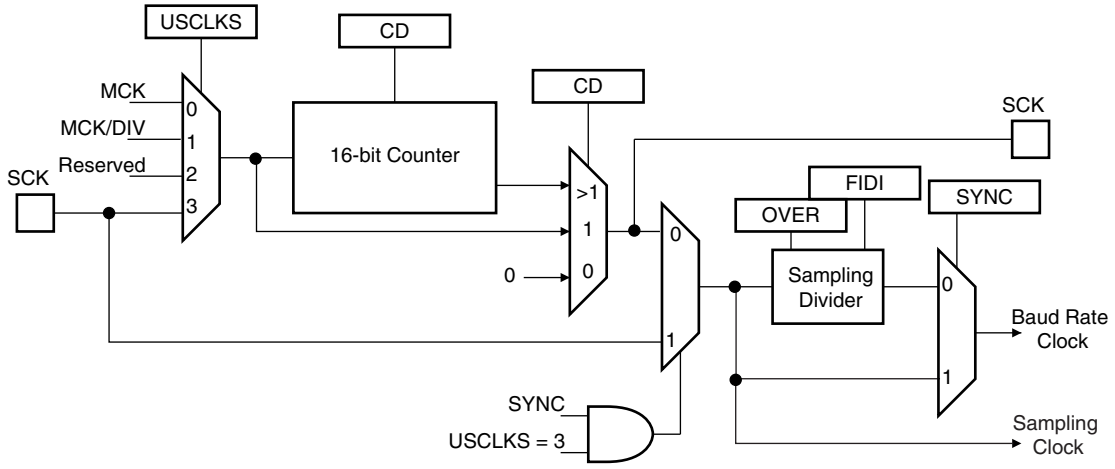
The Baud Rate Generator clock source can be selected by setting the USCLKS field in the Mode Register (US_MR) between:

- the Master Clock MCK
- a division of the Master Clock, the divider being product dependent, but generally set to 8
- the external clock, available on the SCK pin

The Baud Rate Generator is based upon a 16-bit divider, which is programmed with the CD field of the Baud Rate Generator Register (US_BRGR). If CD is programmed at 0, the Baud Rate Generator does not generate any clock. If CD is programmed at 1, the divider is bypassed and becomes inactive.

If the external SCK clock is selected, the duration of the low and high levels of the signal provided on the SCK pin must be longer than a Master Clock (MCK) period. The frequency of the signal provided on SCK must be at least 4.5 times lower than MCK.

Figure 34-3. Baud Rate Generator



34.6.1.1 Baud Rate in Asynchronous Mode

If the USART is programmed to operate in asynchronous mode, the selected clock is first divided by CD, which is field programmed in the Baud Rate Generator Register (US_BRGR). The resulting clock is provided to the receiver as a sampling clock and then divided by 16 or 8, depending on the programming of the OVER bit in US_MR.

If OVER is set to 1, the receiver sampling is 8 times higher than the baud rate clock. If OVER is cleared, the sampling is performed at 16 times the baud rate clock.

The following formula performs the calculation of the Baud Rate.

$$Baudrate = \frac{SelectedClock}{(8(2-Over)CD)}$$

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.

34.6.1.2 Baud Rate Calculation Example

Table 34-2 shows calculations of CD to obtain a baud rate at 38400 bauds for different source clock frequencies. This table also shows the actual resulting baud rate and the error.

Table 34-2. Baud Rate Example (OVER = 0)

Source Clock	Expected Baud Rate	Calculation Result	CD	Actual Baud Rate	Error
MHz	Bit/s			Bit/s	
3 686 400	38 400	6.00	6	38 400.00	0.00%
4 915 200	38 400	8.00	8	38 400.00	0.00%
5 000 000	38 400	8.14	8	39 062.50	1.70%
7 372 800	38 400	12.00	12	38 400.00	0.00%
8 000 000	38 400	13.02	13	38 461.54	0.16%
12 000 000	38 400	19.53	20	37 500.00	2.40%
12 288 000	38 400	20.00	20	38 400.00	0.00%
14 318 180	38 400	23.30	23	38 908.10	1.31%
14 745 600	38 400	24.00	24	38 400.00	0.00%
18 432 000	38 400	30.00	30	38 400.00	0.00%
24 000 000	38 400	39.06	39	38 461.54	0.16%
24 576 000	38 400	40.00	40	38 400.00	0.00%
25 000 000	38 400	40.69	40	38 109.76	0.76%
32 000 000	38 400	52.08	52	38 461.54	0.16%
32 768 000	38 400	53.33	53	38 641.51	0.63%
33 000 000	38 400	53.71	54	38 194.44	0.54%
40 000 000	38 400	65.10	65	38 461.54	0.16%
50 000 000	38 400	81.38	81	38 580.25	0.47%

The baud rate is calculated with the following formula:

$$BaudRate = MCK / (CD \times 16)$$

The baud rate error is calculated with the following formula. It is not recommended to work with an error higher than 5%.

$$Error = 1 - \left(\frac{ExpectedBaudRate}{ActualBaudRate}\right)$$

34.6.1.3 Fractional Baud Rate in Asynchronous Mode

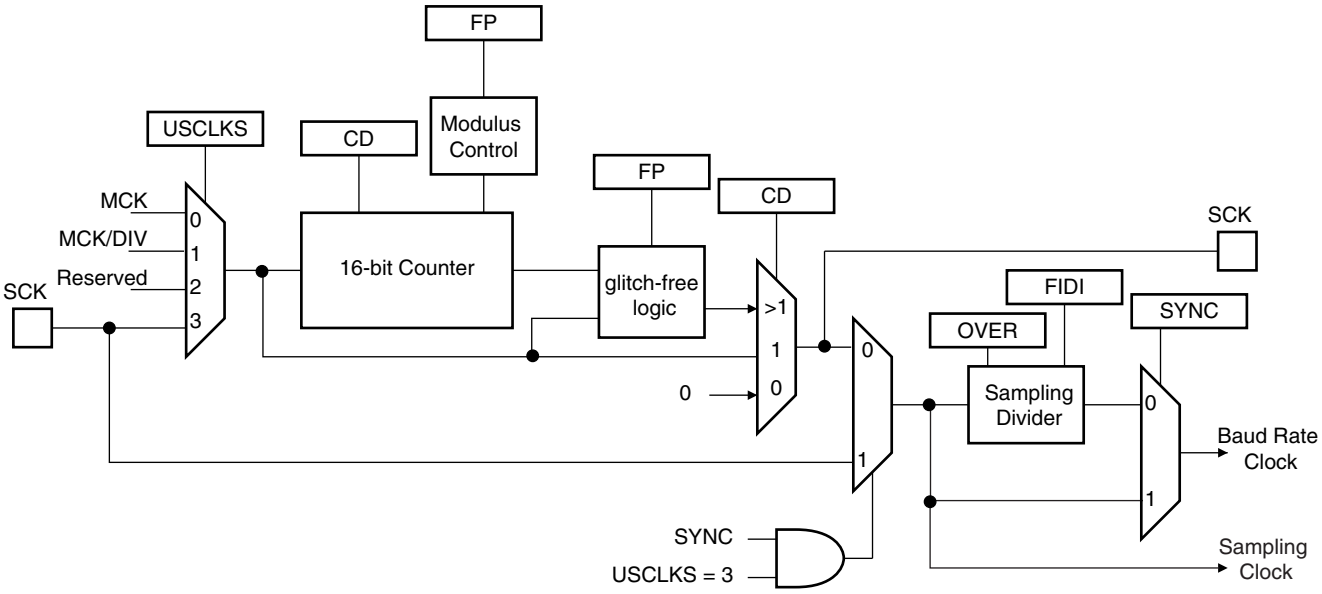
The Baud Rate generator previously defined is subject to the following limitation: the output frequency changes by only integer multiples of the reference frequency. An approach to this problem is to integrate a fractional N clock generator that has a high resolution. The generator architecture is modified 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 34-4. Fractional Baud Rate Generator



34.6.1.4 *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.

34.6.1.5 *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 34-3.

Table 34-3. Binary and Decimal Values for Di

DI field	0001	0010	0011	0100	0101	0110	1000	1001
Di (decimal)	1	2	4	8	16	32	12	20

Fi is a binary value encoded on a 4-bit field, named FI, as represented in Table 34-4.

Table 34-4. Binary and Decimal Values for Fi

FI field	0000	0001	0010	0011	0100	0101	0110	1001	1010	1011	1100	1101
Fi (decimal	372	372	558	744	1116	1488	1860	512	768	1024	1536	2048

Table 34-5 shows the resulting Fi/Di Ratio, which is the ratio between the ISO7816 clock and the baud rate clock.

Table 34-5. Possible Values for the Fi/Di Ratio

Fi/Di	372	558	774	1116	1488	1806	512	768	1024	1536	2048
1	372	558	744	1116	1488	1860	512	768	1024	1536	2048
2	186	279	372	558	744	930	256	384	512	768	1024
4	93	139.5	186	279	372	465	128	192	256	384	512
8	46.5	69.75	93	139.5	186	232.5	64	96	128	192	256
16	23.25	34.87	46.5	69.75	93	116.2	32	48	64	96	128
32	11.62	17.43	23.25	34.87	46.5	58.13	16	24	32	48	64
12	31	46.5	62	93	124	155	42.66	64	85.33	128	170.6
20	18.6	27.9	37.2	55.8	74.4	93	25.6	38.4	51.2	76.8	102.4

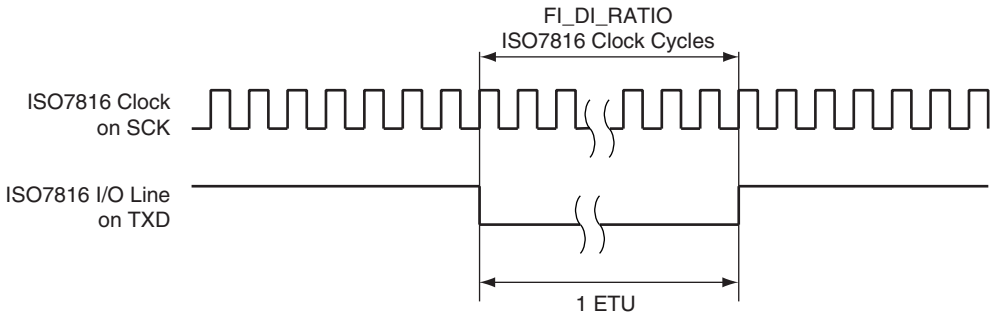
If the USART is configured in ISO7816 Mode, the clock selected by the USCLKS field in the Mode Register (US_MR) is first divided by the value programmed in the field CD in the Baud Rate Generator Register (US_BRGR). The resulting clock can be provided to the SCK pin to feed the smart card clock inputs. This means that the CLKO bit can be set in US_MR.

This clock is then divided by the value programmed in the FI_DI_RATIO field in the FI_DI_Ratio register (US_FIDI). This is performed by the Sampling Divider, which performs a division by up to 2047 in ISO7816 Mode. The non-integer values of the Fi/Di Ratio are not supported and the user must program the FI_DI_RATIO field to a value as close as possible to the expected value.

The FI_DI_RATIO field resets to the value 0x174 (372 in decimal) and is the most common divider between the ISO7816 clock and the bit rate (Fi = 372, Di = 1).

Figure 34-5 shows the relation between the Elementary Time Unit, corresponding to a bit time, and the ISO 7816 clock.

Figure 34-5. Elementary Time Unit (ETU)



34.6.2 Receiver and Transmitter Control

After reset, the receiver is disabled. The user must enable the receiver by setting the RXEN bit in the Control Register (US_CR). However, the receiver registers can be programmed before the receiver clock is enabled.

After reset, the transmitter is disabled. The user must enable it by setting the TXEN bit in the Control Register (US_CR). However, the transmitter registers can be programmed before being enabled.

The Receiver and the Transmitter can be enabled together or independently.

At any time, the software can perform a reset on the receiver or the transmitter of the USART by setting the corresponding bit, RSTRX and RSTTX respectively, in the Control Register (US_CR). The software resets clear the status flag and reset internal state machines but the user interface configuration registers hold the value configured prior to software reset. Regardless of what the receiver or the transmitter is performing, the communication is immediately stopped.

The user can also independently disable the receiver or the transmitter by setting RXDIS and TXDIS respectively in US_CR. If the receiver is disabled during a character reception, the USART waits until the end of reception of the current character, then the reception is stopped. If the transmitter is disabled while it is operating, the USART waits the end of transmission of both the current character and character being stored in the Transmit Holding Register (US_THR). If a timeguard is programmed, it is handled normally.

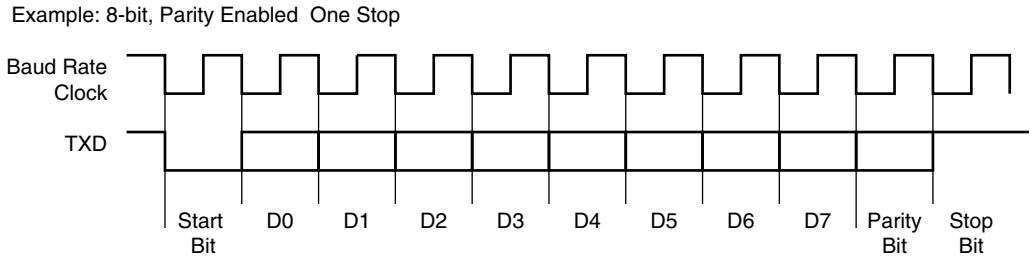
34.6.3 Synchronous and Asynchronous Modes

34.6.3.1 Transmitter Operations

The transmitter performs the same in both synchronous and asynchronous operating modes (SYNC = 0 or SYNC = 1). One start bit, up to 9 data bits, one optional parity bit and up to two stop bits are successively shifted out on the TXD pin at each falling edge of the programmed serial clock.

The number of data bits is selected by the CHRL field and the MODE 9 bit in the Mode Register (US_MR). Nine bits are selected by setting the MODE 9 bit regardless of the CHRL field. The parity bit is set according to the PAR field in US_MR. The even, odd, space, marked or none parity bit can be configured. The MSBF field in US_MR configures which data bit is sent first. If written at 1, the most significant bit is sent first. At 0, the less significant bit is sent first. The number of stop bits is selected by the NBSTOP field in US_MR. The 1.5 stop bit is supported in asynchronous mode only.

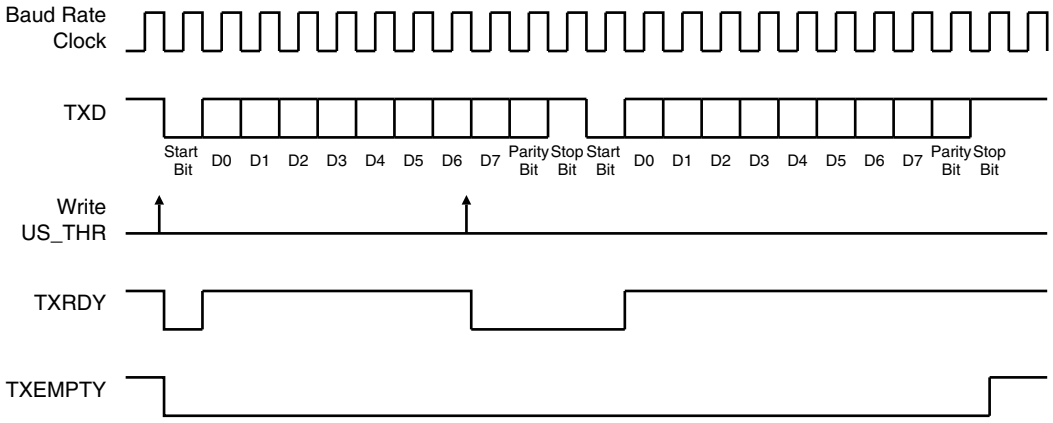
Figure 34-6. Character Transmit



The characters are sent by writing in the Transmit Holding Register (US_THR). The transmitter reports two status bits in the Channel Status Register (US_CSR): TXRDY (Transmitter Ready), which indicates that US_THR is empty and TXEMPTY, which indicates that all the characters written in US_THR have been processed. When the current character processing is completed, the last character written in US_THR is transferred into the Shift Register of the transmitter and US_THR becomes empty, thus TXRDY 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.

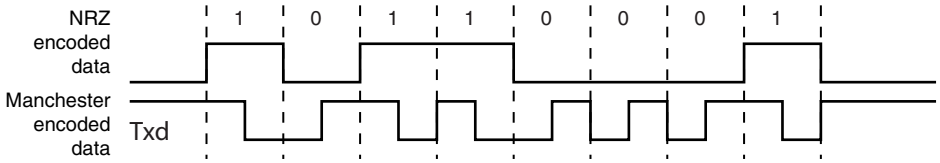
Figure 34-7. Transmitter Status



34.6.3.2 Manchester Encoder

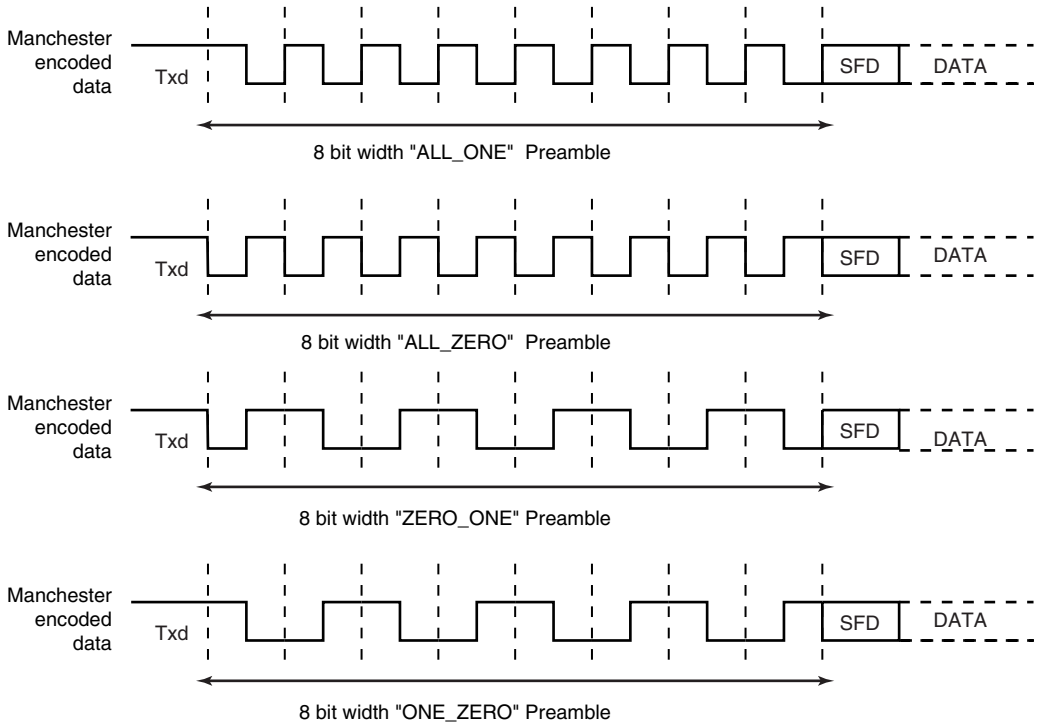
When the Manchester encoder is in use, characters transmitted through the USART are encoded based on biphase Manchester II format. To enable this mode, set the MAN field in the US_MR register to 1. Depending on polarity configuration, a logic level (zero or one), is transmitted as a coded signal one-to-zero or zero-to-one. Thus, a transition always occurs at the midpoint of each bit time. It consumes more bandwidth than the original NRZ signal (2x) but the receiver has more error control since the expected input must show a change at the center of a bit cell. An example of Manchester encoded sequence is: the byte 0xB1 or 10110001 encodes to 10 01 10 10 01 01 01 10, assuming the default polarity of the encoder. Figure 34-8 illustrates this coding scheme.

Figure 34-8. NRZ to Manchester Encoding



The Manchester encoded character can also be encapsulated by adding both a configurable preamble and a start frame delimiter pattern. Depending on the configuration, the preamble is a training sequence, composed of a pre-defined pattern with a programmable length from 1 to 15 bit times. If the preamble length is set to 0, the preamble waveform is not generated prior to any character. The preamble pattern is chosen among the following sequences: ALL_ONE, ALL_ZERO, ONE_ZERO or ZERO_ONE, writing the field TX_PP in the US_MAN register, the field TX_PL is used to configure the preamble length. Figure 34-9 illustrates and defines the valid patterns. To improve flexibility, the encoding scheme can be configured using the TX_MPOL field in the US_MAN register. If the TX_MPOL field is set to zero (default), a logic zero is encoded with a zero-to-one transition and a logic one is encoded with a one-to-zero transition. If the TX_MPOL field is set to one, a logic one is encoded with a one-to-zero transition and a logic zero is encoded with a zero-to-one transition.

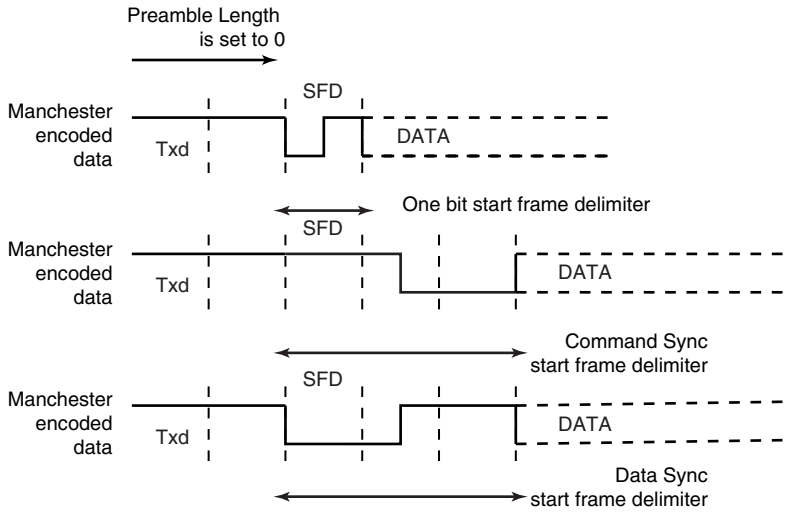
Figure 34-9. Preamble Patterns, Default Polarity Assumed



A start frame delimiter is to be configured using the ONEBIT field in the US_MR register. It consists of a user-defined pattern that indicates the beginning of a valid data. Figure 34-10 illustrates these patterns. If the start frame delimiter, also known as start bit, is one bit, (ONEBIT at 1), a logic zero is Manchester encoded and indicates that a new character is being sent serially on the line. If the start frame delimiter is a synchronization pattern also referred to as sync (ONEBIT at 0), a sequence of 3 bit times is sent serially on the line to indicate the start of a new character. The sync waveform is in itself an invalid Manchester waveform as the transition

occurs at the middle of the second bit time. Two distinct sync patterns are used: the command sync and the data sync. The command sync has a logic one level for one and a half bit times, then a transition to logic zero for the second one and a half bit times. If the MODSYNC field in the US_MR register is set to 1, the next character is a command. If it is set to 0, the next character is a data. When direct memory access is used, the MODSYNC field can be immediately updated with a modified character located in memory. To enable this mode, VAR_SYNC field in US_MR register must be set to 1. In this case, the MODSYNC field in US_MR is bypassed and the sync configuration is held in the TXSYNH in the US_THR register. The USART character format is modified and includes sync information.

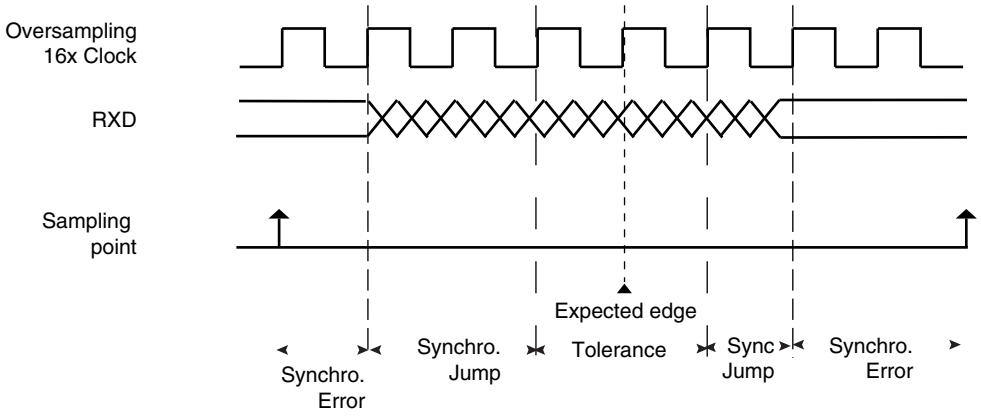
Figure 34-10. Start Frame Delimiter



34.6.3.3 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 34-11. Bit Resynchronization



34.6.3.4 Asynchronous Receiver

If the USART is programmed in asynchronous operating mode (SYNC = 0), the receiver oversamples the RXD input line. The oversampling is either 16 or 8 times the Baud Rate clock, depending on the OVER bit in the Mode Register (US_MR).

The receiver samples the RXD line. If the line is sampled during one half of a bit time at 0, a start bit is detected and data, parity and stop bits are successively sampled on the bit rate clock.

If the oversampling is 16, (OVER at 0), a start is detected at the eighth sample at 0. Then, data bits, parity bit and stop bit are sampled on each 16 sampling clock cycle. If the oversampling is 8 (OVER at 1), a start bit is detected at the fourth sample at 0. Then, data bits, parity bit and stop bit are sampled on each 8 sampling clock cycle.

The number of data bits, first bit sent and parity mode are selected by the same fields and bits as the transmitter, i.e. respectively CHRL, MODE9, MSBF and PAR. For the synchronization mechanism **only**, the number of stop bits has no effect on the receiver as it considers only one stop bit, regardless of the field NBSTOP, so that resynchronization between the receiver and the transmitter can occur. Moreover, as soon as the stop bit is sampled, the receiver starts looking for a new start bit so that resynchronization can also be accomplished when the transmitter is operating with one stop bit.

Figure 34-12 and Figure 34-13 illustrate start detection and character reception when USART operates in asynchronous mode.

Figure 34-12. Asynchronous Start Detection

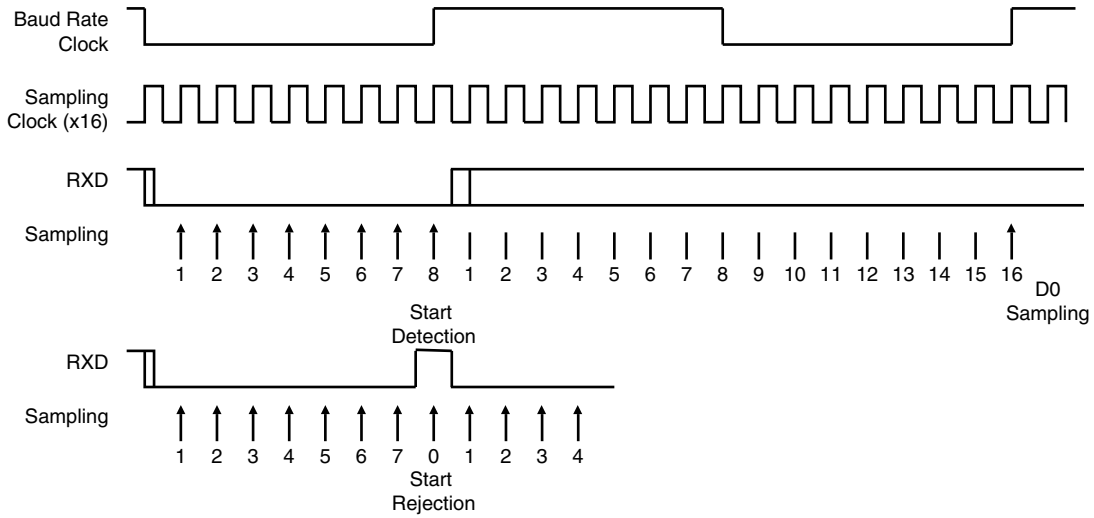
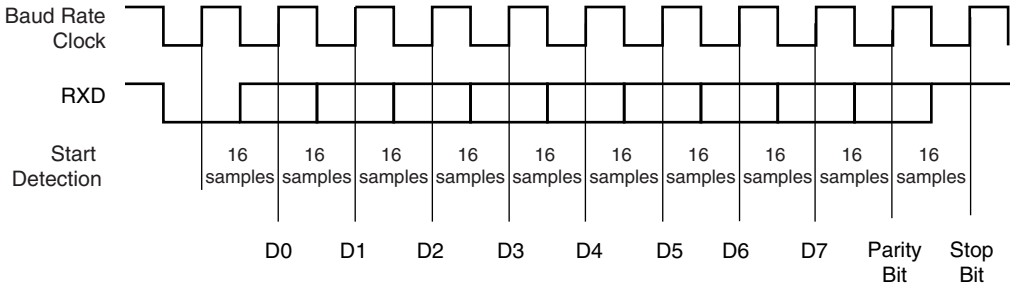


Figure 34-13. Asynchronous Character Reception

Example: 8-bit, Parity Enabled



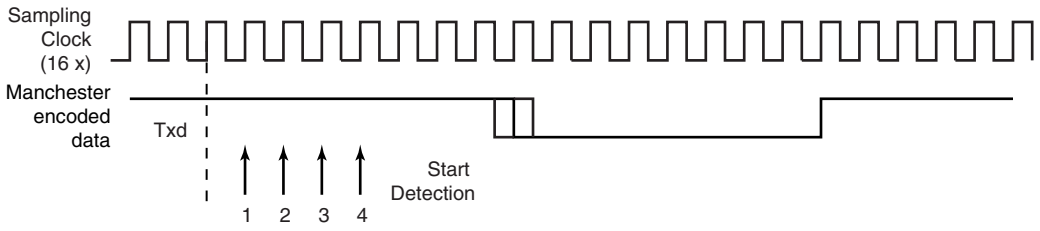
34.6.3.5 Manchester Decoder

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 34-9 for available preamble patterns.

Unlike preamble, the start frame delimiter is shared between Manchester Encoder and Decoder. So, if ONEBIT field is set to 1, only a zero encoded Manchester can be detected as a valid start frame delimiter. If ONEBIT is set to 0, only a sync pattern is detected as a valid start frame delimiter. Decoder operates by detecting transition on incoming stream. If RXD is sampled during one quarter of a bit time at zero, a start bit is detected. See Figure 34-14. The sample pulse rejection mechanism applies.

Figure 34-14. Asynchronous Start Bit Detection



The receiver is activated and starts Preamble and Frame Delimiter detection, sampling the data at one quarter and then three quarters. If a valid preamble pattern or start frame delimiter is detected, the receiver continues decoding with the same synchronization. If the stream does not match a valid pattern or a valid start frame delimiter, the receiver re-synchronizes on the next valid edge.The minimum time threshold to estimate the bit value is three quarters of a bit time.

If a valid preamble (if used) followed with a valid start frame delimiter is detected, the incoming stream is decoded into NRZ data and passed to USART for processing. Figure 34-15 illustrates Manchester pattern mismatch. When incoming data stream is passed to the USART, the receiver is also able to detect Manchester code violation. A code violation is a lack of transition in the middle of a bit cell. In this case, MANE flag in US_CSR register is raised. It is cleared by writing the Control Register (US_CR) with the RSTSTA bit at 1. See Figure 34-16 for an example of Manchester error detection during data phase.

Figure 34-15. Preamble Pattern Mismatch

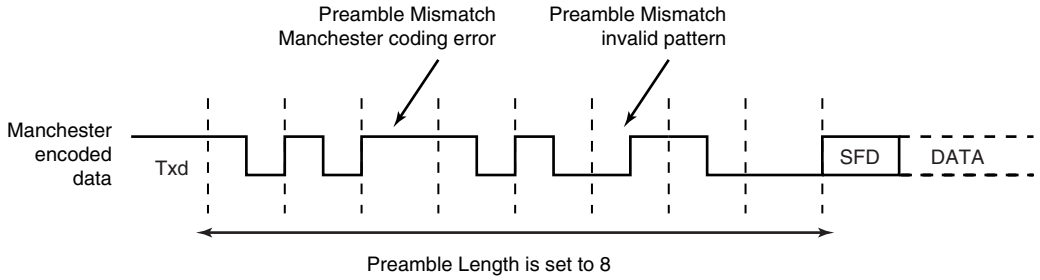
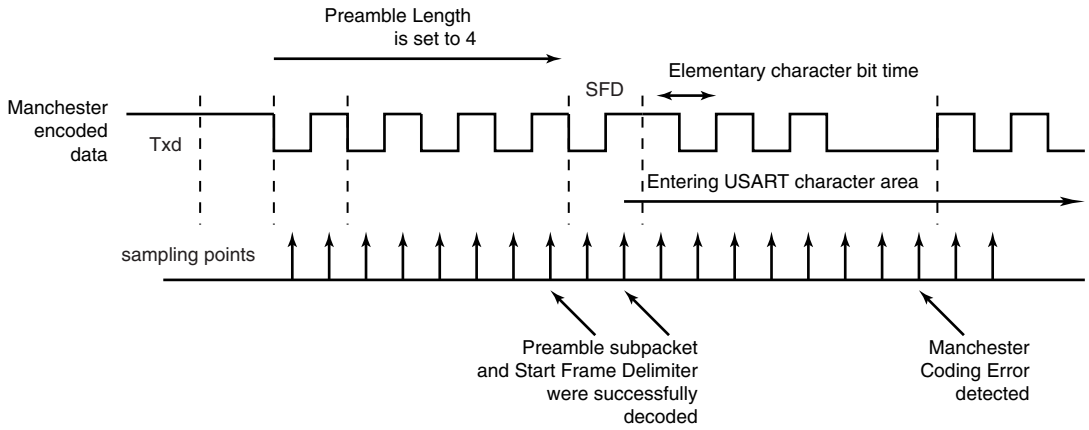


Figure 34-16. Manchester Error Flag



When the start frame delimiter is a sync pattern (ONEBIT field at 0), both command and data delimiter are supported. If a valid sync is detected, the received character is written as RXCHR field in the US_RHR register and the RXSYNH is updated. RXCHR is set to 1 when the received character is a command, and it is set to 0 if the received character is a data. This mechanism alleviates and simplifies the direct memory access as the character contains its own sync field in the same register.

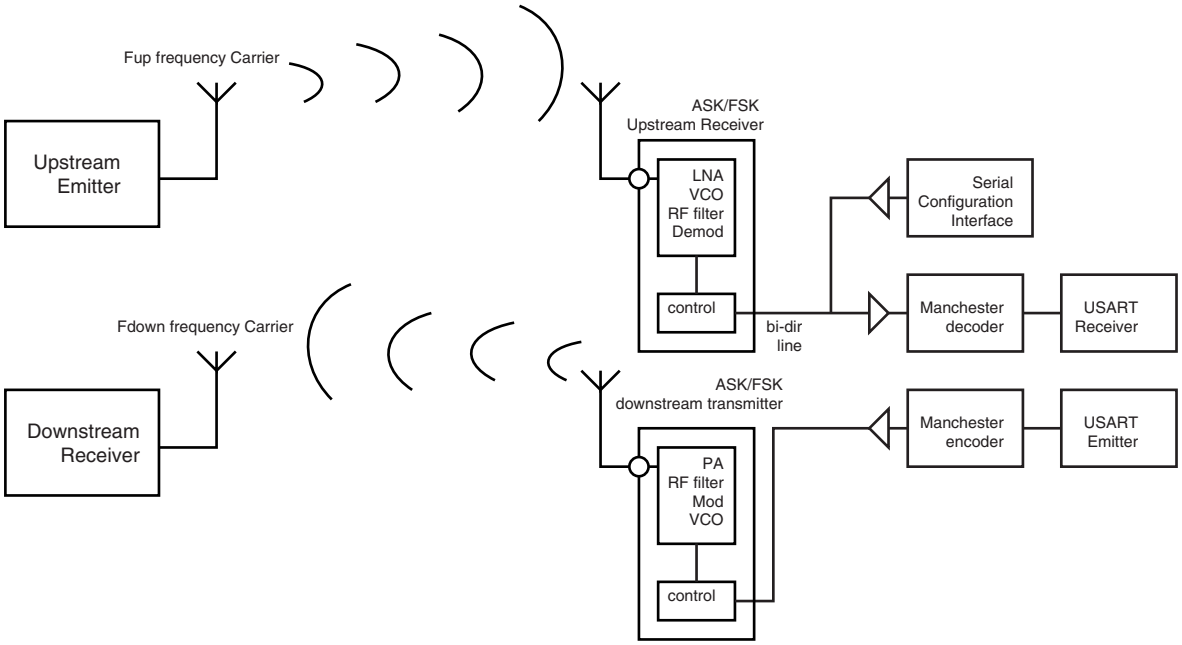
As the decoder is setup to be used in unipolar mode, the first bit of the frame has to be a zero-to-one transition.

34.6.3.6 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 34-17.

Figure 34-17. Manchester Encoded Characters RF Transmission



The USART module is configured as a Manchester encoder/decoder. Looking at the downstream communication channel, Manchester encoded characters are serially sent to the RF emitter. This may also include a user defined preamble and a start frame delimiter. Mostly, preamble is used in the RF receiver to distinguish between a valid data from a transmitter and signals due to noise. The Manchester stream is then modulated. See Figure 34-18 for an example of ASK modulation scheme. When a logic one is sent to the ASK modulator, the power amplifier, referred to as PA, is enabled and transmits an RF signal at downstream frequency. When a logic zero is transmitted, the RF signal is turned off. If the FSK modulator is activated, two different frequencies are used to transmit data. When a logic 1 is sent, the modulator outputs an RF signal at frequency F0 and switches to F1 if the data sent is a 0. See Figure 34-19.

From the receiver side, another carrier frequency is used. The RF receiver performs a bit check operation examining demodulated data stream. If a valid pattern is detected, the receiver

switches to receiving mode. The demodulated stream is sent to the Manchester decoder. Because of bit checking inside RF IC, the data transferred to the microcontroller is reduced by a user-defined number of bits. The Manchester preamble length is to be defined in accordance with the RF IC configuration.

Figure 34-18. ASK Modulator Output

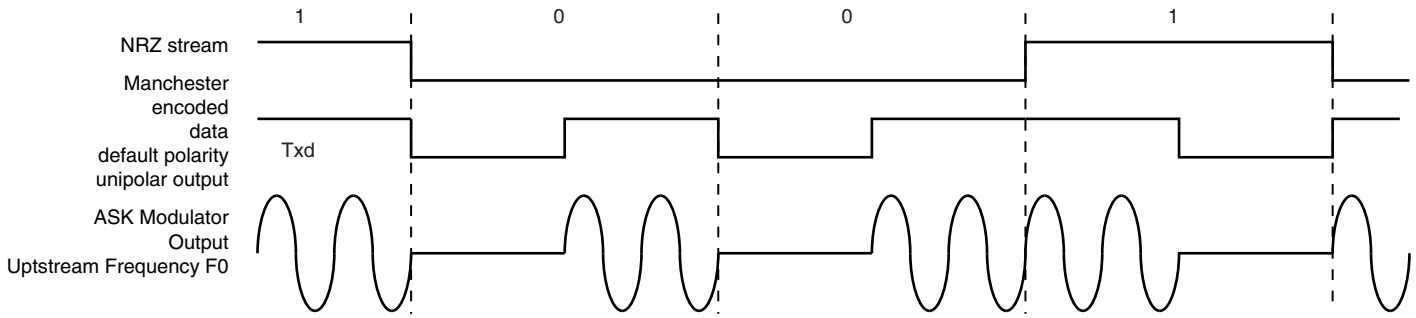
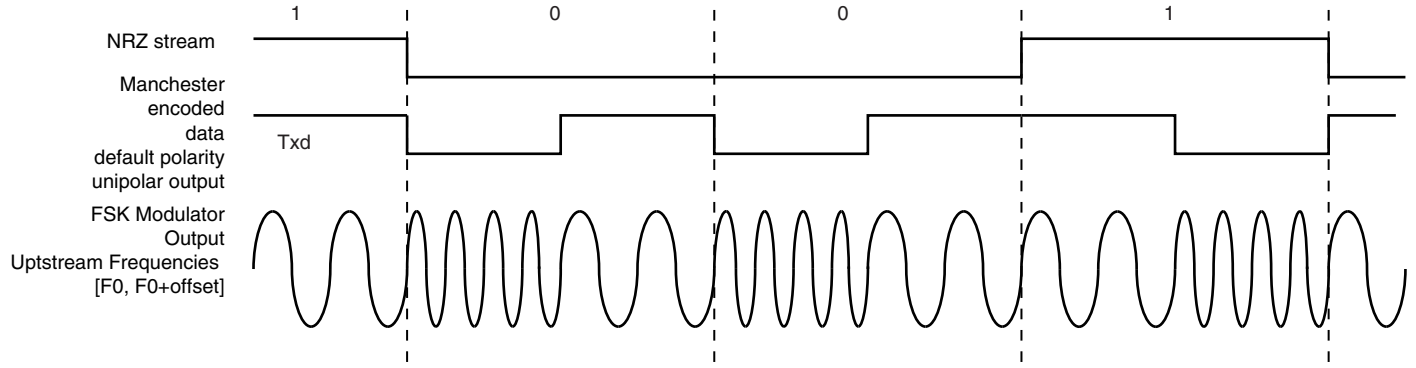


Figure 34-19. FSK Modulator Output



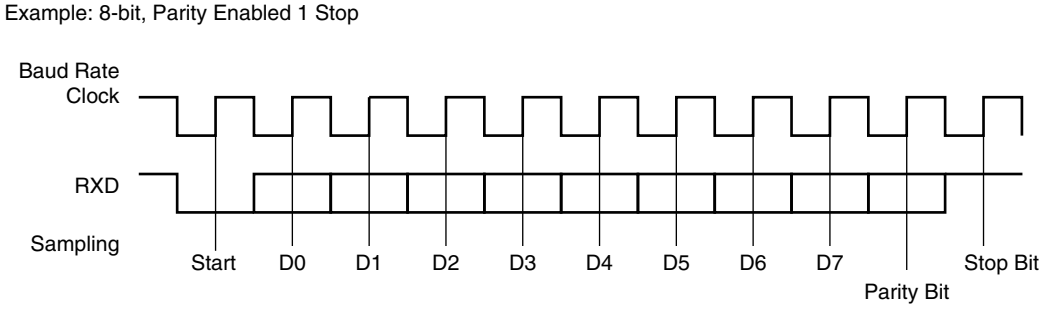
34.6.3.7 Synchronous Receiver

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 34-20 illustrates a character reception in synchronous mode.

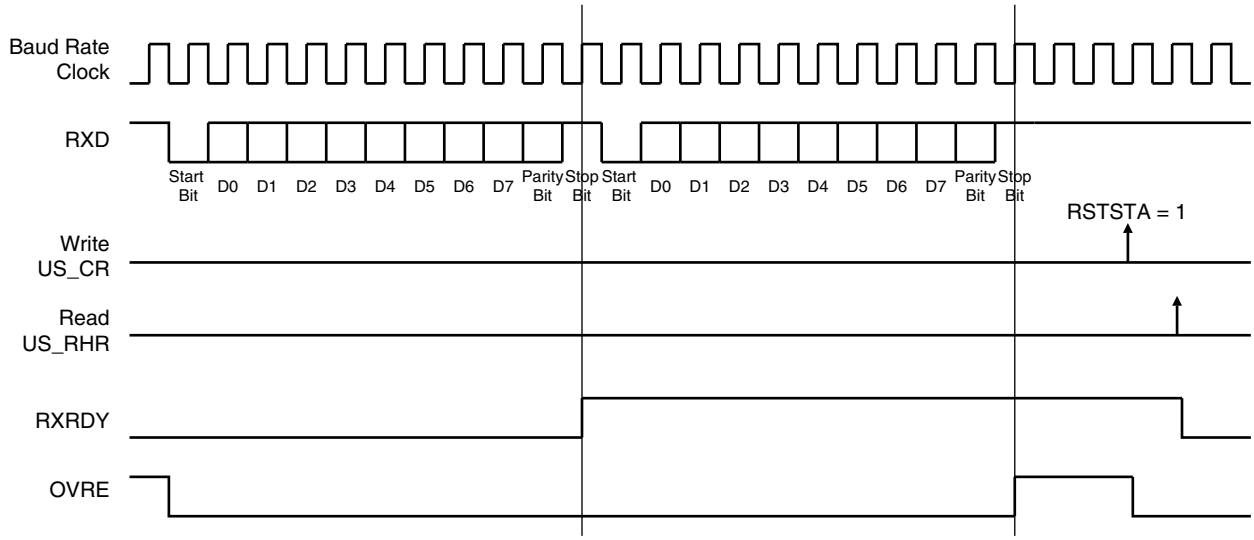
Figure 34-20. Synchronous Mode Character Reception



34.6.3.8 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 34-21. Receiver Status



34.6.3.9 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 538. Even and odd parity bit generation and error detection are supported.

If even parity is selected, the parity generator of the transmitter drives the parity bit at 0 if a number of 1s in the character data bit is even, and at 1 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If odd parity is selected, the parity generator of the transmitter drives the parity bit at 1 if a number of 1s in the character data bit is even, and at 0 if the number of 1s is odd. Accordingly, the receiver parity checker counts the number of received 1s and reports a parity error if the sampled parity bit does not correspond. If the mark parity is used, the parity generator of the transmitter drives the parity bit at 1 for all characters. The receiver parity checker reports an error if the parity bit is sampled at 0. If the space parity is used, the parity generator of the transmitter drives the parity bit at 0 for all characters. The receiver parity checker reports an error if the parity bit is sampled at 1. If parity is disabled, the transmitter does not generate any parity bit and the receiver does not report any parity error.

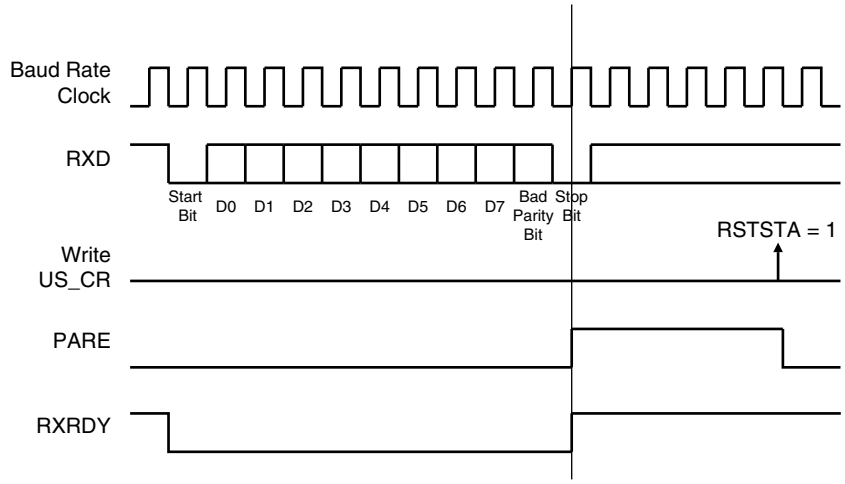
Table 34-6 shows an example of the parity bit for the character 0x41 (character ASCII “A”) depending on the configuration of the USART. Because there are two bits at 1, 1 bit is added when a parity is odd, or 0 is added when a parity is even.

Table 34-6. Parity Bit Examples

Character	Hexa	Binary	Parity Bit	Parity Mode
A	0x41	0100 0001	1	Odd
A	0x41	0100 0001	0	Even
A	0x41	0100 0001	1	Mark
A	0x41	0100 0001	0	Space
A	0x41	0100 0001	None	None

When the receiver detects a parity error, it sets the PARE (Parity Error) bit in the Channel Status Register (US_CSR). The PARE bit can be cleared by writing the Control Register (US_CR) with the RSTSTA bit at 1. Figure 34-22 illustrates the parity bit status setting and clearing.

Figure 34-22. Parity Error



34.6.3.10 *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.

34.6.3.11 *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 34-23, the behavior of TXRDY and TXEMPTY status bits is modified by the programming of a timeguard. TXRDY rises only when the start bit of the next character is sent, and thus remains at 0 during the timeguard transmission if a character has been written in US_THR. TXEMPTY remains low until the timeguard transmission is completed as the timeguard is part of the current character being transmitted.

Figure 34-23. Timeguard Operations

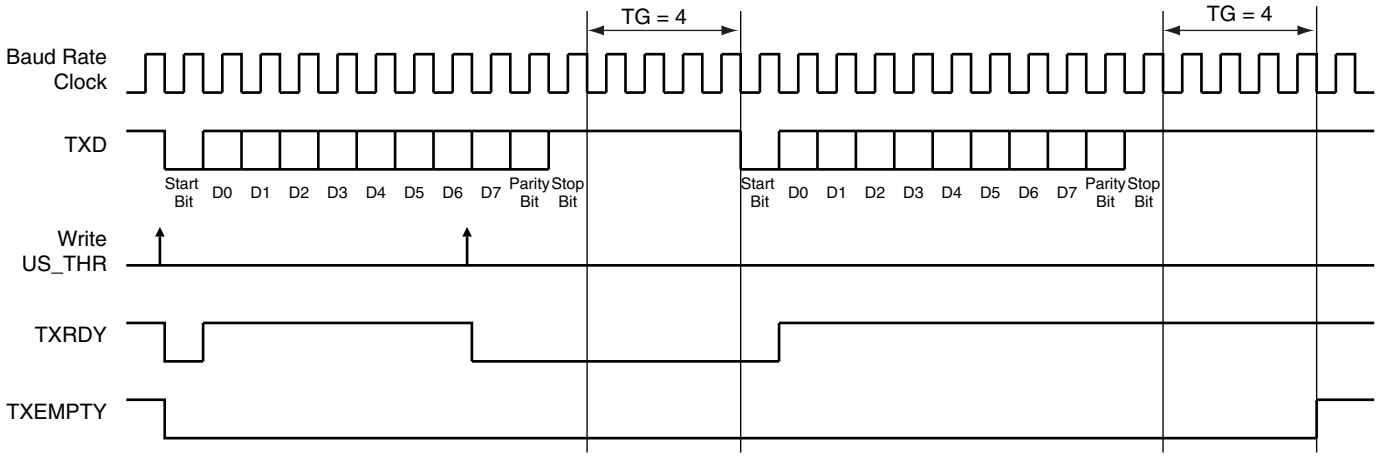


Table 34-7 indicates the maximum length of a timeguard period that the transmitter can handle in relation to the function of the Baud Rate.

Table 34-7. Maximum Timeguard Length Depending on Baud Rate

Baud Rate	Bit time	Timeguard
Bit/sec	µs	ms
1 200	833	212.50
9 600	104	26.56
14400	69.4	17.71
19200	52.1	13.28
28800	34.7	8.85
33400	29.9	7.63
56000	17.9	4.55
57600	17.4	4.43
115200	8.7	2.21

34.6.3.12 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 34-24 shows the block diagram of the Receiver Time-out feature.

Figure 34-24. Receiver Time-out Block Diagram

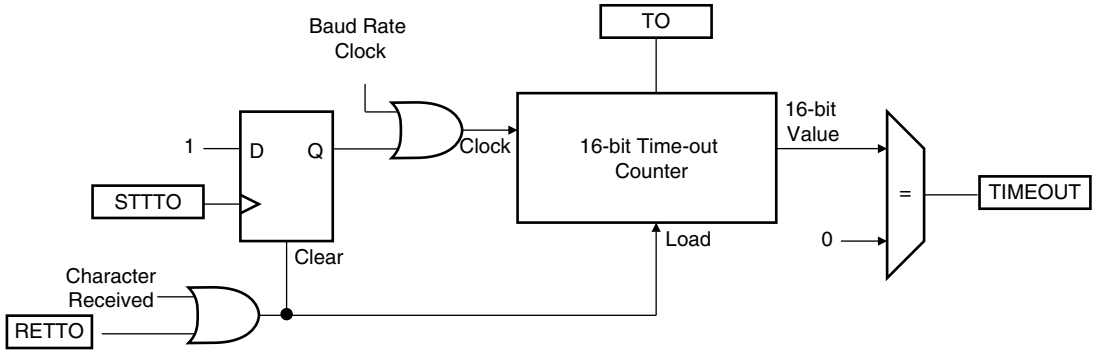


Table 34-8 gives the maximum time-out period for some standard baud rates.

Table 34-8. Maximum Time-out Period

Baud Rate	Bit Time	Time-out
bit/sec	µs	ms
600	1 667	109 225
1 200	833	54 613
2 400	417	27 306
4 800	208	13 653
9 600	104	6 827
14400	69	4 551
19200	52	3 413
28800	35	2 276
33400	30	1 962

Table 34-8. Maximum Time-out Period (Continued)

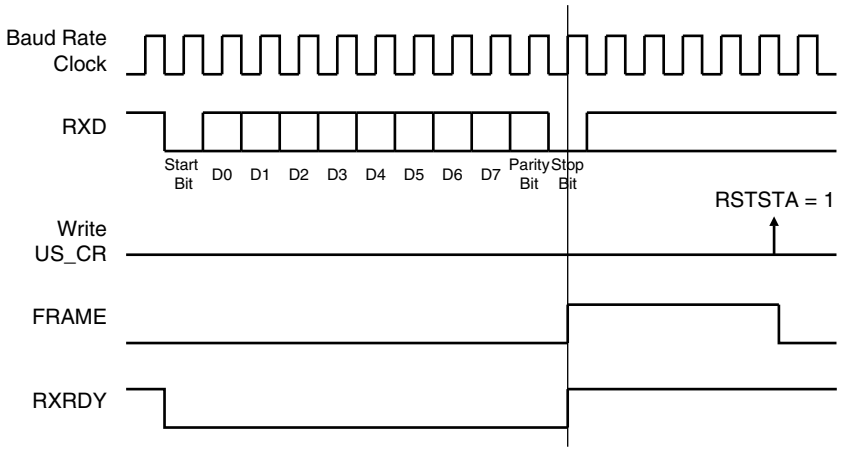
Baud Rate	Bit Time	Time-out
56000	18	1 170
57600	17	1 138
200000	5	328

34.6.3.13 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 34-25. Framing Error Status



34.6.3.14 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.

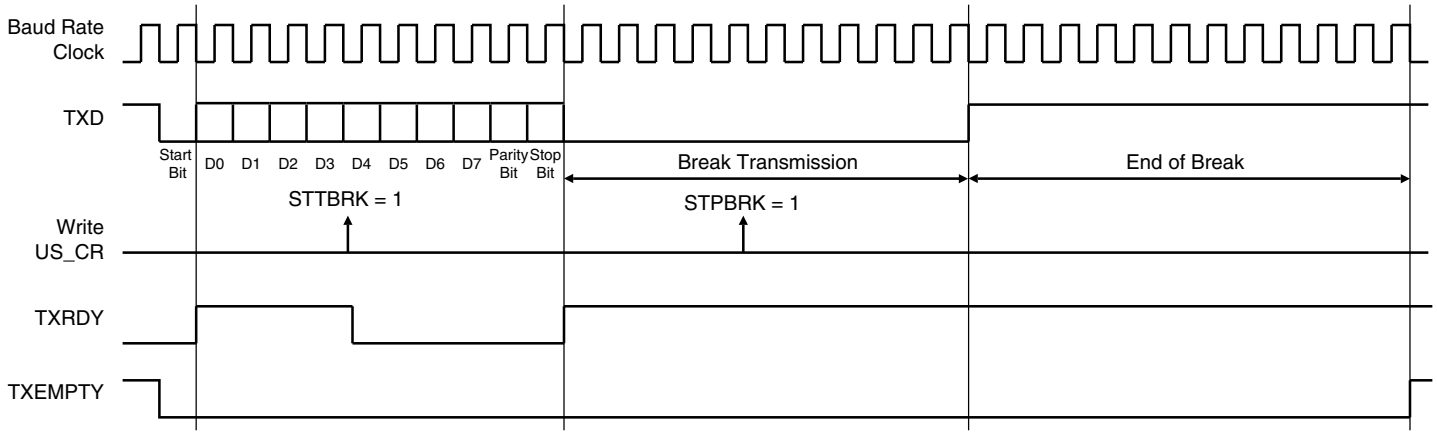
Writing US_CR with the both STTBRK and STPBRK bits at 1 can lead to an unpredictable result. All STPBRK commands requested without a previous STTBRK command are ignored. A byte written into the Transmit Holding Register while a break is pending, but not started, is ignored.

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 34-26 illustrates the effect of both the Start Break (STTBRK) and Stop Break (STPBRK) commands on the TXD line.

Figure 34-26. Break Transmission



34.6.3.15 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.

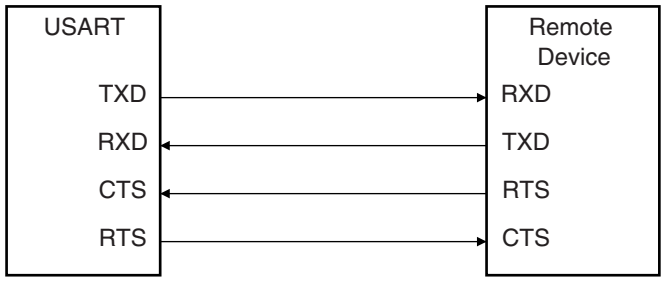
When the low stop bit is detected, the receiver asserts the RXBRK bit in US_CSR. This bit may be cleared by writing the Control Register (US_CR) with the bit RSTSTA at 1.

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.

34.6.3.16 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 34-27.

Figure 34-27. Connection with a Remote Device for Hardware Handshaking



Setting the USART to operate with hardware handshaking is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x2.

The USART behavior when hardware handshaking is enabled is the same as the behavior in standard synchronous or asynchronous mode, except that the receiver drives the RTS pin as described below and the level on the CTS pin modifies the behavior of the transmitter as described below. Using this mode requires using the PDC channel for reception. The transmitter can handle hardware handshaking in any case.

Figure 34-28 shows how the receiver operates if hardware handshaking is enabled. The RTS pin is driven high if the receiver is disabled and if the status RXBUFF (Receive Buffer Full) coming from the PDC channel is high. Normally, the remote device does not start transmitting while its CTS pin (driven by RTS) is high. As soon as the Receiver is enabled, the RTS falls, indicating to the remote device that it can start transmitting. Defining a new buffer to the PDC clears the status bit RXBUFF and, as a result, asserts the pin RTS low.

Figure 34-28. Receiver Behavior when Operating with Hardware Handshaking

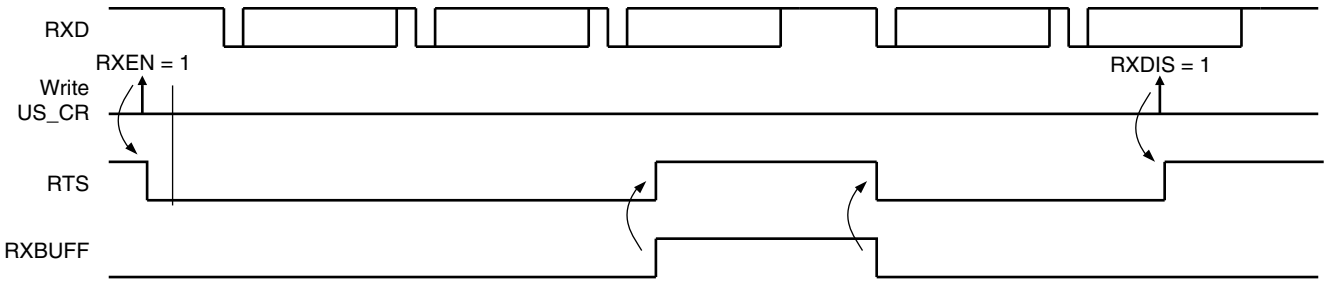
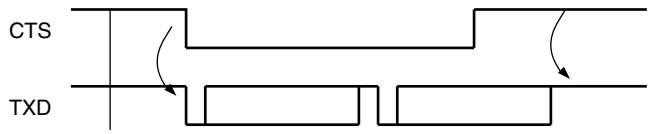


Figure 34-29 shows how the transmitter operates if hardware handshaking is enabled. The CTS pin disables the transmitter. If a character is being processing, the transmitter is disabled only after the completion of the current character and transmission of the next character happens as soon as the pin CTS falls.

Figure 34-29. Transmitter Behavior when Operating with Hardware Handshaking



34.6.4 ISO7816 Mode

The USART features an ISO7816-compatible operating mode. This mode permits interfacing with smart cards and Security Access Modules (SAM) communicating through an ISO7816 link. Both T = 0 and T = 1 protocols defined by the ISO7816 specification are supported.

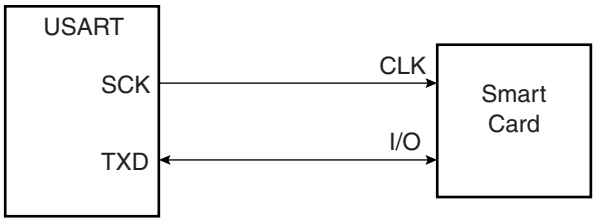
Setting the USART in ISO7816 mode is performed by writing the USART_MODE field in the Mode Register (US_MR) to the value 0x4 for protocol T = 0 and to the value 0x5 for protocol T = 1.

34.6.4.1 ISO7816 Mode Overview

The ISO7816 is a half duplex communication on only one bidirectional line. The baud rate is determined by a division of the clock provided to the remote device (see "Baud Rate Generator" on page 523).

The USART connects to a smart card as shown in Figure 34-30. The TXD line becomes bidirectional and the Baud Rate Generator feeds the ISO7816 clock on the SCK pin. As the TXD pin becomes bidirectional, its output remains driven by the output of the transmitter but only when the transmitter is active while its input is directed to the input of the receiver. The USART is considered as the master of the communication as it generates the clock.

Figure 34-30. Connection of a Smart Card to the USART



When operating in ISO7816, either in T = 0 or T = 1 modes, the character format is fixed. The configuration is 8 data bits, even parity and 1 or 2 stop bits, regardless of the values programmed in the CHRL, MODE9, PAR and CHMODE fields. MSBF can be used to transmit LSB or MSB first. Parity Bit (PAR) can be used to transmit in normal or inverse mode. Refer to "USART Mode Register" on page 555 and "PAR: Parity Type" on page 556.

The USART cannot operate concurrently in both receiver and transmitter modes as the communication is unidirectional at a time. It has to be configured according to the required mode by enabling or disabling either the receiver or the transmitter as desired. Enabling both the receiver and the transmitter at the same time in ISO7816 mode may lead to unpredictable results.

The ISO7816 specification defines an inverse transmission format. Data bits of the character must be transmitted on the I/O line at their negative value. The USART does not support this format and the user has to perform an exclusive OR on the data before writing it in the Transmit Holding Register (US_THR) or after reading it in the Receive Holding Register (US_RHR).

34.6.4.2 Protocol T = 0

In T = 0 protocol, a character is made up of one start bit, eight data bits, one parity bit and one guard time, which lasts two bit times. The transmitter shifts out the bits and does not drive the I/O line during the guard time.

If no parity error is detected, the I/O line remains at 1 during the guard time and the transmitter can continue with the transmission of the next character, as shown in Figure 34-31.

If a parity error is detected by the receiver, it drives the I/O line at 0 during the guard time, as shown in Figure 34-32. This error bit is also named NACK, for Non Acknowledge. In this case, the character lasts 1 bit time more, as the guard time length is the same and is added to the error bit time which lasts 1 bit time.

When the USART is the receiver and it detects an error, it does not load the erroneous character in the Receive Holding Register (US_RHR). It appropriately sets the PARE bit in the Status Register (US_SR) so that the software can handle the error.

Figure 34-31. T = 0 Protocol without Parity Error

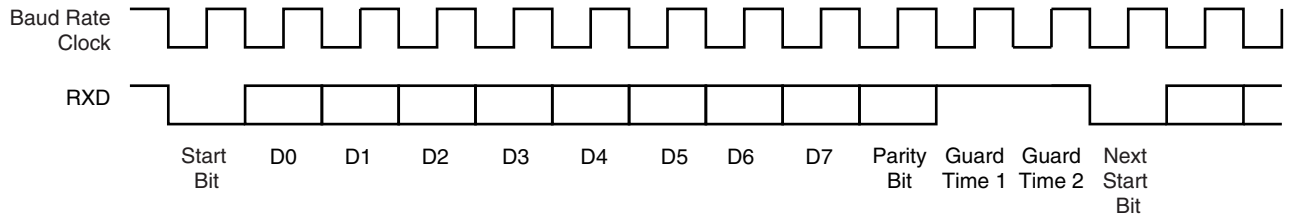
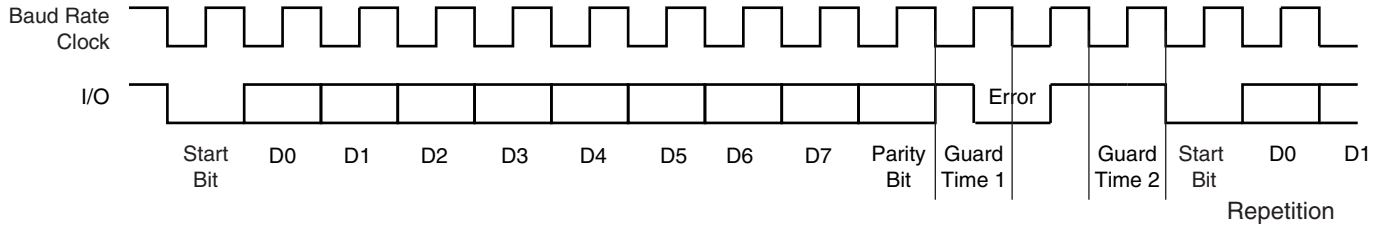


Figure 34-32. T = 0 Protocol with Parity Error



34.6.4.3 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.

34.6.4.4 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.

34.6.4.5 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.

34.6.4.6 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.

34.6.4.7 Protocol T = 1

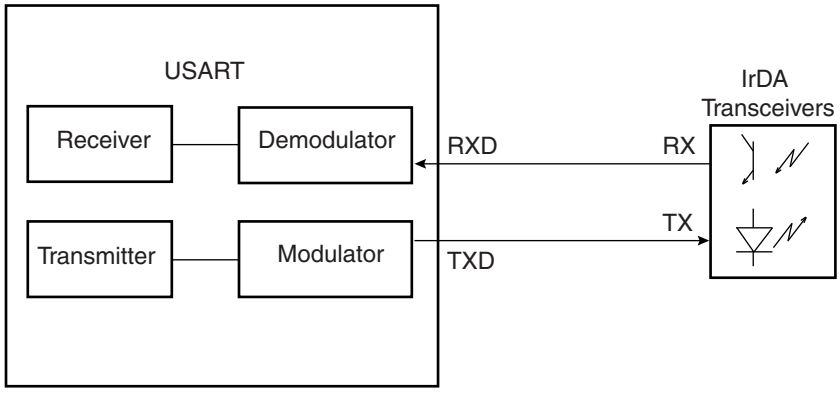
When operating in ISO7816 protocol T = 1, the transmission is similar to an asynchronous format with only one stop bit. The parity is generated when transmitting and checked when receiving. Parity error detection sets the PARE bit in the Channel Status Register (US_CSR).

34.6.5 IrDA Mode

The USART features an IrDA mode supplying half-duplex point-to-point wireless communication. It embeds the modulator and demodulator which allows a glueless connection to the infrared transceivers, as shown in Figure 34-33. The modulator and demodulator are compliant with the IrDA specification version 1.1 and support data transfer speeds ranging from 2.4 Kb/s to 115.2 Kb/s.

The USART IrDA mode is enabled by setting the USART_MODE field in the Mode Register (US_MR) to the value 0x8. The IrDA Filter Register (US_IF) allows configuring the demodulator filter. The USART transmitter and receiver operate in a normal asynchronous mode and all parameters are accessible. Note that the modulator and the demodulator are activated.

Figure 34-33. Connection to IrDA Transceivers



The receiver and the transmitter must be enabled or disabled according to the direction of the transmission to be managed.

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

34.6.5.1 IrDA Modulation

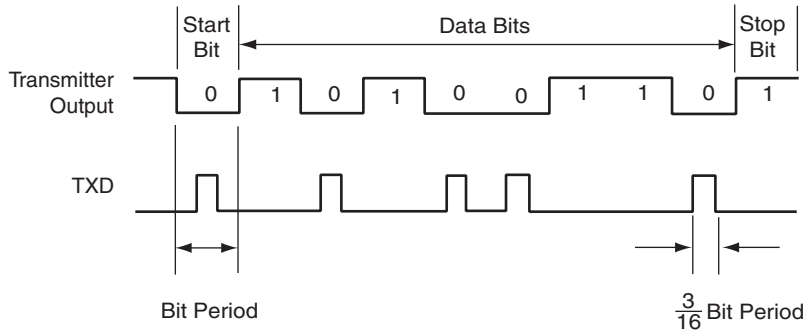
For baud rates up to and including 115.2 Kbits/sec, the RZI modulation scheme is used. “0” is represented by a light pulse of 3/16th of a bit time. Some examples of signal pulse duration are shown in Table 34-9.

Table 34-9. IrDA Pulse Duration

Baud Rate	Pulse Duration (3/16)
2.4 Kb/s	78.13 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 34-34 shows an example of character transmission.

Figure 34-34. IrDA Modulation



34.6.5.2 IrDA Baud Rate

Table 34-10 gives some examples of CD values, baud rate error and pulse duration. Note that the requirement on the maximum acceptable error of ±1.87% must be met.

Table 34-10. IrDA Baud Rate Error

Peripheral Clock	Baud Rate	CD	Baud Rate Error	Pulse Time
3 686 400	115 200	2	0.00%	1.63
20 000 000	115 200	11	1.38%	1.63
32 768 000	115 200	18	1.25%	1.63
40 000 000	115 200	22	1.38%	1.63
3 686 400	57 600	4	0.00%	3.26
20 000 000	57 600	22	1.38%	3.26
32 768 000	57 600	36	1.25%	3.26

Table 34-10. IrDA Baud Rate Error (Continued)

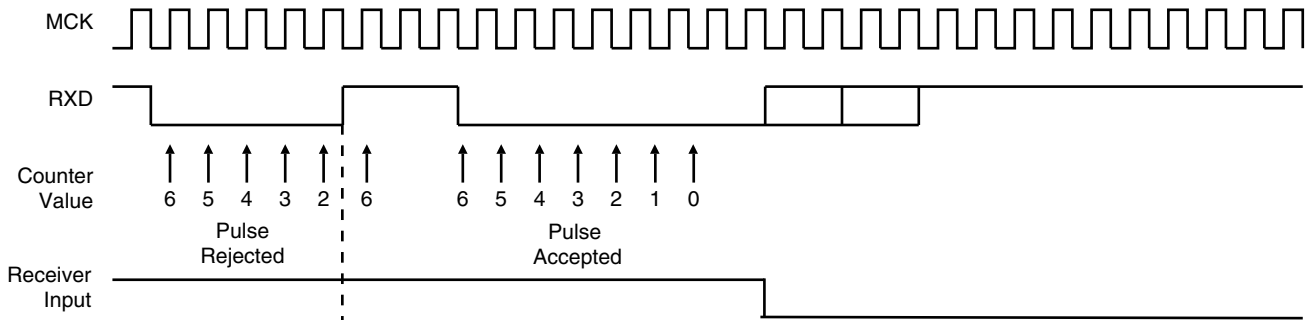
Peripheral Clock	Baud Rate	CD	Baud Rate Error	Pulse Time
40 000 000	57 600	43	0.93%	3.26
3 686 400	38 400	6	0.00%	4.88
20 000 000	38 400	33	1.38%	4.88
32 768 000	38 400	53	0.63%	4.88
40 000 000	38 400	65	0.16%	4.88
3 686 400	19 200	12	0.00%	9.77
20 000 000	19 200	65	0.16%	9.77
32 768 000	19 200	107	0.31%	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

34.6.5.3 IrDA Demodulator

The demodulator is based on the IrDA Receive filter comprised of an 8-bit down counter which is loaded with the value programmed in US_IF. When a falling edge is detected on the RXD pin, the Filter Counter starts counting down at the Master Clock (MCK) speed. If a rising edge is detected on the RXD pin, the counter stops and is reloaded with US_IF. If no rising edge is detected when the counter reaches 0, the input of the receiver is driven low during one bit time.

Figure 34-35 illustrates the operations of the IrDA demodulator.

Figure 34-35. IrDA Demodulator Operations

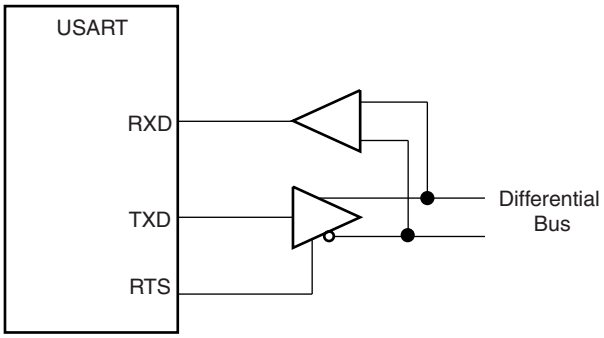


As the IrDA mode uses the same logic as the ISO7816, note that the FI_DI_RATIO field in US_FIDI must be set to a value higher than 0 in order to assure IrDA communications operate correctly.

34.6.6 RS485 Mode

The USART features the RS485 mode to enable line driver control. While operating in RS485 mode, the USART behaves as though in asynchronous or synchronous mode and configuration of all the parameters is possible. The difference is that the RTS pin is driven high when the transmitter is operating. The behavior of the RTS pin is controlled by the TXEMPTY bit. A typical connection of the USART to a RS485 bus is shown in Figure 34-36.

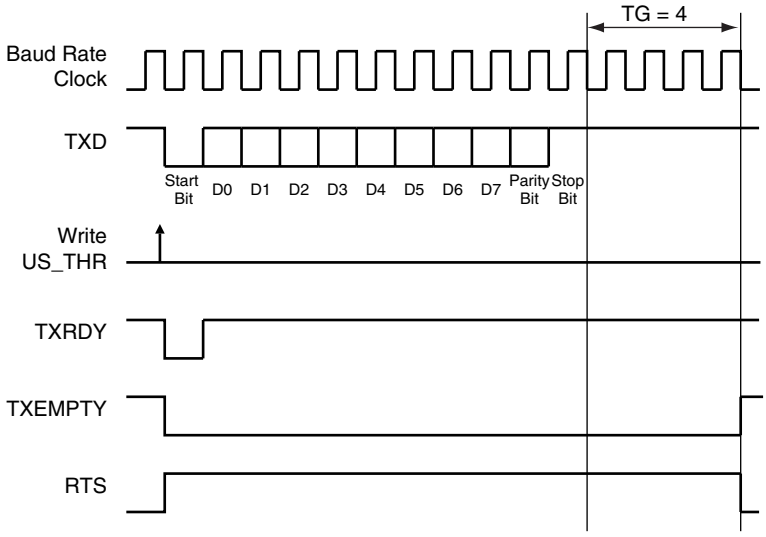
Figure 34-36. Typical Connection to a RS485 Bus



The USART is set in RS485 mode by programming the USART_MODE field in the Mode Register (US_MR) to the value 0x1.

The RTS pin is at a level inverse to the TXEMPTY bit. Significantly, the RTS pin remains high when a timeguard is programmed so that the line can remain driven after the last character completion. Figure 34-37 gives an example of the RTS waveform during a character transmission when the timeguard is enabled.

Figure 34-37. Example of RTS Drive with Timeguard



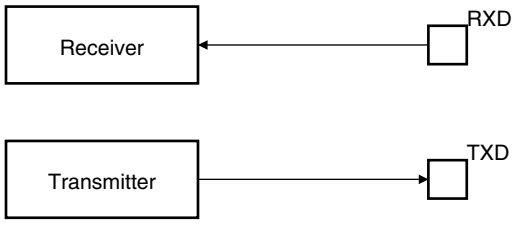
34.6.7 Test Modes

The USART can be programmed to operate in three different test modes. The internal loopback capability allows on-board diagnostics. In the loopback mode the USART interface pins are disconnected or not and reconfigured for loopback internally or externally.

34.6.7.1 Normal Mode

Normal mode connects the RXD pin on the receiver input and the transmitter output on the TXD pin.

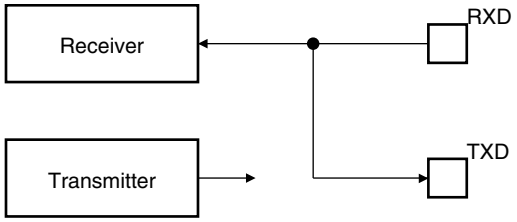
Figure 34-38. Normal Mode Configuration



34.6.7.2 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 34-39. Programming the transmitter has no effect on the TXD pin. The RXD pin is still connected to the receiver input, thus the receiver remains active.

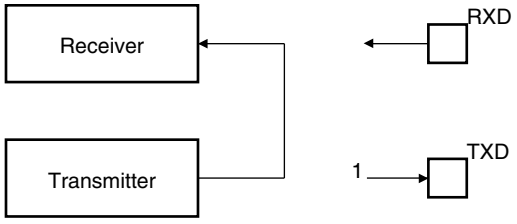
Figure 34-39. Automatic Echo Mode Configuration



34.6.7.3 Local Loopback Mode

Local loopback mode connects the output of the transmitter directly to the input of the receiver, as shown in Figure 34-40. The TXD and RXD pins are not used. The RXD pin has no effect on the receiver and the TXD pin is continuously driven high, as in idle state.

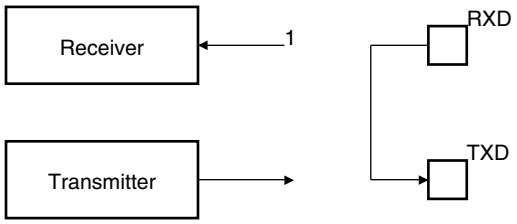
Figure 34-40. Local Loopback Mode Configuration



34.6.7.4 *Remote Loopback Mode*

Remote loopback mode directly connects the RXD pin to the TXD pin, as shown in Figure 34-41. The transmitter and the receiver are disabled and have no effect. This mode allows bit-by-bit retransmission.

Figure 34-41. Remote Loopback Mode Configuration



34.7 Universal Synchronous Asynchronous Receiver Transmitter (USART) User Interface

Table 34-12. 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	–	–	–

34.7.1 USART Control Register

Name: US_CR

Addresses: 0xFFF8C000 (0), 0xFFF90000 (1), 0xFFF94000 (2)

Access Type: 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.

34.7.2 USART Mode Register

Name: US_MR

Addresses: 0xFFF8C004 (0), 0xFFF90004 (1), 0xFFF94004 (2)

Access Type: Read-write

31	30	29	28	27	26	25	24
ONEBIT	MODSYNC–	MAN	FILTER	–	MAX_ITERATION		
23	22	21	20	19	18	17	16
–	VAR_SYNC	DSNACK	INACK	OVER	CLKO	MODE9	MSBF
15	14	13	12	11	10	9	8
CHMODE		NBSTOP		PAR			SYNC
7	6	5	4	3	2	1	0
CHRL		USCLKS		USART_MODE			

- **USART_MODE**

USART_MODE				Mode of the USART
0	0	0	0	Normal
0	0	0	1	RS485
0	0	1	0	Hardware Handshaking
0	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.

34.7.3 USART Interrupt Enable Register

Name: US_IER

Addresses: 0xFFF8C008 (0), 0xFFF90008 (1), 0xFFF94008 (2)

Access Type: Write-only

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

34.7.4 USART Interrupt Disable Register

Name: US_IDR

Addresses: 0xFFF8C00C (0), 0xFFF9000C (1), 0xFFF9400C (2)

Access Type: Write-only

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

34.7.5 USART Interrupt Mask Register

Name: US_IMR

Addresses: 0xFFF8C010 (0), 0xFFF90010 (1), 0xFFF94010 (2)

Access Type: Read-only

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

34.7.6 USART Channel Status Register

Name: US_CSR

Addresses: 0xFFF8C014 (0), 0xFFF90014 (1), 0xFFF94014 (2)

Access Type: 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.

34.7.7 USART Receive Holding Register

Name: US_RHR

Addresses: 0xFFF8C018 (0), 0xFFF90018 (1), 0xFFF94018 (2)

Access Type: Read-only

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

- **RXCHR: Received Character**

Last character received if RXRDY is set.

- **RXSYNH: Received Sync**

0: Last Character received is a Data.

1: Last Character received is a Command.

34.7.8 USART Transmit Holding Register

Name: US_THR

Addresses: 0xFFF8C01C (0), 0xFFF9001C (1), 0xFFF9401C (2)

Access Type: Write-only

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

- **TXCHR: Character to be Transmitted**

Next character to be transmitted after the current character if TXRDY is not set.

- **TXSYNH: Sync Field to be transmitted**

0: The next character sent is encoded as a data. Start Frame Delimiter is DATA SYNC.

1: The next character sent is encoded as a command. Start Frame Delimiter is COMMAND SYNC.

34.7.9 USART Baud Rate Generator Register

Name: US_BRGR

Addresses: 0xFFF8C020 (0), 0xFFF90020 (1), 0xFFF94020 (2)

Access Type: Read-write

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

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

15	14	13	12	11	10	9	8
CD							

7	6	5	4	3	2	1	0
CD							

- **CD: Clock Divider**

CD	USART_MODE ≠ ISO7816			USART_MODE = ISO7816
	SYNC = 0		SYNC = 1	
	OVER = 0	OVER = 1		
0	Baud Rate Clock Disabled			
1 to 65535	Baud Rate = Selected Clock/16/CD	Baud Rate = Selected Clock/8/CD	Baud Rate = Selected Clock /CD	Baud Rate = Selected Clock/CD/FI_DI_RATIO

- **FP: Fractional Part**

0: Fractional divider is disabled.

1 - 7: Baudrate resolution, defined by FP x 1/8.

34.7.10 USART Receiver Time-out Register

Name: US_RTOR

Addresses: 0xFFF8C024 (0), 0xFFF90024 (1), 0xFFF94024 (2)

Access Type: Read-write

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

- **TO: Time-out Value**

0: The Receiver Time-out is disabled.

1 - 65535: The Receiver Time-out is enabled and the Time-out delay is TO x Bit Period.

34.7.11 USART Transmitter Timeguard Register

Name: US_TTGR

Addresses: 0xFFF8C028 (0), 0xFFF90028 (1), 0xFFF94028 (2)

Access Type: Read-write

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

- **TG: Timeguard Value**

0: The Transmitter Timeguard is disabled.

1 - 255: The Transmitter timeguard is enabled and the timeguard delay is TG x Bit Period.

34.7.12 USART FI DI RATIO Register

Name: US_FIDI

Addresses: 0xFFF8C040 (0), 0xFFF90040 (1), 0xFFF94040 (2)

Access Type: Read-write

Reset Value: 0x174

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

- **FI_DI_RATIO: FI Over DI Ratio Value**

0: If ISO7816 mode is selected, the Baud Rate Generator generates no signal.

1 - 2047: If ISO7816 mode is selected, the Baud Rate is the clock provided on SCK divided by FI_DI_RATIO.

34.7.13 USART Number of Errors Register

Name: US_NER

Addresses: 0xFFF8C044 (0), 0xFFF90044 (1), 0xFFF94044 (2)

Access Type: Read-only

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

- **NB_ERRORS: Number of Errors**

Total number of errors that occurred during an ISO7816 transfer. This register automatically clears when read.

34.7.14 USART IrDA FILTER Register

Name: US_IF

Addresses: 0xFFF8C04C (0), 0xFFF9004C (1), 0xFFF9404C (2)

Access Type: Read-write

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

- **IRDA_FILTER: IrDA Filter**

Sets the filter of the IrDA demodulator.

34.7.15 USART Manchester Configuration Register

Name: US_MAN

Access Type: 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			

- **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.

35. Synchronous Serial Controller (SSC)

35.1 Overview

The Atmel Synchronous Serial Controller (SSC) provides a synchronous communication link with external devices. It supports many serial synchronous communication protocols generally used in audio and telecom applications such as I2S, Short Frame Sync, Long Frame Sync, etc.

The SSC contains an independent receiver and transmitter and a common clock divider. The receiver and the transmitter each interface with three signals: the TD/RD signal for data, the TK/RK signal for the clock and the TF/RF signal for the Frame Sync. The transfers can be programmed to start automatically or on different events detected on the Frame Sync signal.

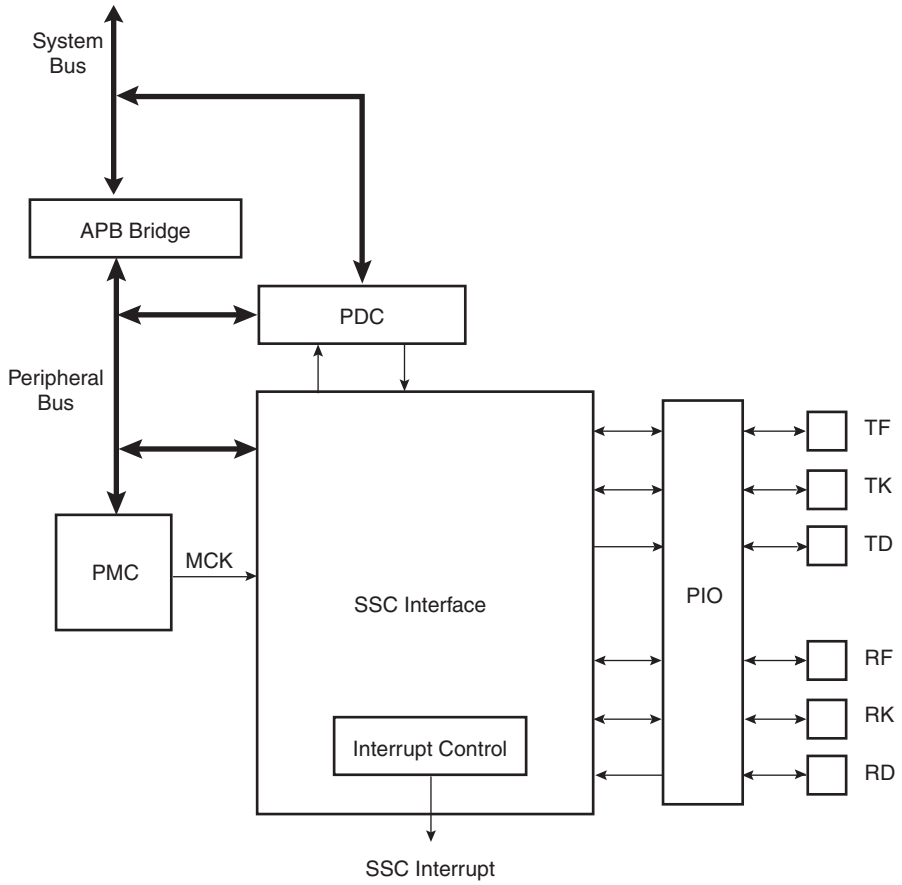
The SSC's high-level of programmability and its two dedicated PDC channels of up to 32 bits permit a continuous high bit rate data transfer without processor intervention.

Featuring connection to two PDC channels, the SSC permits interfacing with low processor overhead to the following:

- CODEC's in master or slave mode
- DAC through dedicated serial interface, particularly I2S
- Magnetic card reader

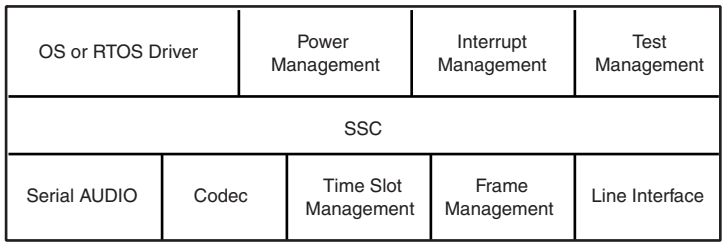
35.2 Block Diagram

Figure 35-1. Block Diagram



35.3 Application Block Diagram

Figure 35-2. Application Block Diagram



35.4 Pin Name List

Table 35-1. I/O Lines Description

Pin Name	Pin Description	Type
RF	Receiver Frame Synchro	Input/Output
RK	Receiver Clock	Input/Output
RD	Receiver Data	Input
TF	Transmitter Frame Synchro	Input/Output
TK	Transmitter Clock	Input/Output
TD	Transmitter Data	Output

35.5 Product Dependencies

35.5.1 I/O Lines

The pins used for interfacing the compliant external devices may be multiplexed with PIO lines.

Before using the SSC receiver, the PIO controller must be configured to dedicate the SSC receiver I/O lines to the SSC peripheral mode.

Before using the SSC transmitter, the PIO controller must be configured to dedicate the SSC transmitter I/O lines to the SSC peripheral mode.

35.5.2 Power Management

The SSC is not continuously clocked. The SSC interface may be clocked through the Power Management Controller (PMC), therefore the programmer must first configure the PMC to enable the SSC clock.

35.5.3 Interrupt

The SSC interface has an interrupt line connected to the Advanced Interrupt Controller (AIC). Handling interrupts requires programming the AIC before configuring the SSC.

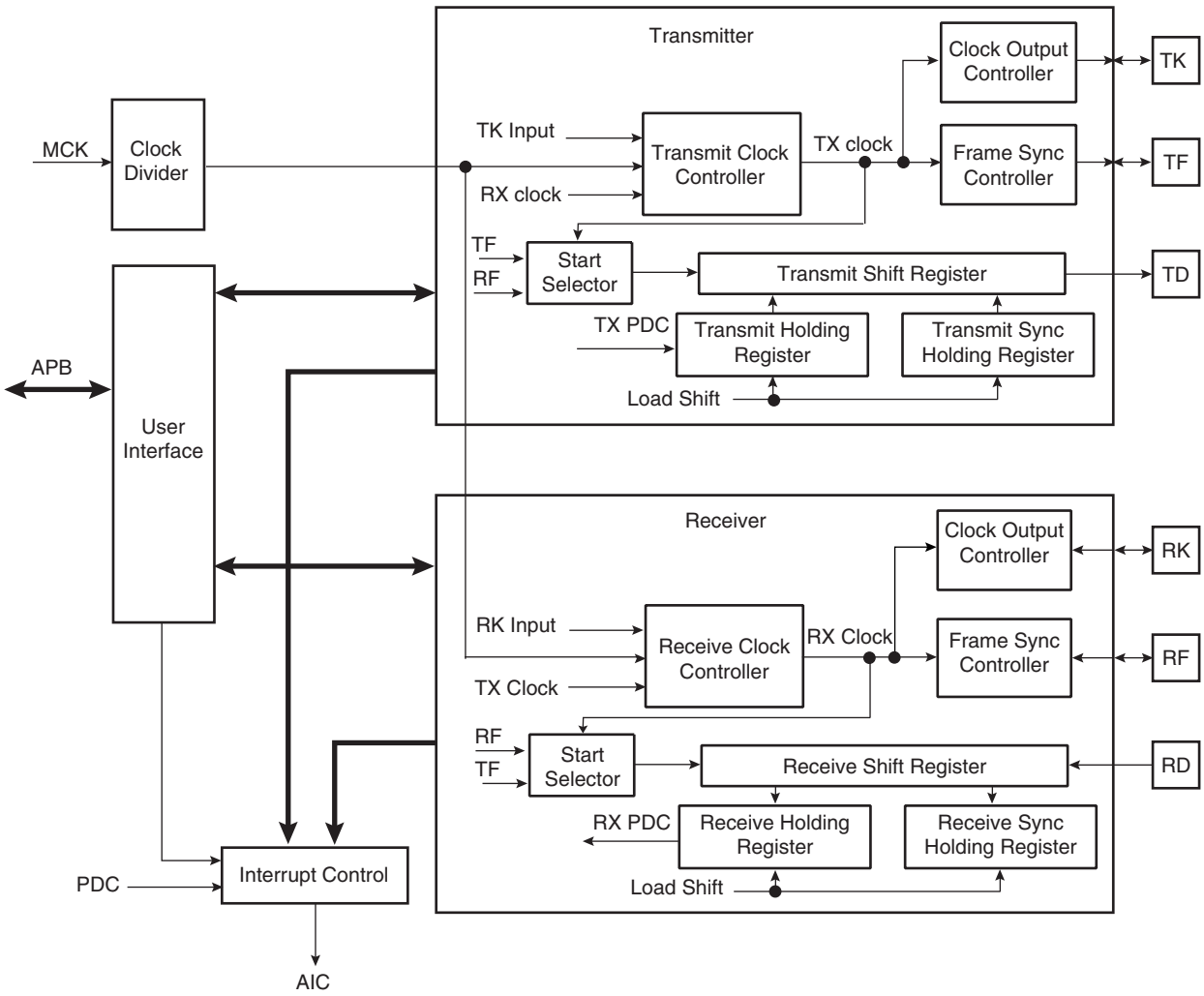
All SSC interrupts can be enabled/disabled configuring the SSC Interrupt mask register. Each pending and unmasked SSC interrupt will assert the SSC interrupt line. The SSC interrupt service routine can get the interrupt origin by reading the SSC interrupt status register.

35.6 Functional Description

This chapter contains the functional description of the following: SSC Functional Block, Clock Management, Data format, Start, Transmitter, Receiver and Frame Sync.

The receiver and transmitter operate separately. However, they can work synchronously by programming the receiver to use the transmit clock and/or to start a data transfer when transmission starts. Alternatively, this can be done by programming the transmitter to use the receive clock and/or to start a data transfer when reception starts. The transmitter and the receiver can be programmed to operate with the clock signals provided on either the TK or RK pins. This allows the SSC to support many slave-mode data transfers. The maximum clock speed allowed on the TK and RK pins is the master clock divided by 2.

Figure 35-3. SSC Functional Block Diagram



35.6.1 Clock Management

The transmitter clock can be generated by:

- an external clock received on the TK I/O pad
- the receiver clock
- the internal clock divider

The receiver clock can be generated by:

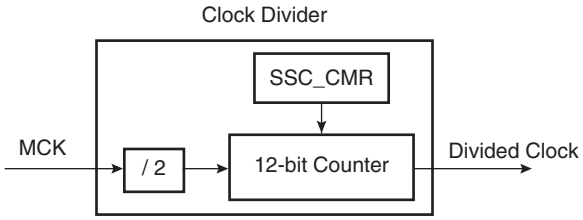
- an external clock received on the RK I/O pad
- the transmitter clock
- the internal clock divider

Furthermore, the transmitter block can generate an external clock on the TK I/O pad, and the receiver block can generate an external clock on the RK I/O pad.

This allows the SSC to support many Master and Slave Mode data transfers.

35.6.1.1 Clock Divider

Figure 35-4. Divided Clock Block Diagram



The Master Clock divider is determined by the 12-bit field DIV counter and comparator (so its maximal value is 4095) in the Clock Mode Register SSC_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 35-5. Divided Clock Generation

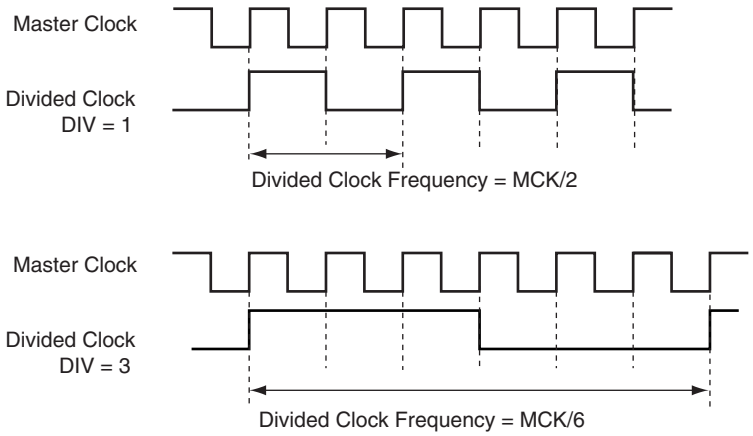


Table 35-2.

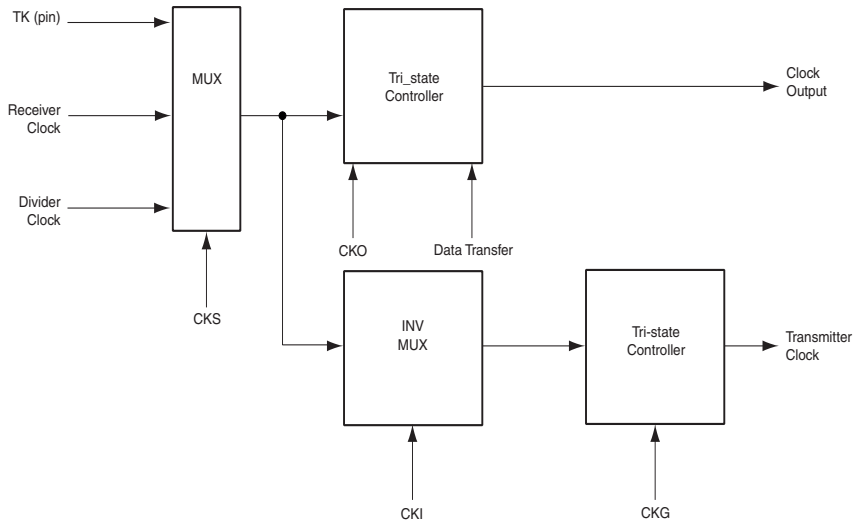
Maximum	Minimum
MCK / 2	MCK / 8190

35.6.1.2 Transmitter Clock Management

The transmitter clock is generated from the receiver clock or the divider clock or an external clock scanned on the TK I/O pad. The transmitter clock is selected by the CKS field in SSC_TCMR (Transmit Clock Mode Register). Transmit Clock can be inverted independently by the CKI bits in SSC_TCMR.

The transmitter can also drive the TK I/O pad continuously or be limited to the actual data transfer. The clock output is configured by the SSC_TCMR register. The Transmit Clock Inversion (CKI) bits have no effect on the clock outputs. Programming the TCMR register to select TK pin (CKS field) and at the same time Continuous Transmit Clock (CKO field) might lead to unpredictable results.

Figure 35-6. Transmitter Clock Management

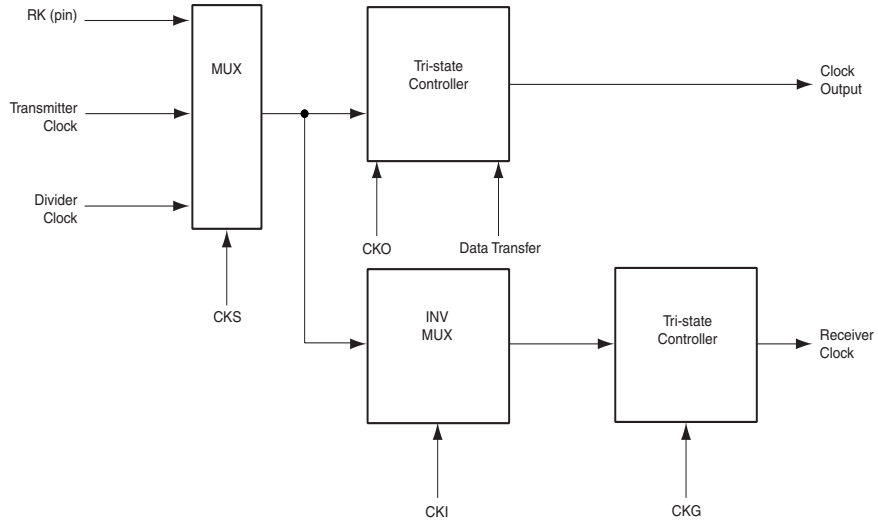


35.6.1.3 *Receiver Clock Management*

The receiver clock is generated from the transmitter clock or the divider clock or an external clock scanned on the RK I/O pad. The Receive Clock is selected by the CKS field in SSC_RCMR (Receive Clock Mode Register). Receive Clocks can be inverted independently by the CKI bits in SSC_RCMR.

The receiver can also drive the RK I/O pad continuously or be limited to the actual data transfer. The clock output is configured by the SSC_RCMR register. The Receive Clock Inversion (CKI) bits have no effect on the clock outputs. Programming the RCMR register to select RK pin (CKS field) and at the same time Continuous Receive Clock (CKO field) can lead to unpredictable results.

Figure 35-7. Receiver Clock Management



35.6.1.4 Serial Clock Ratio Considerations

The Transmitter and the Receiver can be programmed to operate with the clock signals provided on either the TK or RK pins. This allows the SSC to support many slave-mode data transfers. In this case, the maximum clock speed allowed on the RK pin is:

- Master Clock divided by 2 if Receiver Frame Synchro is input
- Master Clock divided by 3 if Receiver Frame Synchro is output

In addition, the maximum clock speed allowed on the TK pin is:

- Master Clock divided by 6 if Transmit Frame Synchro is input
- Master Clock divided by 2 if Transmit Frame Synchro is output

35.6.2 Transmitter Operations

A transmitted frame is triggered by a start event and can be followed by synchronization data before data transmission.

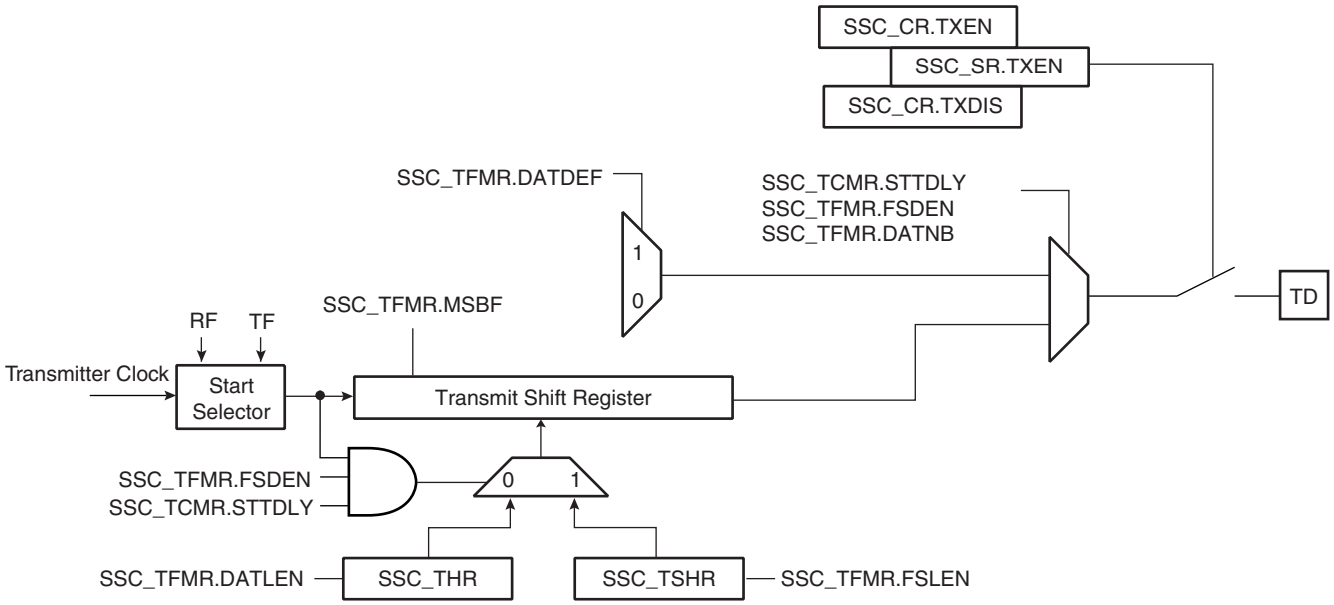
The start event is configured by setting the Transmit Clock Mode Register (SSC_TCMR). See “Start” on page 581.

The frame synchronization is configured setting the Transmit Frame Mode Register (SSC_TFMR). See “Frame Sync” on page 583.

To transmit data, the transmitter uses a shift register clocked by the transmitter clock signal and the start mode selected in the SSC_TCMR. Data is written by the application to the SSC_THR register then transferred to the shift register according to the data format selected.

When both the SSC_THR and the transmit shift register are empty, the status flag TXEMPTY is set in SSC_SR. When the Transmit Holding register is transferred in the Transmit shift register, the status flag TXRDY is set in SSC_SR and additional data can be loaded in the holding register.

Figure 35-8. Transmitter Block Diagram



35.6.3 Receiver Operations

A received frame is triggered by a start event and can be followed by synchronization data before data transmission.

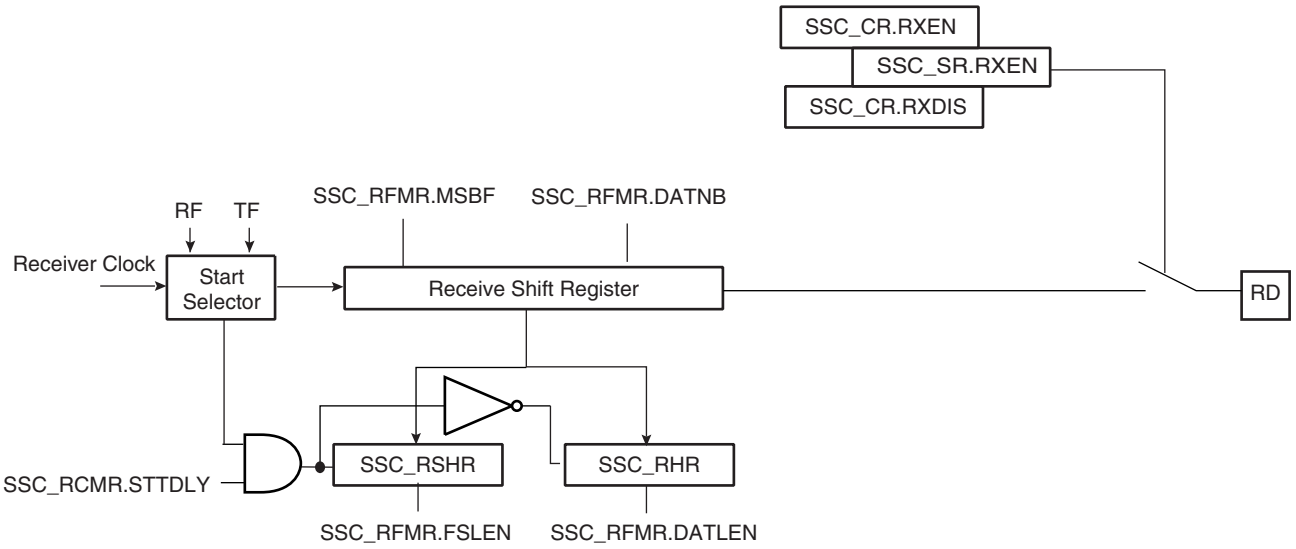
The start event is configured setting the Receive Clock Mode Register (SSC_RCMR). See “Start” on page 581.

The frame synchronization is configured setting the Receive Frame Mode Register (SSC_RFMR). See “Frame Sync” on page 583.

The receiver uses a shift register clocked by the receiver clock signal and the start mode selected in the SSC_RCMR. The data is transferred from the shift register depending on the data format selected.

When the receiver shift register is full, the SSC transfers this data in the holding register, the status flag RXRDY is set in SSC_SR and the data can be read in the receiver holding register. If another transfer occurs before read of the RHR register, the status flag OVERUN is set in SSC_SR and the receiver shift register is transferred in the RHR register.

Figure 35-9. Receiver Block Diagram



35.6.4 Start

The transmitter and receiver can both be programmed to start their operations when an event occurs, respectively in the Transmit Start Selection (START) field of SSC_TCMR and in the Receive Start Selection (START) field of SSC_RCMR.

Under the following conditions the start event is independently programmable:

- Continuous. In this case, the transmission starts as soon as a word is written in SSC_THR and the reception starts as soon as the Receiver is enabled.
- Synchronously with the transmitter/receiver
- On detection of a falling/rising edge on TF/RF
- On detection of a low level/high level on TF/RF
- On detection of a level change or an edge on TF/RF

A start can be programmed in the same manner on either side of the Transmit/Receive Clock Register (RCMR/TCMR). Thus, the start could be on TF (Transmit) or RF (Receive).

Moreover, the Receiver can start when data is detected in the bit stream with the Compare Functions.

Detection on TF/RF input/output is done by the field FSOS of the Transmit/Receive Frame Mode Register (TFMR/RFMR).

Figure 35-10. Transmit Start Mode

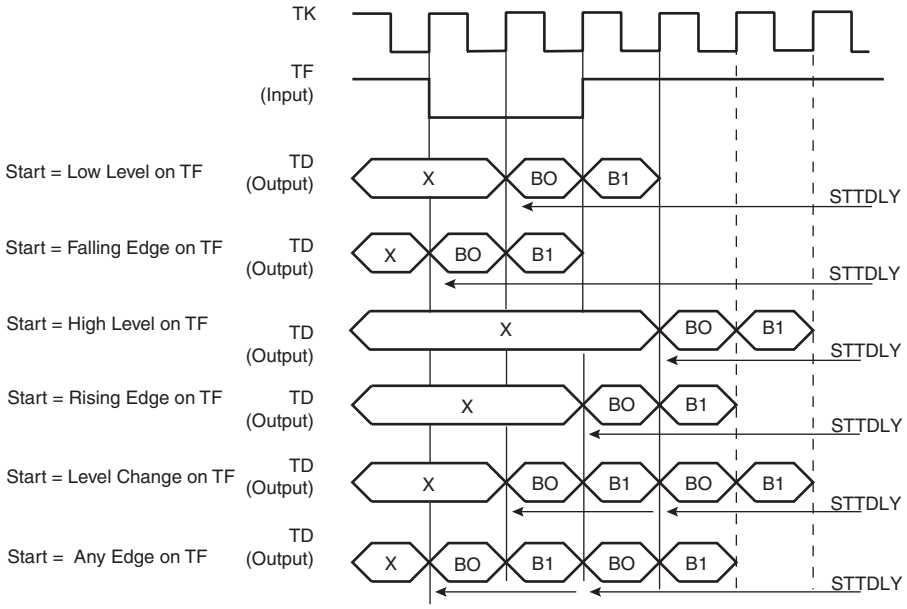
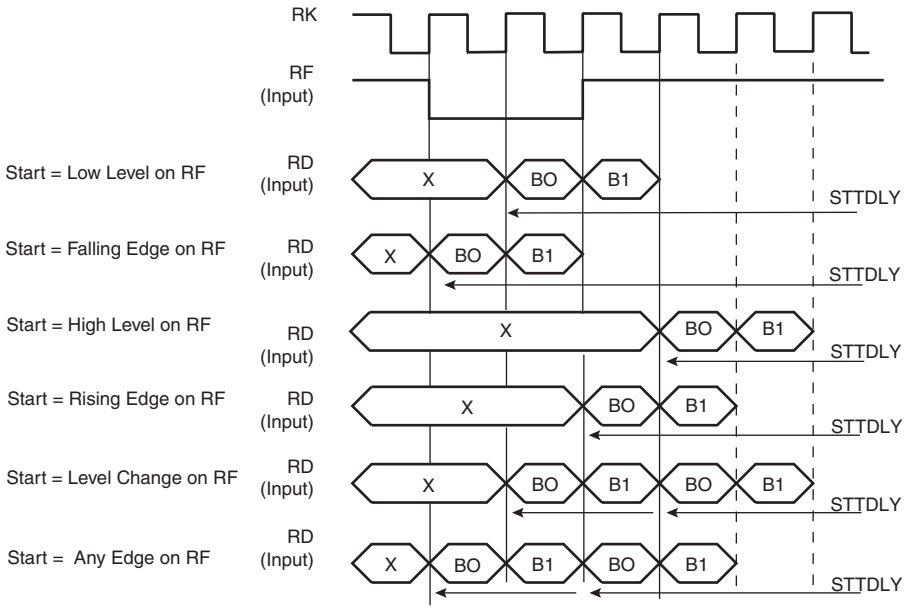


Figure 35-11. Receive Pulse/Edge Start Modes



35.6.5 Frame Sync

The Transmitter and Receiver Frame Sync pins, TF and RF, can be programmed to generate different kinds of frame synchronization signals. The Frame Sync Output Selection (FSOS) field in the Receive Frame Mode Register (SSC_RFMR) and in the Transmit Frame Mode Register (SSC_TFMR) are used to select the required waveform.

- Programmable low or high levels during data transfer are supported.
- Programmable high levels before the start of data transfers or toggling are also supported.

If a pulse waveform is selected, the Frame Sync Length (FSLEN) field in SSC_RFMR and SSC_TFMR programs the length of the pulse, from 1 bit time up to 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.

35.6.5.1 Frame Sync Data

Frame Sync Data transmits or receives a specific tag during the Frame Sync signal.

During the Frame Sync signal, the Receiver can sample the RD line and store the data in the Receive Sync Holding Register and the transmitter can transfer Transmit Sync Holding Register in the Shifter Register. The data length to be sampled/shifted out during the Frame Sync signal is programmed by the FSLEN field in SSC_RFMR/SSC_TFMR and has a maximum value of 16.

Concerning the Receive Frame Sync Data operation, if the Frame Sync Length is equal to or lower than the delay between the start event and the actual data reception, the data sampling operation is performed in the Receive Sync Holding Register through the Receive Shift Register.

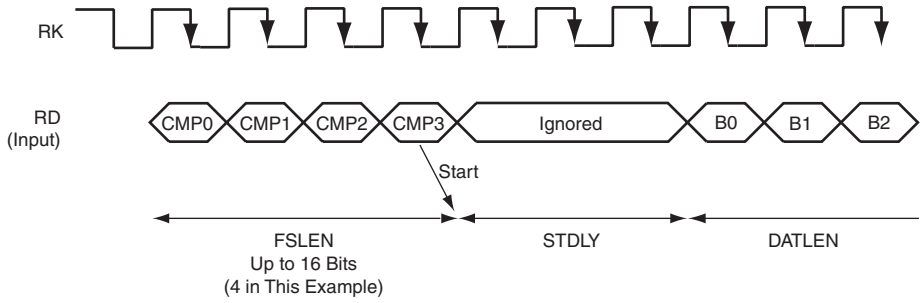
The Transmit Frame Sync Operation is performed by the transmitter only if the bit Frame Sync Data Enable (FSDEN) in SSC_TFMR is set. If the Frame Sync length is equal to or lower than the delay between the start event and the actual data transmission, the normal transmission has priority and the data contained in the Transmit Sync Holding Register is transferred in the Transmit Register, then shifted out.

35.6.5.2 Frame Sync Edge Detection

The Frame Sync Edge detection is programmed by the FSEDGE field in SSC_RFMR/SSC_TFMR. This sets the corresponding flags RXSYN/TXSYN in the SSC Status Register (SSC_SR) on frame synchro edge detection (signals RF/TF).

35.6.6 Receive Compare Modes

Figure 35-12. Receive Compare Modes



35.6.6.1 *Compare Functions*

Length of the comparison patterns (Compare 0, Compare 1) and thus the number of bits they are compared to is defined by FSLEN, but with a maximum value of 16 bits. Comparison is always done by comparing the last bits received with the comparison pattern. Compare 0 can be one start event of the Receiver. In this case, the receiver compares at each new sample the last bits received at the Compare 0 pattern contained in the Compare 0 Register (SSC_RC0R). When this start event is selected, the user can program the Receiver to start a new data transfer either by writing a new Compare 0, or by receiving continuously until Compare 1 occurs. This selection is done with the bit (STOP) in SSC_RCMR.

35.6.7 Data Format

The data framing format of both the transmitter and the receiver are programmable through the Transmitter Frame Mode Register (SSC_TFMR) and the Receiver Frame Mode Register (SSC_RFMR). In either case, the user can independently select:

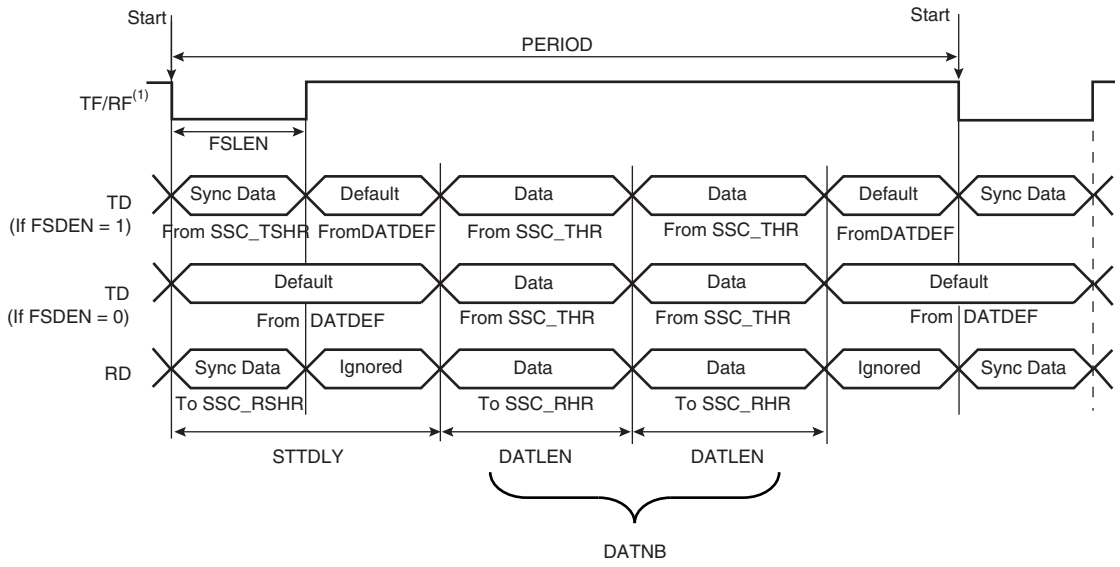
- the event that starts the data transfer (START)
- the delay in number of bit periods between the start event and the first data bit (STTDLY)
- the length of the data (DATLEN)
- the number of data to be transferred for each start event (DATNB).
- the length of synchronization transferred for each start event (FSLEN)
- the bit sense: most or lowest significant bit first (MSBF)

Additionally, the transmitter can be used to transfer synchronization and select the level driven on the TD pin while not in data transfer operation. This is done respectively by the Frame Sync Data Enable (FSDEN) and by the Data Default Value (DATDEF) bits in SSC_TFMR.

Table 35-3. Data Frame Registers

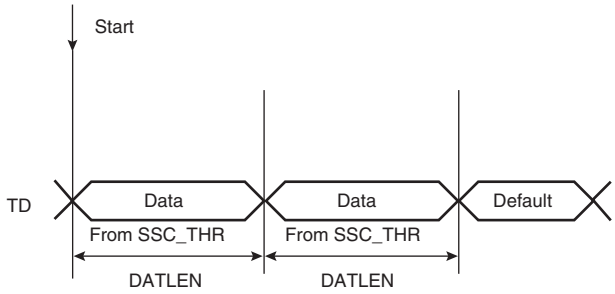
Transmitter	Receiver	Field	Length	Comment
SSC_TFMR	SSC_RFMR	DATLEN	Up to 32	Size of word
SSC_TFMR	SSC_RFMR	DATNB	Up to 16	Number of words transmitted in frame
SSC_TFMR	SSC_RFMR	MSBF		Most significant bit first
SSC_TFMR	SSC_RFMR	FSLEN	Up to 16	Size of Synchro data register
SSC_TFMR		DATDEF	0 or 1	Data default value ended
SSC_TFMR		FSDEN		Enable send SSC_TSHR
SSC_TCMR	SSC_RCMR	PERIOD	Up to 512	Frame size
SSC_TCMR	SSC_RCMR	STTDLY	Up to 255	Size of transmit start delay

Figure 35-13. Transmit and Receive Frame Format in Edge/Pulse Start Modes



Note: 1. Example of input on falling edge of TF/RF.

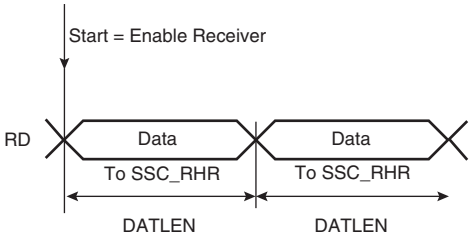
Figure 35-14. Transmit Frame Format in Continuous Mode



Start: 1. TXEMPTY set to 1
2. Write into the SSC_THR

Note: 1. STTDLY is set to 0. In this example, SSC_THR is loaded twice. FSDEN value has no effect on the transmission. SyncData cannot be output in continuous mode.

Figure 35-15. Receive Frame Format in Continuous Mode



Note: 1. STTDLY is set to 0.

35.6.8 Loop Mode

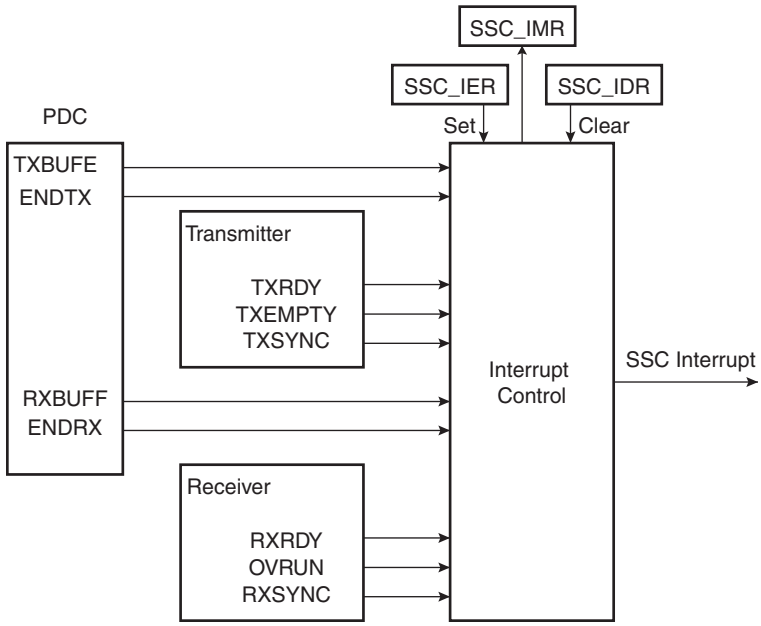
The receiver can be programmed to receive transmissions from the transmitter. This is done by setting the Loop Mode (LOOP) bit in SSC_RFMR. In this case, RD is connected to TD, RF is connected to TF and RK is connected to TK.

35.6.9 Interrupt

Most bits in SSC_SR have a corresponding bit in interrupt management registers.

The SSC can be programmed to generate an interrupt when it detects an event. The interrupt is controlled by writing SSC_IER (Interrupt Enable Register) and SSC_IDR (Interrupt Disable Register) These registers enable and disable, respectively, the corresponding interrupt by setting and clearing the corresponding bit in SSC_IMR (Interrupt Mask Register), which controls the generation of interrupts by asserting the SSC interrupt line connected to the AIC.

Figure 35-16. Interrupt Block Diagram



35.7 SSC Application Examples

The SSC can support several serial communication modes used in audio or high speed serial links. Some standard applications are shown in the following figures. All serial link applications supported by the SSC are not listed here.

Figure 35-17. Audio Application Block Diagram

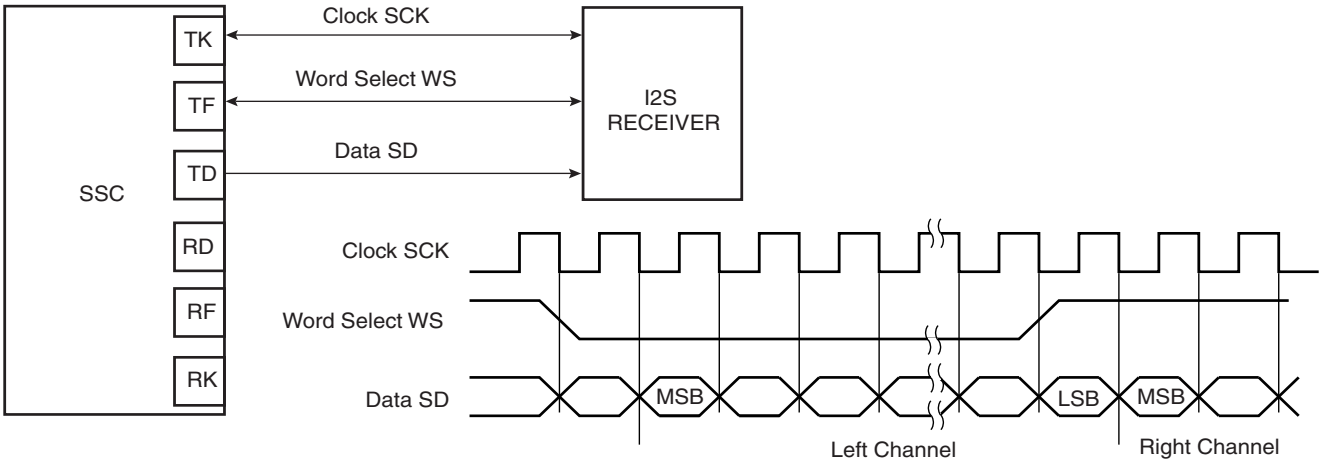


Figure 35-18. Codec Application Block Diagram

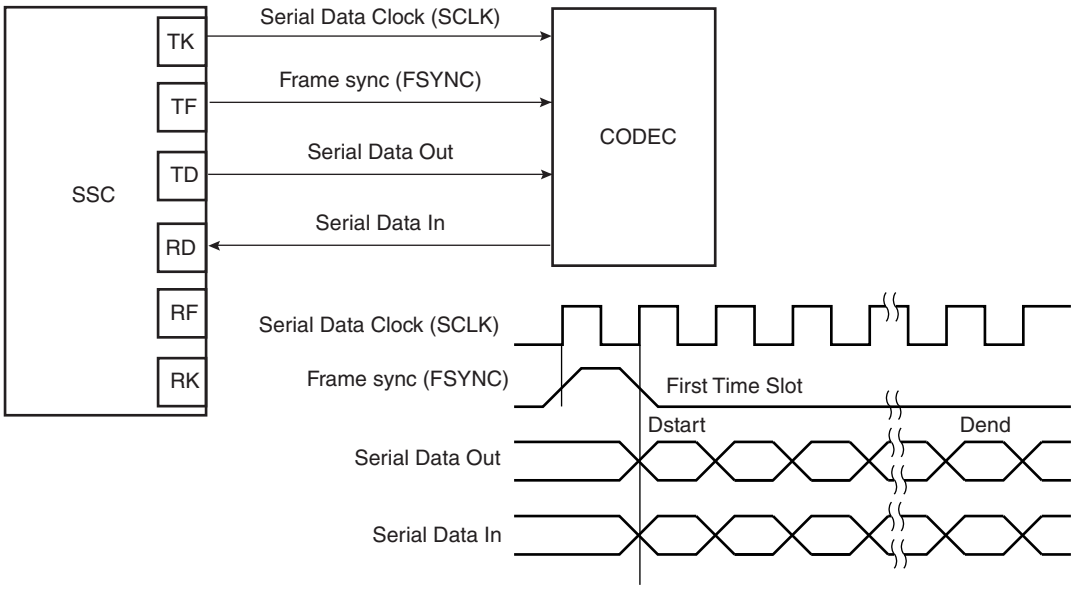
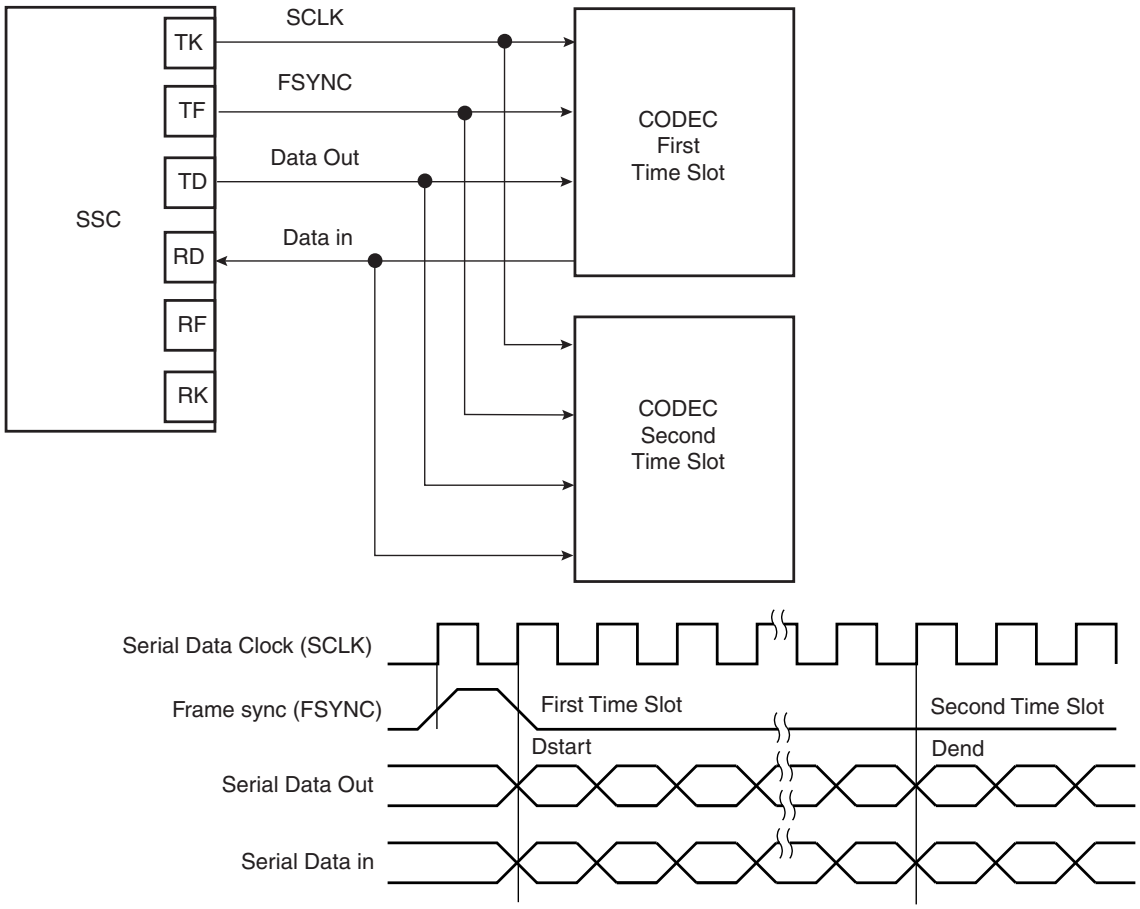


Figure 35-19. Time Slot Application Block Diagram



35.8 Synchronous Serial Controller (SSC) User Interface

Table 35-4. Register Mapping

Offset	Register	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)	–	–	–

35.8.1 SSC Control Register

Name: SSC_CR

Addresses: 0xFFF98000 (0), 0xFFF9C000 (1)

Access Type: Write-only

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

- **RXEN: Receive Enable**

0 = No effect.

1 = Enables Receive if RXDIS is not set.

- **RXDIS: Receive Disable**

0 = No effect.

1 = Disables Receive. If a character is currently being received, disables at end of current character reception.

- **TXEN: Transmit Enable**

0 = No effect.

1 = Enables Transmit if TXDIS is not set.

- **TXDIS: Transmit Disable**

0 = No effect.

1 = Disables Transmit. If a character is currently being transmitted, disables at end of current character transmission.

- **SWRST: Software Reset**

0 = No effect.

1 = Performs a software reset. Has priority on any other bit in SSC_CR.

35.8.2 SSC Clock Mode Register

Name: SSC_CMR

Addresses: 0xFFF98004 (0), 0xFFF9C004 (1)

Access Type: Read-write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	DIV			

7	6	5	4	3	2	1	0
DIV							

- **DIV: Clock Divider**

0 = The Clock Divider is not active.

Any Other Value: The Divided Clock equals the Master Clock divided by 2 times DIV. The maximum bit rate is MCK/2. The minimum bit rate is MCK/2 x 4095 = MCK/8190.

35.8.3 SSC Receive Clock Mode Register

Name: SSC_RCMR

Addresses: 0xFFF98010 (0), 0xFFF9C010 (1)

Access Type: Read-write

31	30	29	28	27	26	25	24
PERIOD							

23	22	21	20	19	18	17	16
STTDLY							

15	14	13	12	11	10	9	8
–	–	–	STOP	START			

7	6	5	4	3	2	1	0
CKG		CKI	CKO			CKS	

- **CKS: Receive Clock Selection**

CKS	Selected Receive Clock
0x0	Divided Clock
0x1	TK Clock signal
0x2	RK pin
0x3	Reserved

- **CKO: Receive Clock Output Mode Selection**

CKO	Receive Clock Output Mode	RK pin
0x0	None	Input-only
0x1	Continuous Receive Clock	Output
0x2	Receive Clock only during data transfers	Output
0x3-0x7	Reserved	

- **CKI: Receive Clock Inversion**

0 = The data inputs (Data and Frame Sync signals) are sampled on Receive Clock falling edge. The Frame Sync signal output is shifted out on Receive Clock rising edge.

1 = The data inputs (Data and Frame Sync signals) are sampled on Receive Clock rising edge. The Frame Sync signal output is shifted out on Receive Clock falling edge.

CKI affects only the Receive Clock and not the output clock signal.

- **CKG: Receive Clock Gating Selection**

CKG	Receive Clock Gating
0x0	None, continuous clock
0x1	Receive Clock enabled only if RF Low
0x2	Receive Clock enabled only if RF High
0x3	Reserved

- **START: Receive Start Selection**

START	Receive Start
0x0	Continuous, as soon as the receiver is enabled, and immediately after the end of transfer of the previous data.
0x1	Transmit start
0x2	Detection of a low level on RF signal
0x3	Detection of a high level on RF signal
0x4	Detection of a falling edge on RF signal
0x5	Detection of a rising edge on RF signal
0x6	Detection of any level change on RF signal
0x7	Detection of any edge on RF signal
0x8	Compare 0
0x9-0xF	Reserved

- **STOP: Receive Stop Selection**

0 = After completion of a data transfer when starting with a Compare 0, the receiver stops the data transfer and waits for a new compare 0.

1 = After starting a receive with a Compare 0, the receiver operates in a continuous mode until a Compare 1 is detected.

- **STTDLY: Receive Start Delay**

If STTDLY is not 0, a delay of STTDLY clock cycles is inserted between the start event and the actual start of reception. When the Receiver is programmed to start synchronously with the Transmitter, the delay is also applied.

Note: It is very important that STTDLY be set carefully. If STTDLY must be set, it should be done in relation to TAG (Receive Sync Data) reception.

- **PERIOD: Receive Period Divider Selection**

This field selects the divider to apply to the selected Receive Clock in order to generate a new Frame Sync Signal. If 0, no PERIOD signal is generated. If not 0, a PERIOD signal is generated each 2 x (PERIOD+1) Receive Clock.

35.8.4 SSC Receive Frame Mode Register

Name: SSC_RFMR

Addresses: 0xFFF98014 (0), 0xFFF9C014 (1)

Access Type: Read-write

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

- **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

35.8.5 SSC Transmit Clock Mode Register

Name: SSC_TCMR

Addresses: 0xFFF98018 (0), 0xFFF9C018 (1)

Access Type: Read-write

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

- **CKS: Transmit Clock Selection**

CKS	Selected Transmit Clock
0x0	Divided Clock
0x1	RK Clock signal
0x2	TK Pin
0x3	Reserved

- **CKO: Transmit Clock Output Mode Selection**

CKO	Transmit Clock Output Mode	TK pin
0x0	None	Input-only
0x1	Continuous Transmit Clock	Output
0x2	Transmit Clock only during data transfers	Output
0x3-0x7	Reserved	

- **CKI: Transmit Clock Inversion**

0 = The data outputs (Data and Frame Sync signals) are shifted out on Transmit Clock falling edge. The Frame sync signal input is sampled on Transmit clock rising edge.

1 = The data outputs (Data and Frame Sync signals) are shifted out on Transmit Clock rising edge. The Frame sync signal input is sampled on Transmit clock falling edge.

CKI affects only the Transmit Clock and not the output clock signal.

- **CKG: Transmit Clock Gating Selection**

CKG	Transmit Clock Gating
0x0	None, continuous clock
0x1	Transmit Clock enabled only if TF Low
0x2	Transmit Clock enabled only if TF High
0x3	Reserved

- **START: Transmit Start Selection**

START	Transmit Start
0x0	Continuous, as soon as a word is written in the SSC_THR Register (if Transmit is enabled), and immediately after the end of transfer of the previous data.
0x1	Receive start
0x2	Detection of a low level on TF signal
0x3	Detection of a high level on TF signal
0x4	Detection of a falling edge on TF signal
0x5	Detection of a rising edge on TF signal
0x6	Detection of any level change on TF signal
0x7	Detection of any edge on TF signal
0x8 - 0xF	Reserved

- **STTDLY: Transmit Start Delay**

If STTDLY is not 0, a delay of STTDLY clock cycles is inserted between the start event and the actual start of transmission of data. When the Transmitter is programmed to start synchronously with the Receiver, the delay is also applied.

Note: STTDLY must be set carefully. If STTDLY is too short in respect to TAG (Transmit Sync Data) emission, data is emitted instead of the end of TAG.

- **PERIOD: Transmit Period Divider Selection**

This field selects the divider to apply to the selected Transmit Clock to generate a new Frame Sync Signal. If 0, no period signal is generated. If not 0, a period signal is generated at each 2 x (PERIOD+1) Transmit Clock.

35.8.6 SSC Transmit Frame Mode Register

Name: SSC_TFMR

Addresses: 0xFFF9801C (0), 0xFFF9C01C (1)

Access Type: Read-write

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

- **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

35.8.7 SSC Receive Holding Register

Name: SSC_RHR

Addresses: 0xFFF98020 (0), 0xFFF9C020 (1)

Access Type: Read-only

31	30	29	28	27	26	25	24
RDAT							

23	22	21	20	19	18	17	16
RDAT							

15	14	13	12	11	10	9	8
RDAT							

7	6	5	4	3	2	1	0
RDAT							

- **RDAT: Receive Data**

Right aligned regardless of the number of data bits defined by DATLEN in SSC_RFMR.

35.8.8 SSC Transmit Holding Register

Name: SSC_THR

Addresses: 0xFFF98024 (0), 0xFFF9C024 (1)

Access Type: Write-only

31	30	29	28	27	26	25	24
TDAT							

23	22	21	20	19	18	17	16
TDAT							

15	14	13	12	11	10	9	8
TDAT							

7	6	5	4	3	2	1	0
TDAT							

- **TDAT: Transmit Data**

Right aligned regardless of the number of data bits defined by DATLEN in SSC_TFMR.

35.8.9 SSC Receive Synchronization Holding Register

Name: SSC_RSHR

Addresses: 0xFFF98030 (0), 0xFFF9C030 (1)

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
RSDAT							

7	6	5	4	3	2	1	0
RSDAT							

- **RSDAT: Receive Synchronization Data**

35.8.10 SSC Transmit Synchronization Holding Register

Name: SSC_TSHR

Addresses: 0xFFF98034 (0), 0xFFF9C034 (1)

Access Type: Read-write

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

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

15	14	13	12	11	10	9	8
TSDAT							

7	6	5	4	3	2	1	0
TSDAT							

- **TSDAT: Transmit Synchronization Data**

35.8.11 SSC Receive Compare 0 Register

Name: SSC_RC0R

Addresses: 0xFFF98038 (0), 0xFFF9C038 (1)

Access Type: Read-write

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

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

15	14	13	12	11	10	9	8
CP0							

7	6	5	4	3	2	1	0
CP0							

- **CP0: Receive Compare Data 0**

35.8.12 SSC Receive Compare 1 Register

Name: SSC_RC1R

Addresses: 0xFFF9803C (0), 0xFFF9C03C (1)

Access Type: Read-write

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

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

15	14	13	12	11	10	9	8
CP1							

7	6	5	4	3	2	1	0
CP1							

- **CP1: Receive Compare Data 1**

35.8.13 SSC Status Register

Name: SSC_SR

Addresses: 0xFFF98040 (0), 0xFFF9C040 (1)

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	RXEN	TXEN
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready**

0 = Data has been loaded in SSC_THR and is waiting to be loaded in the Transmit Shift Register (TSR).

1 = SSC_THR is empty.

- **TXEMPTY: Transmit Empty**

0 = Data remains in SSC_THR or is currently transmitted from TSR.

1 = Last data written in SSC_THR has been loaded in TSR and last data loaded in TSR has been transmitted.

- **ENDTX: End of Transmission**

0 = The register SSC_TCR has not reached 0 since the last write in SSC_TCR or SSC_TNCR.

1 = The register SSC_TCR has reached 0 since the last write in SSC_TCR or SSC_TNCR.

- **TXBUFE: Transmit Buffer Empty**

0 = SSC_TCR or SSC_TNCR have a value other than 0.

1 = Both SSC_TCR and SSC_TNCR have a value of 0.

- **RXRDY: Receive Ready**

0 = SSC_RHR is empty.

1 = Data has been received and loaded in SSC_RHR.

- **OVRUN: Receive Overrun**

0 = No data has been loaded in SSC_RHR while previous data has not been read since the last read of the Status Register.

1 = Data has been loaded in SSC_RHR while previous data has not yet been read since the last read of the Status Register.

- **ENDRX: End of Reception**

0 = Data is written on the Receive Counter Register or Receive Next Counter Register.

1 = End of PDC transfer when Receive Counter Register has arrived at zero.

- **RXBUFF: Receive Buffer Full**

0 = SSC_RCR or SSC_RNCR have a value other than 0.

1 = Both SSC_RCR and SSC_RNCR have a value of 0.

- **CP0: Compare 0**

0 = A compare 0 has not occurred since the last read of the Status Register.

1 = A compare 0 has occurred since the last read of the Status Register.

- **CP1: Compare 1**

0 = A compare 1 has not occurred since the last read of the Status Register.

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

- **TXSYN: Transmit Sync**

0 = A Tx Sync has not occurred since the last read of the Status Register.

1 = A Tx Sync has occurred since the last read of the Status Register.

- **RXSYN: Receive Sync**

0 = An Rx Sync has not occurred since the last read of the Status Register.

1 = An Rx Sync has occurred since the last read of the Status Register.

- **TXEN: Transmit Enable**

0 = Transmit is disabled.

1 = Transmit is enabled.

- **RXEN: Receive Enable**

0 = Receive is disabled.

1 = Receive is enabled.

35.8.14 SSC Interrupt Enable Register

Name: SSC_IER

Addresses: 0xFFF98044 (0), 0xFFF9C044 (1)

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Enable**

0 = 0 = No effect.

1 = Enables the Transmit Ready Interrupt.

- **TXEMPTY: Transmit Empty Interrupt Enable**

0 = No effect.

1 = Enables the Transmit Empty Interrupt.

- **ENDTX: End of Transmission Interrupt Enable**

0 = No effect.

1 = Enables the End of Transmission Interrupt.

- **TXBUFE: Transmit Buffer Empty Interrupt Enable**

0 = No effect.

1 = Enables the Transmit Buffer Empty Interrupt

- **RXRDY: Receive Ready Interrupt Enable**

0 = No effect.

1 = Enables the Receive Ready Interrupt.

- **OVRUN: Receive Overrun Interrupt Enable**

0 = No effect.

1 = Enables the Receive Overrun Interrupt.

- **ENDRX: End of Reception Interrupt Enable**

0 = No effect.

1 = Enables the End of Reception Interrupt.

- **RXBUFF: Receive Buffer Full Interrupt Enable**

0 = No effect.

1 = Enables the Receive Buffer Full Interrupt.

- **CP0: Compare 0 Interrupt Enable**

0 = No effect.

1 = Enables the Compare 0 Interrupt.

- **CP1: Compare 1 Interrupt Enable**

0 = No effect.

1 = Enables the Compare 1 Interrupt.

- **TXSYN: Tx Sync Interrupt Enable**

0 = No effect.

1 = Enables the Tx Sync Interrupt.

- **RXSYN: Rx Sync Interrupt Enable**

0 = No effect.

1 = Enables the Rx Sync Interrupt.

35.8.15 SSC Interrupt Disable Register

Name: SSC_IDR

Addresses: 0xFFF98048 (0), 0xFFF9C048 (1)

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Disable**

0 = No effect.

1 = Disables the Transmit Ready Interrupt.

- **TXEMPTY: Transmit Empty Interrupt Disable**

0 = No effect.

1 = Disables the Transmit Empty Interrupt.

- **ENDTX: End of Transmission Interrupt Disable**

0 = No effect.

1 = Disables the End of Transmission Interrupt.

- **TXBUFE: Transmit Buffer Empty Interrupt Disable**

0 = No effect.

1 = Disables the Transmit Buffer Empty Interrupt.

- **RXRDY: Receive Ready Interrupt Disable**

0 = No effect.

1 = Disables the Receive Ready Interrupt.

- **OVRUN: Receive Overrun Interrupt Disable**

0 = No effect.

1 = Disables the Receive Overrun Interrupt.

- **ENDRX: End of Reception Interrupt Disable**

0 = No effect.

1 = Disables the End of Reception Interrupt.

- **RXBUFF: Receive Buffer Full Interrupt Disable**

0 = No effect.

1 = Disables the Receive Buffer Full Interrupt.

- **CP0: Compare 0 Interrupt Disable**

0 = No effect.

1 = Disables the Compare 0 Interrupt.

- **CP1: Compare 1 Interrupt Disable**

0 = No effect.

1 = Disables the Compare 1 Interrupt.

- **TXSYN: Tx Sync Interrupt Enable**

0 = No effect.

1 = Disables the Tx Sync Interrupt.

- **RXSYN: Rx Sync Interrupt Enable**

0 = No effect.

1 = Disables the Rx Sync Interrupt.

35.8.16 SSC Interrupt Mask Register

Name: SSC_IMR

Addresses: 0xFFF9804C (0), 0xFFF9C04C (1)

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	CP1	CP0
7	6	5	4	3	2	1	0
RXBUF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: Transmit Ready Interrupt Mask**

0 = The Transmit Ready Interrupt is disabled.

1 = The Transmit Ready Interrupt is enabled.

- **TXEMPTY: Transmit Empty Interrupt Mask**

0 = The Transmit Empty Interrupt is disabled.

1 = The Transmit Empty Interrupt is enabled.

- **ENDTX: End of Transmission Interrupt Mask**

0 = The End of Transmission Interrupt is disabled.

1 = The End of Transmission Interrupt is enabled.

- **TXBUFE: Transmit Buffer Empty Interrupt Mask**

0 = The Transmit Buffer Empty Interrupt is disabled.

1 = The Transmit Buffer Empty Interrupt is enabled.

- **RXRDY: Receive Ready Interrupt Mask**

0 = The Receive Ready Interrupt is disabled.

1 = The Receive Ready Interrupt is enabled.

- **OVRUN: Receive Overrun Interrupt Mask**

0 = The Receive Overrun Interrupt is disabled.

1 = The Receive Overrun Interrupt is enabled.

- **ENDRX: End of Reception Interrupt Mask**

0 = The End of Reception Interrupt is disabled.

1 = The End of Reception Interrupt is enabled.

- **RXBUFF: Receive Buffer Full Interrupt Mask**

0 = The Receive Buffer Full Interrupt is disabled.

1 = The Receive Buffer Full Interrupt is enabled.

- **CP0: Compare 0 Interrupt Mask**

0 = The Compare 0 Interrupt is disabled.

1 = The Compare 0 Interrupt is enabled.

- **CP1: Compare 1 Interrupt Mask**

0 = The Compare 1 Interrupt is disabled.

1 = The Compare 1 Interrupt is enabled.

- **TXSYN: Tx Sync Interrupt Mask**

0 = The Tx Sync Interrupt is disabled.

1 = The Tx Sync Interrupt is enabled.

- **RXSYN: Rx Sync Interrupt Mask**

0 = The Rx Sync Interrupt is disabled.

1 = The Rx Sync Interrupt is enabled.

36. AC97 Controller (AC97C)

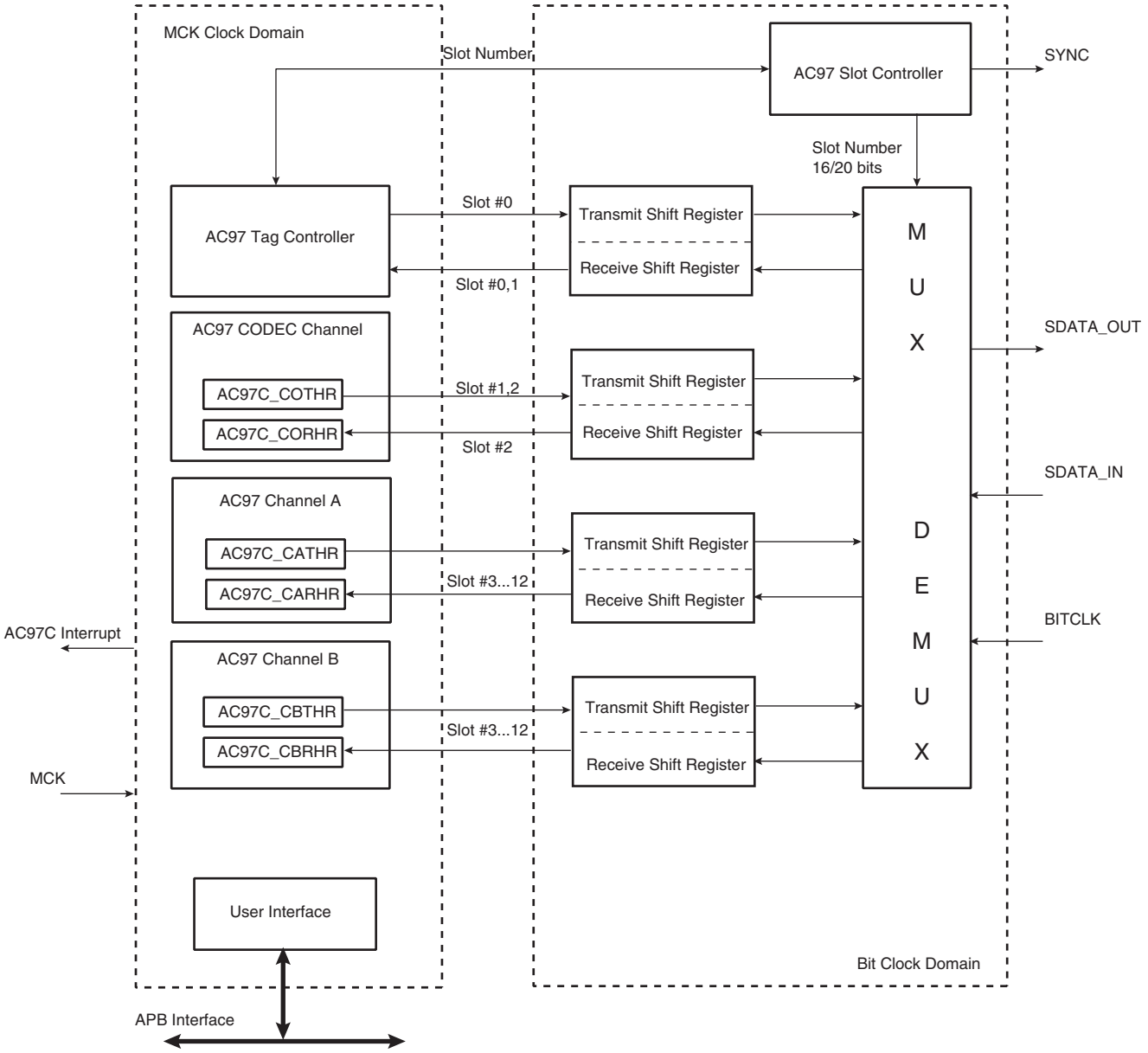
36.1 Overview

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.

36.2 Block Diagram

Figure 36-1. Functional Block Diagram



36.3 Pin Name List

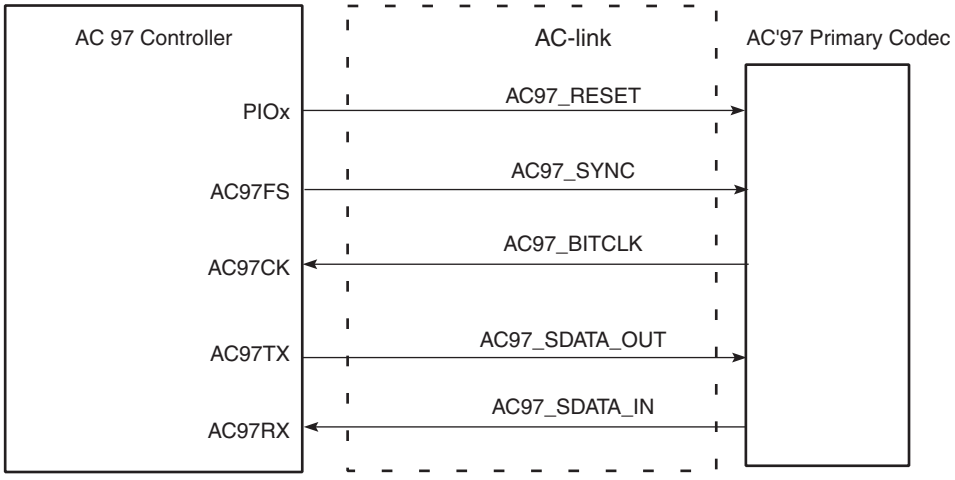
Table 36-1. I/O Lines Description

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.

36.4 Application Block Diagram

Figure 36-2. Application Block diagram



36.5 Product Dependencies

36.5.1 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.

36.5.2 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.

36.5.3 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).

36.6 Functional Description

36.6.1 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 36-3. Bidirectional AC-link Frame with Slot Assignment

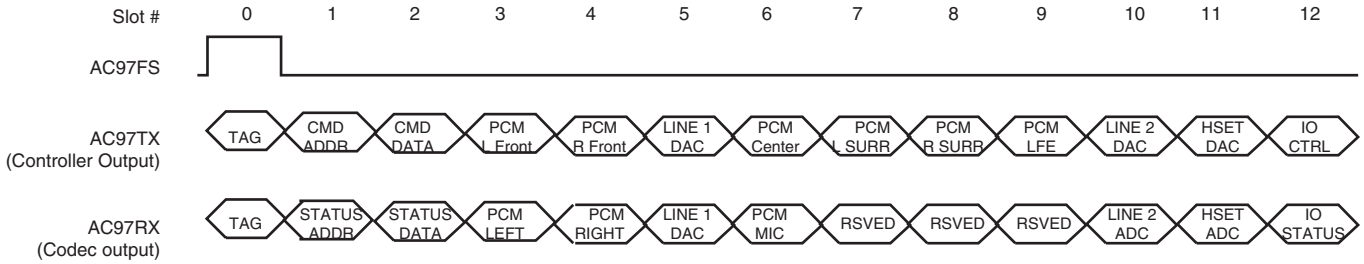


Table 36-2. AC-link Output Slots Transmitted from the AC97C Controller

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 36-3. 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

36.6.1.1 Slot Description

36.6.1.2 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.

36.6.1.3 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.

36.6.1.4 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.

36.6.1.5 Data Slots [3:12]

Slots [3:12] are 20-bit wide data slots, they usually carry audio PCM or/and modem I/O data.

36.6.2 AC97 Controller Channel Organization

The AC97 Controller features a Codec channel and 2 logical channels: Channel A, Channel B.

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, Channel B 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 or Channel B . The slot to channel assignment is configured by two registers:

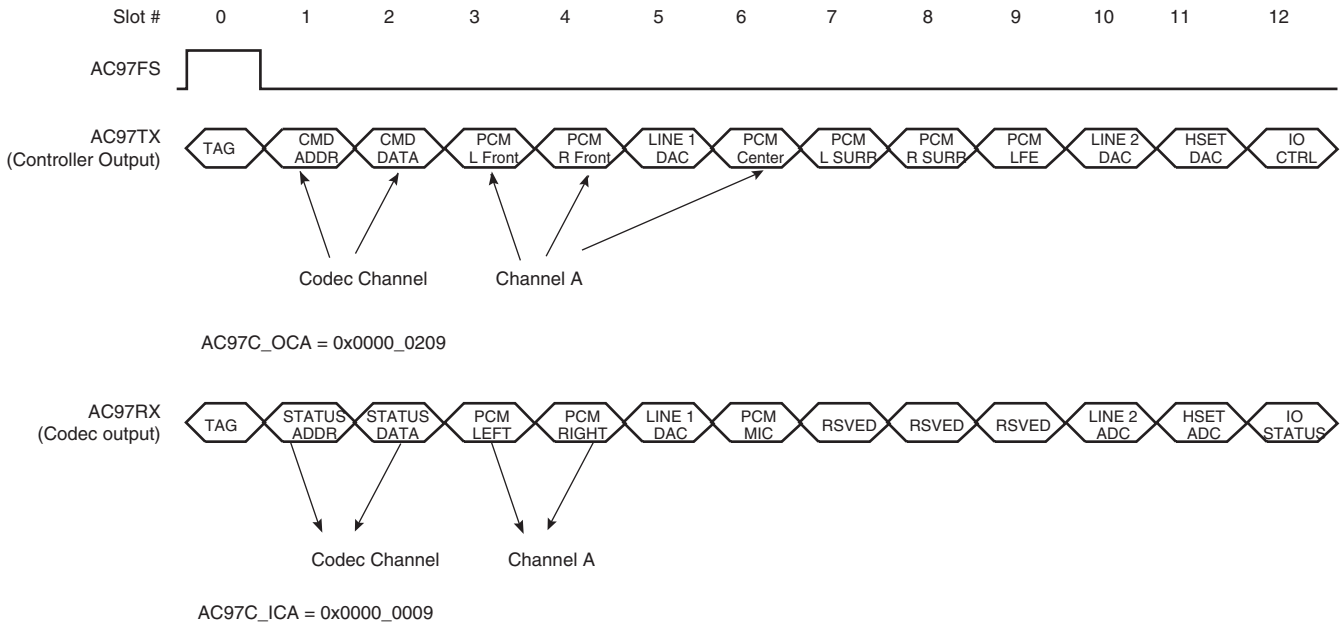
- AC97 Controller Input Channel Assignment Register (AC97C_ICA)
- AC97 Controller Output Channel Assignment Register (AC97C_OCA)

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, or to Channel Bthrough 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 36-4. Logical Channel Assignment



36.6.2.1 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, Channel Bby writing the SIZE bit field in AC97C Channel x Mode Register (AC97C_CAMR), (AC97C_CBMR). 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, Channel B by writing CEM bit field in (AC97C_CAMR), (AC97C_CBMR) 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 and/or Channel B by writing CEN bit field in AC97C_CxMR register.

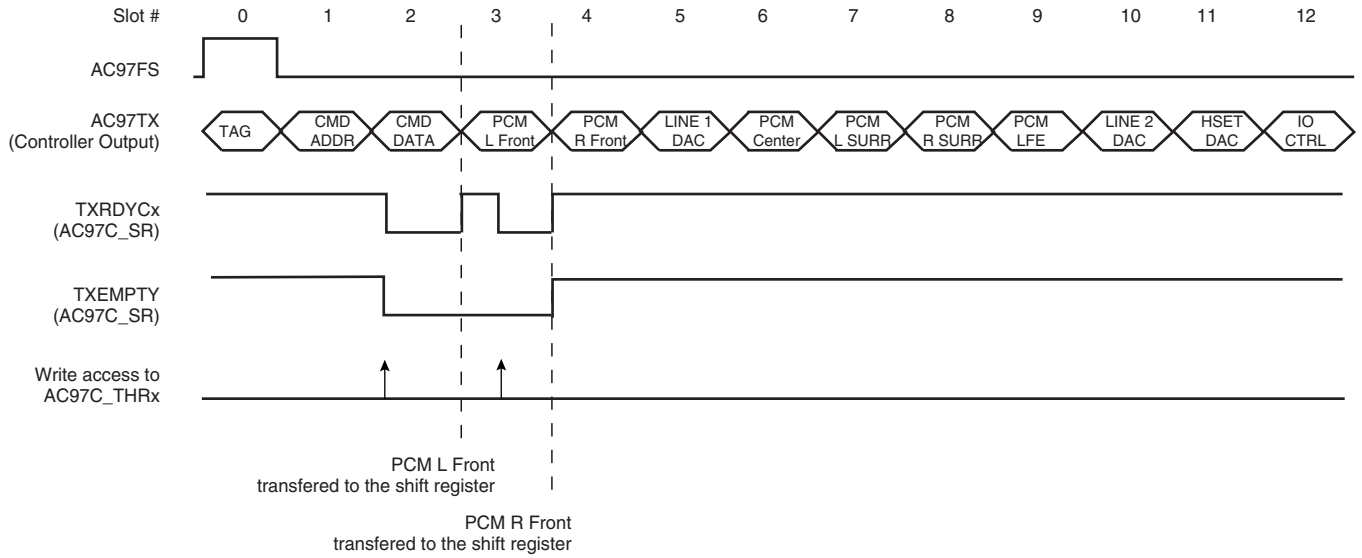
36.6.2.2 Transmit Operation

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 36-5. 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.

36.6.2.3 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.

36.6.2.4 Receive Operation

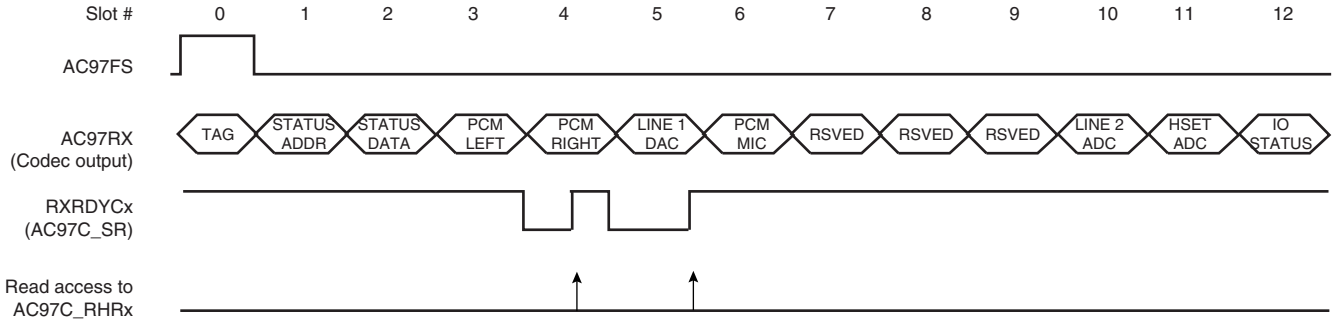
The AC97 Controller can also receive data from AC97 Codec. Data is received in the channel's shift register and then transferred to the AC97 Controller Channel x Read Holding Register. To read the newly received data, the application must perform the following steps:

- 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 36-6. 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.

36.6.2.5 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.

36.6.2.6 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.

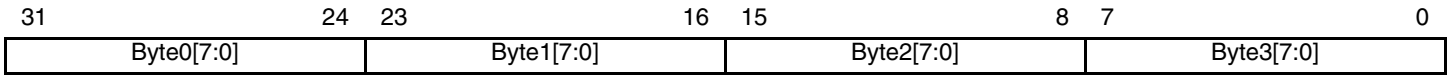
The application can disable event interrupts by writing in AC97 Controller Interrupt Disable Register (AC97C_IDR). The AC97 Controller Interrupt Mask Register (AC97C_IMR) shows which event can trigger an interrupt and which one cannot.

36.6.2.7 Endianness

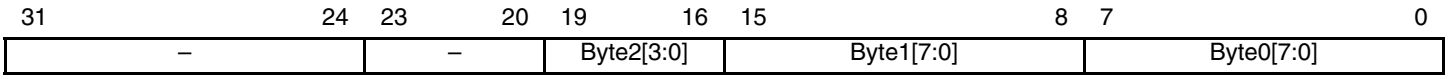
Endianness can be managed automatically for each channel, except for the Codec channel, by writing to Channel Endianness Mode (CEM) in AC97C_CxMR. This enables transferring data on AC-link in Big Endian format without any additional operation.

36.6.2.8 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).



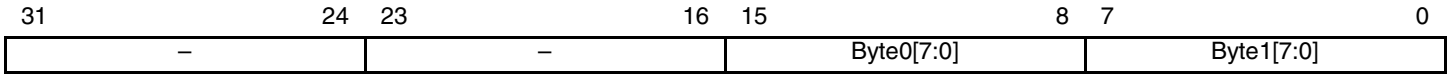
Word stored in Channel x Transmit Holding Register (AC97C_CxTHR) (data to transmit).



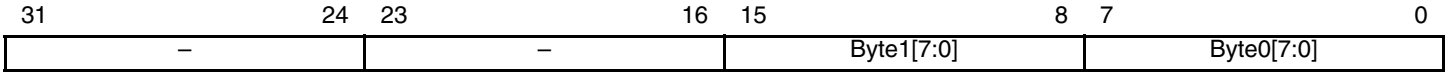
Data transmitted on appropriate slot: data[19:0] = {Byte2[3:0], Byte1[7:0], Byte0[7:0]}.

36.6.2.9 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).



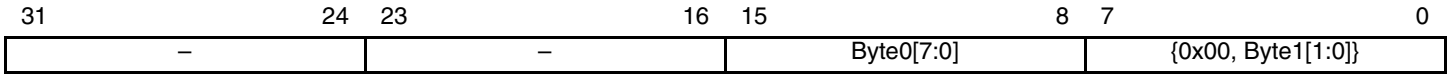
Halfword stored in AC97 Controller Channel x Transmit Holding Register (AC97C_CxTHR) (data to transmit).



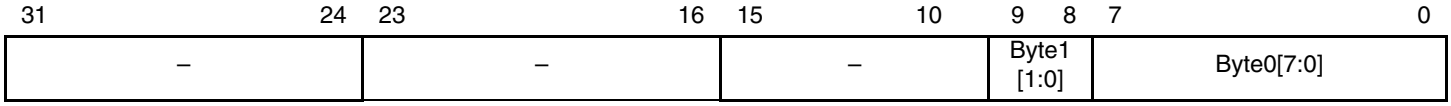
Data emitted on related slot: data[19:0] = {0x0, Byte1[7:0], Byte0[7:0]}.

36.6.2.10 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).



Halfword stored in AC97 Controller Channel x Transmit Holding Register (AC97C_CxTHR) (data to transmit).



Data emitted on related slot: data[19:0] = {0x000, Byte1[1:0], Byte0[7:0]}.

36.6.2.11 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

36.6.2.12 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]

36.6.2.13 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]

36.6.3 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.

36.6.4 Power Management

36.6.4.1 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.
- Disable Channel B clearing CEN field in the AC97C_CBMR register.
- Write 0x2680 value in the AC97C_COTHR register.
- Poll the TXEMPTY flag in AC97C_CxSR registers for the 2 channels.

At this point AC97 Codec is in low power mode.

36.6.4.2 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.

36.6.4.3 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.

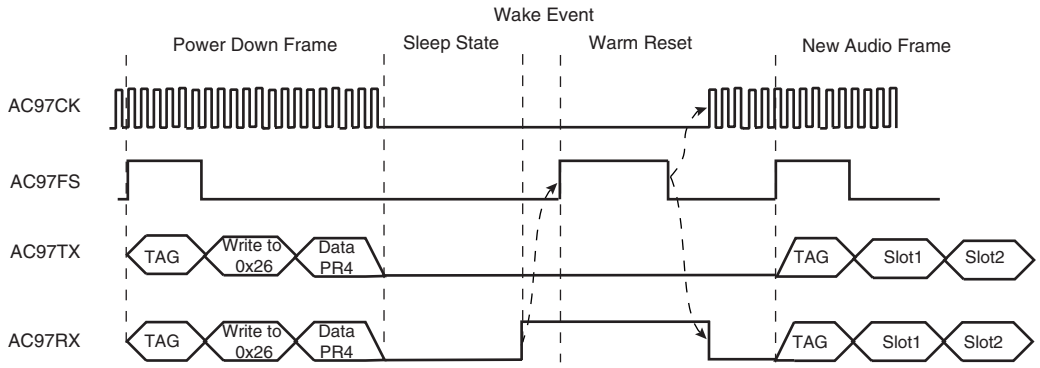
36.6.4.4 Wake-up Triggered by the AC97 Codec

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 36-7. AC97 Power-Down/Up Sequence



36.6.4.5 *AC97 Codec Reset*

There are three ways to reset an AC97 Codec.

36.6.4.6 *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.

36.6.4.7 *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.

36.7 AC97 Controller (AC97C) User Interface

Table 36-4. 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
0x30	Channel B Receive Holding Register	AC97C_CBRHR	Read	0x0
0x34	Channel B Transmit Holding Register	AC97C_CBTHR	Write	–
0x38	Channel B Status Register	AC97C_CBSR	Read	0x0
0x3C	Channel B Mode Register	AC97C_CBMR	Read-write	0x0
0x40	Codec 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	–	–	–

36.7.1 AC97 Controller Mode Register

Register Name: AC97C_MR

Address: 0xFFFA0008

Access Type: Read-Write

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

- **VRA: Variable Rate (for Data Slots 3-12)**

0: Variable Rate is inactive. (48 KHz only)

1: Variable Rate is active.

- **WRST: Warm Reset**

0: Warm Reset is inactive.

1: Warm Reset is active.

- **ENA: 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.

36.7.2 AC97 Controller Input Channel Assignment Register

Register Name:AC97C_ICA

Address: 0xFFFA0010

Access Type: Read-write

31	30	29	28	27	26	25	24
–	–	CHID12			CHID11		
23	22	21	20	19	18	17	16
CHID10			CHID9			CHID8	
15	14	13	12	11	10	9	8
CHID8	CHID7			CHID6			CHID5
7	6	5	4	3	2	1	0
CHID5		CHID4			CHID3		

- **CHIDx: Channel ID** for the input slot x

CHIDx	Selected Receive Channel
0x0	None. No data will be received during this slot time
0x1	Channel A data will be received during this slot time.
0x2	Channel B data will be received during this slot time

36.7.3 AC97 Controller Output Channel Assignment Register

Register Name:AC97C_OCA

Address: 0xFFFA0014

Access Type: Read-write

31	30	29	28	27	26	25	24
–	–	CHID12			CHID11		
23	22	21	20	19	18	17	16
CHID10			CHID9			CHID8	
15	14	13	12	11	10	9	8
CHID8	CHID7			CHID6			CHID5
7	6	5	4	3	2	1	0
CHID5		CHID4			CHID3		

- **CHIDx: Channel ID** for the output slot x

CHIDx	Selected Transmit Channel
0x0	None. No data will be transmitted during this slot time
0x1	Channel A data will be transferred during this slot time.
0x2	Channel B data will be transferred during this slot time

36.7.4 AC97 Controller Codec Channel Receive Holding Register

Register Name:AC97C_CORHR

Address: 0xFFFA0040

Access Type: Read-only

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

- **SDATA: Status Data**

Data sent by the CODEC in the third AC97 input frame slot (Slot 2).

36.7.5 AC97 Controller Codec Channel Transmit Holding Register

Register Name:AC97C_COTHR

Address: 0xFFFA0044

Access Type: Write-only

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

- **READ: Read-write command**

0: Write operation to the CODEC register indexed by the CADDR address.

1: Read operation to the CODEC register indexed by the CADDR address.

This flag is sent during the second 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).

36.7.6 AC97 Controller Channel A, Channel B, Receive Holding Register

Register Name:AC97C_CARHR, AC97C_CBRHR

Address: 0xFFFA0020

Address: 0xFFFA0030

Access Type: Read-only

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

- **RDATA: Receive Data**

Received Data on channel x.

36.7.7 AC97 Controller Channel A, Channel B, Transmit Holding Register

Register Name:AC97C_CATHR, AC97C_CBTHR

Address: 0xFFFA0024

Address: 0xFFFA0034

Access Type: Write-only

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

- **TDATA: Transmit Data**

Data to be sent on channel x.

36.7.8 AC97 Controller Channel A Status Register

Register Name:AC97C_CASR

Address: 0xFFFA0028

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
RXBUFF	ENDRX	–	–	TXBUFE	ENDTX	–	–
7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	UNRUN	TXEMPTY	TXRDY

- **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.

36.7.9 AC97 Controller Channel B Status Register

Register Name:AC97C_CBSR

Address: 0xFFFA0038

Access Type: Read-only

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

- **TXRDY: Channel Transmit Ready**

0: Data has been loaded in Channel Transmit Register and is waiting to be loaded in the Channel Transmit Shift Register.

1: Channel Transmit Register is empty.

- **TXEMPTY: Channel Transmit Empty**

0: Data remains in the Channel Transmit Register or is currently transmitted from the Channel Transmit Shift Register.

1: Data in the Channel Transmit Register have been loaded in the Channel Transmit Shift Register and sent to the codec.

- **UNRUN: Transmit Underrun**

Active only when Variable Rate Mode is enabled (VRA bit set in the AC97C_MR register). Automatically cleared by a processor read operation.

0: No data has been requested from the channel since the last read of the Status Register, or data has been available each time the CODEC requested new data from the channel since the last read of the Status Register.

1: Data has been emitted while no valid data to send has been previously loaded into the Channel Transmit Shift Register since the last read of the Status Register.

- **RXRDY: Channel Receive Ready**

0: Channel Receive Holding Register is empty.

1: Data has been received and loaded in Channel Receive Holding Register.

- **OVRUN: Receive Overrun**

Automatically cleared by a processor read operation.

0: No data has been loaded in the Channel Receive Holding Register while previous data has not been read since the last read of the Status Register.

1: Data has been loaded in the Channel Receive Holding Register while previous data has not yet been read since the last read of the Status Register.

36.7.10 AC97 Controller Codec Status Register

Register Name: AC97C_COSR

Address: 0xFFFA0048

Access Type: Read-only

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

- **TXRDY: Channel Transmit Ready**

0: Data has been loaded in Channel Transmit Register and is waiting to be loaded in the Channel Transmit Shift Register.

1: Channel Transmit Register is empty.

- **TXEMPTY: Channel Transmit Empty**

0: Data remains in the Channel Transmit Register or is currently transmitted from the Channel Transmit Shift Register.

1: Data in the Channel Transmit Register have been loaded in the Channel Transmit Shift Register and sent to the codec.

- **UNRUN: Transmit Underrun**

Active only when Variable Rate Mode is enabled (VRA bit set in the AC97C_MR register). Automatically cleared by a processor read operation.

0: No data has been requested from the channel since the last read of the Status Register, or data has been available each time the CODEC requested new data from the channel since the last read of the Status Register.

1: Data has been emitted while no valid data to send has been previously loaded into the Channel Transmit Shift Register since the last read of the Status Register.

- **RXRDY: Channel Receive Ready**

0: Channel Receive Holding Register is empty.

1: Data has been received and loaded in Channel Receive Holding Register.

- **OVRUN: Receive Overrun**

Automatically cleared by a processor read operation.

0: No data has been loaded in the Channel Receive Holding Register while previous data has not been read since the last read of the Status Register.

1: Data has been loaded in the Channel Receive Holding Register while previous data has not yet been read since the last read of the Status Register.

36.7.11 AC97 Controller Channel A Mode Register

Register Name:AC97C_CAMR

Address: 0xFFFA002C

Access Type: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	PDCEN	CEN	–	–	CEM	SIZE	
15	14	13	12	11	10	9	8
RXBUFF	ENDRX	–	–	TXBUFE	ENDTX	–	–
7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	UNRUN	TXEMPTY	TXRDY

- **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.

36.7.12 AC97 Controller Channel B Mode Register

Register Name:AC97C_CBMR

Address: 0xFFFA003C

Access Type: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	CEN	–	–	CEM	SIZE	
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	OVRUN	RXRDY	–	UNRUN	TXEMPTY	TXRDY

- **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 B Interrupt Enable**
- **TXBUFE: Transmit Buffer Empty for Channel B 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 B 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 B 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 B 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 B Endian Mode**

0: Transferring Data through Channel B is straight forward (Little-Endian).

1: Transferring Data through Channel B from/to a memory is performed with from/to Big-Endian format translation.

- **CEN: Channel B Enable**

0: Data transfer is disabled on Channel B.

1: Data transfer is enabled on Channel B.

36.7.13 AC97 Controller Codec Mode Register

Register Name: AC97C_COMR

Address: 0xFFFA004C

Access Type: Read-write

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

- **TXRDY: Channel Transmit Ready Interrupt Enable**
- **TXEMPTY: Channel Transmit Empty Interrupt Enable**
- **UNRUN: Transmit Underrun Interrupt Enable**
- **RXRDY: Channel Receive Ready Interrupt Enable**
- **OVRUN: Receive Overrun Interrupt Enable**

36.7.14 AC97 Controller Status Register

Register Name:AC97C_SR

Address: 0xFFFA0050

Access Type: Read-only

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

WKUP and SOF flags in AC97C_SR register are automatically cleared by a processor read operation.

- **SOF: Start Of Frame**

0: No Start of Frame has been detected since the last read of the Status Register.

1: At least one Start of frame has been detected since the last read of the Status Register.

- **WKUP: Wake Up detection**

0: No Wake-up has been detected.

1: At least one rising edge on SDATA_IN has been asynchronously detected. That means 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.

- **CBEVT: Channel B Event**

A channel B event occurs when AC97C_CBSR AND AC97C_CBMR is not 0. CBEVT flag is automatically cleared when the channel event condition is cleared.

0: No event on the channel B has been detected since the last read of the Status Register.

1: At least one event on the channel B is active.

36.7.15 AC97 Codec Controller Interrupt Enable Register

Register Name:AC97C_IER

Address: 0xFFFA0054

Access Type: Write-only

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

- **SOF: Start Of Frame**
- **WKUP: Wake Up**
- **COEVT: Codec Event**
- **CAEVT: Channel A Event**
- **CBEVT: Channel B Event**

0: No Effect.

1: Enables the corresponding interrupt.

36.7.16 AC97 Controller Interrupt Disable Register

Register Name:AC97C_IDR

Address: 0xFFFA0058

Access Type: Write-only

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

- **SOF: Start Of Frame**
- **WKUP: Wake Up**
- **COEVT: Codec Event**
- **CAEVT: Channel A Event**
- **CBEVT: Channel B Event**

0: No Effect.

1: Disables the corresponding interrupt.

36.7.17 AC97 Controller Interrupt Mask Register

Register Name:AC97C_IMR

Address: 0xFFFA005C

Access Type: Read-only

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

- **SOF: Start Of Frame**
- **WKUP: Wake Up**
- **COEVT: Codec Event**
- **CAEVT: Channel A Event**
- **CBEVT: Channel B Event**

0: The corresponding interrupt is disabled.

1: The corresponding interrupt is enabled.

37. Controller Area Network (CAN)

37.1 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.

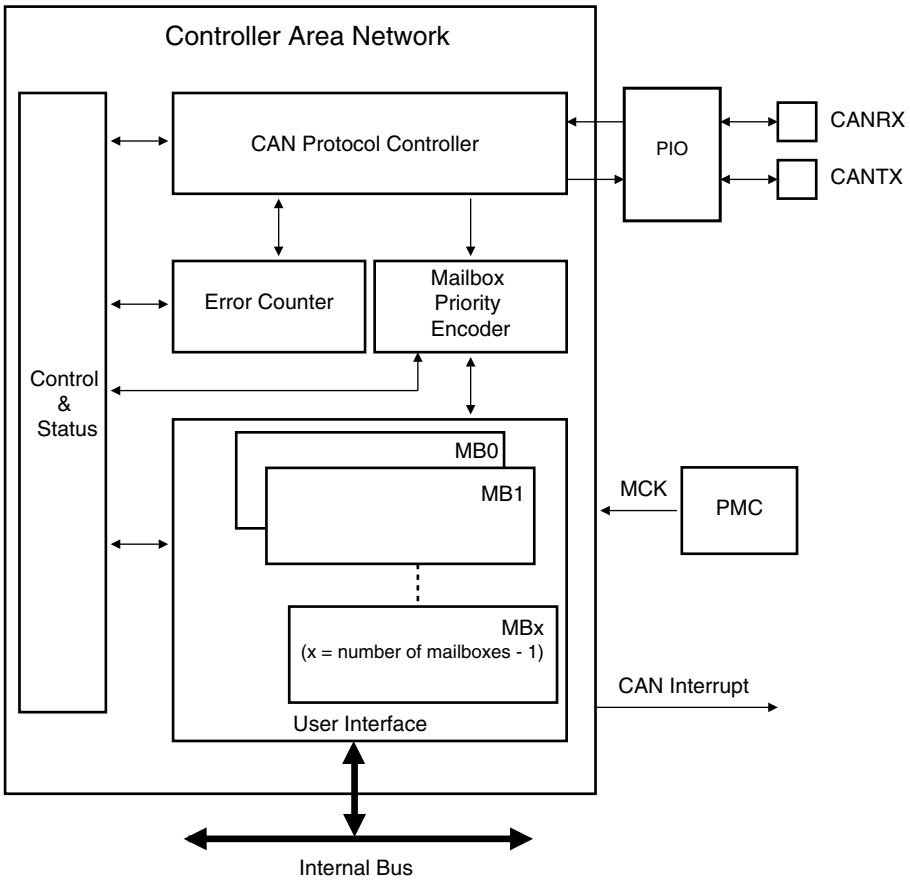
Any mailbox can be programmed for transmission. Several transmission mailboxes can be enabled in the same time. A priority can be defined for each mailbox independently.

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.

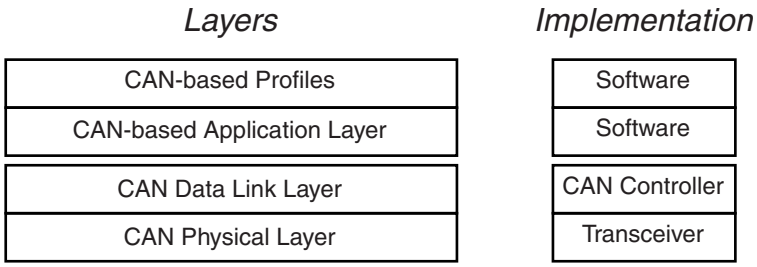
37.2 Block Diagram

Figure 37-1. CAN Block Diagram



37.3 Application Block Diagram

Figure 37-2. Application Block Diagram



37.4 I/O Lines Description

Table 37-1. I/O Lines Description

Name	Description	Type
CANRX	CAN Receive Serial Data	Input
CANTX	CAN Transmit Serial Data	Output

37.5 Product Dependencies

37.5.1 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.

37.5.2 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.

37.5.3 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.

37.6 CAN Controller Features

37.6.1 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.

37.6.2 Mailbox Organization

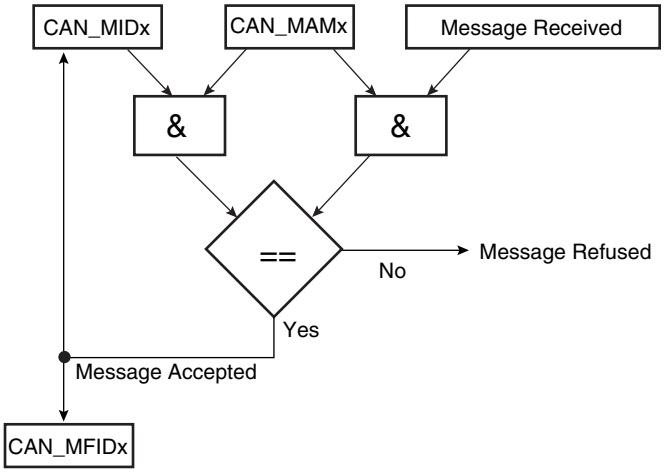
The CAN module has 16 buffers, also called channels or mailboxes. An identifier that corresponds to the CAN identifier is defined for each active mailbox. Message identifiers can match the standard frame identifier or the extended frame identifier. This identifier is defined for the first time during 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.

37.6.2.1 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 37-3. 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 10100010010000100100000100 0 11 00b
ID1 10100010010000100100000100 0 11 01b
ID2 10100010010000100100000100 0 11 10b
ID3 10100010010000100100000100 0 11 11b
ID4 10100010010000100100000100 1 11 00b
ID5 10100010010000100100000100 1 11 01b
ID6 10100010010000100100000100 1 11 10b
ID7 10100010010000100100000100 1 11 11b
```

The CAN_MIDx and CAN_MAMx of Mailbox x must be initialized to the corresponding values:

```
CAN_MIDx = 001 10100010010000100100000100 x 11 xxb
CAN_MAMx = 001 11111111111111111111111111 0 11 00b
```

If Mailbox x receives a message with ID6, then CAN_MIDx and CAN_MFIDx are set:

```
CAN_MIDx = 001 10100010010000100100000100 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]();
```

37.6.2.2 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.

37.6.2.3 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 37.6.3). 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.

37.6.3 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.

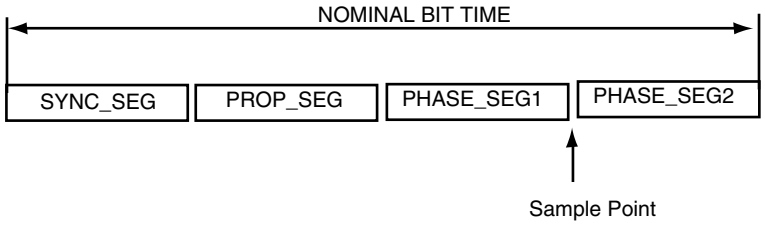
37.6.4 CAN 2.0 Standard Features

37.6.4.1 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 37-4. 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.

$$t_{PRS} = t_{CSC} \times (PROPAG + 1)$$

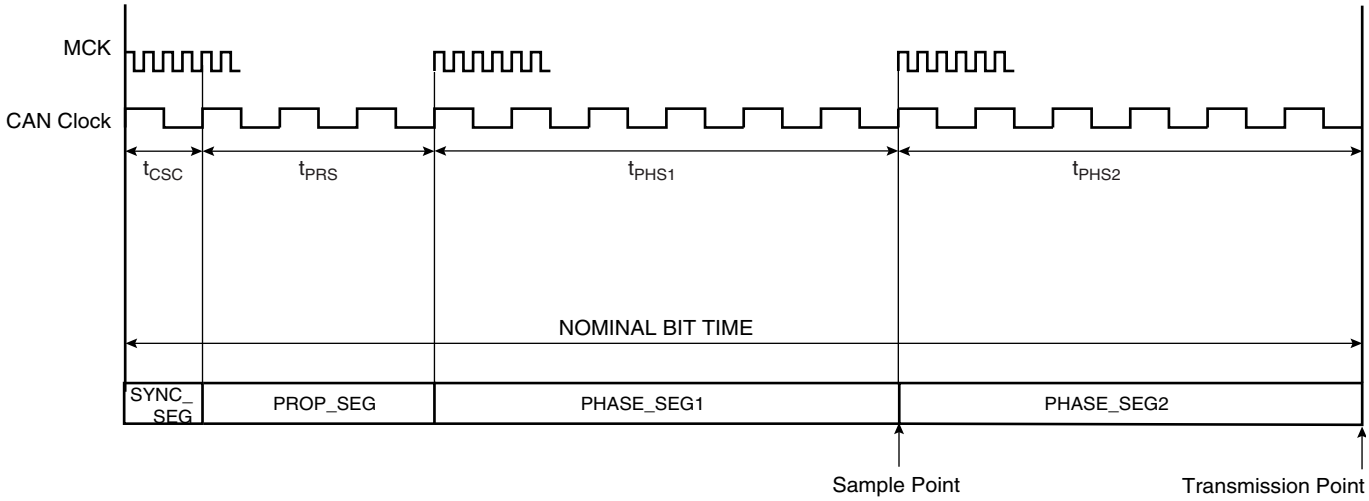
$$t_{PHS1} = t_{CSC} \times (PHASE1 + 1)$$

$$t_{PHS2} = t_{CSC} \times (PHASE2 + 1)$$

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 37-5. 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
```

37.6.4.2 CAN Bus Synchronization

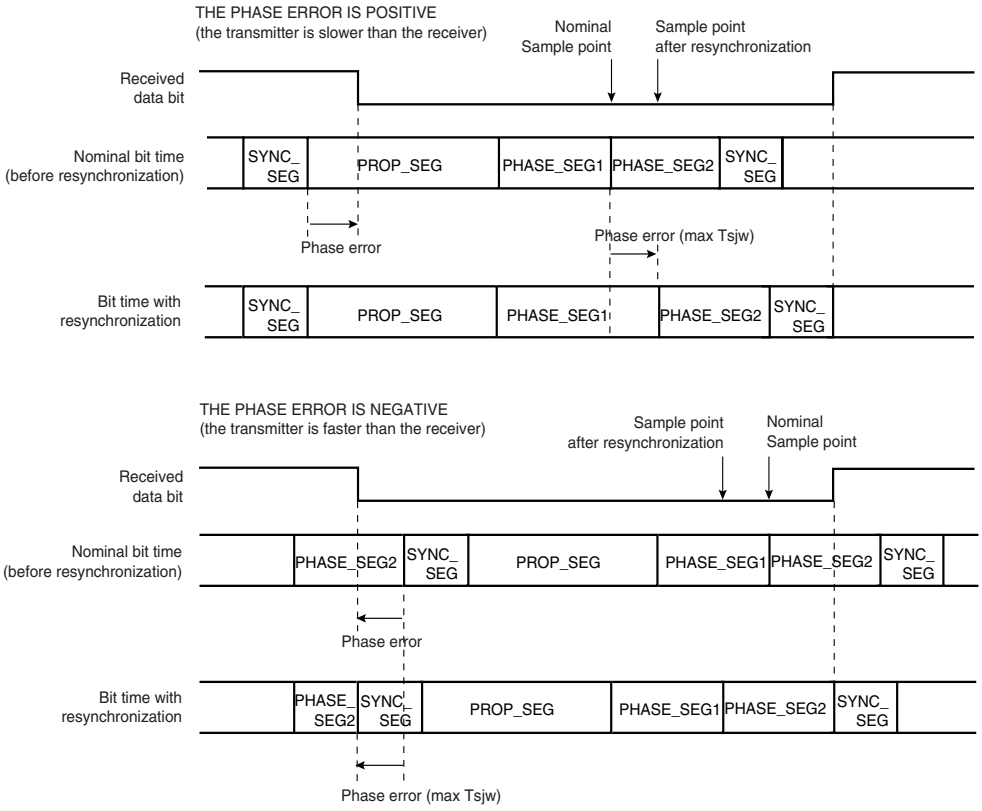
Two types of synchronization are distinguished: “hard synchronization” at the start of a frame and “resynchronization” inside a frame. After a hard synchronization, the bit time is restarted with the end of the SYNC_SEG segment, regardless of the phase error. Resynchronization causes a reduction or increase 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 37-6. CAN Resynchronization



37.6.4.3 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.

37.6.4.4 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):

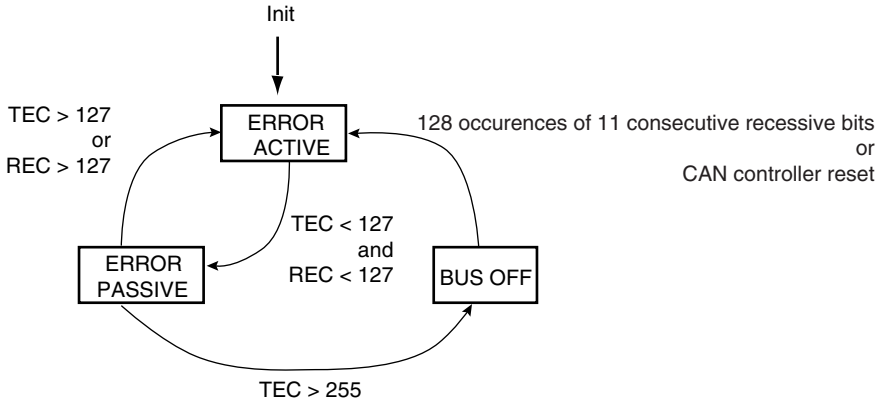
- CRC error (CERR bit in the CAN_SR register): With the CRC, the transmitter calculates a checksum for the CRC bit sequence from the Start of Frame bit until the end of the Data Field. This CRC sequence is transmitted in the CRC field of the Data or Remote Frame.
- Bit-stuffing error (SERR bit in the 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.

37.6.4.5 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 37-7. 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.

37.6.4.6 *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 37.6.4.5 “Fault Confinement” on page 660.

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.

37.6.4.7 *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.

37.6.5 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.

37.6.5.1 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.

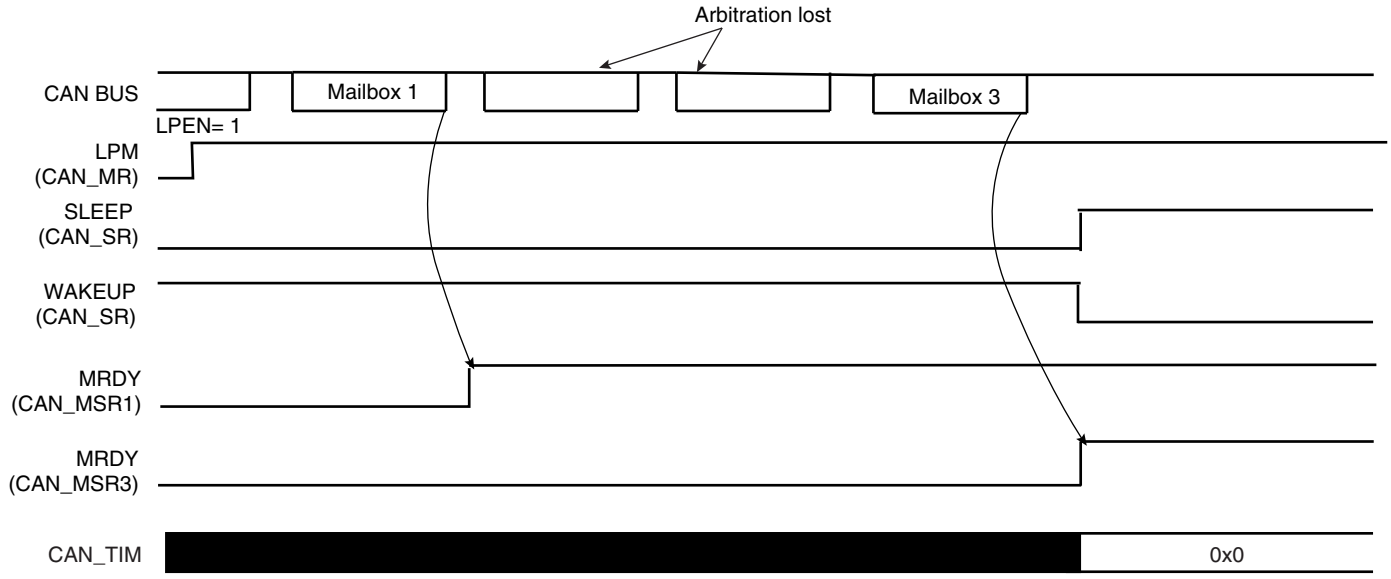
Thus, to enter Low-power Mode, the software application must:

– 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 37-8. Enabling Low-power Mode



37.6.5.2 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

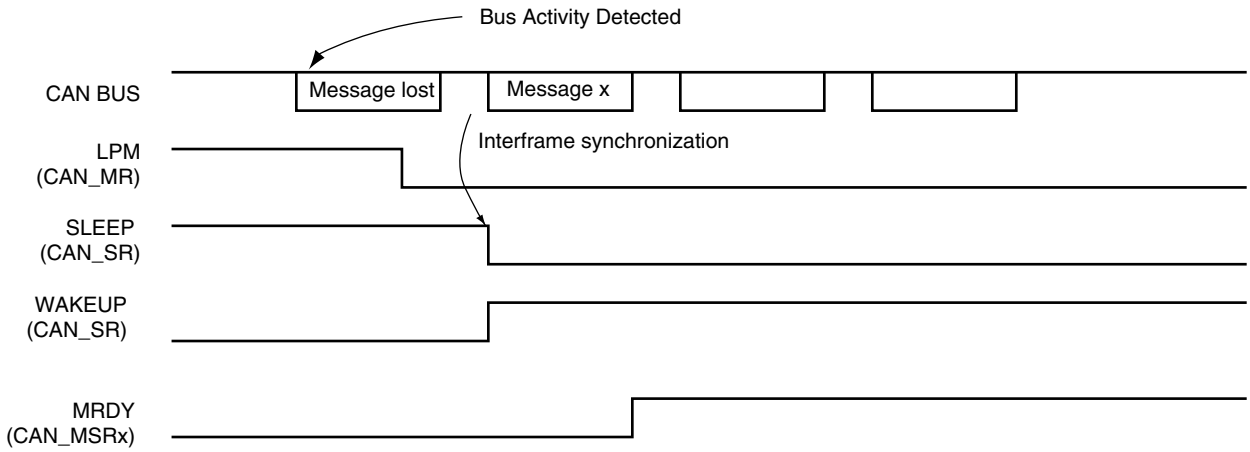
The CAN controller synchronizes itself with the bus activity by checking for eleven consecutive “recessive” bits. Once synchronized, the WAKEUP signal in the CAN_SR register is set.

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 37-9).

Figure 37-9. Disabling Low-power Mode



37.7 Functional Description

37.7.1 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).

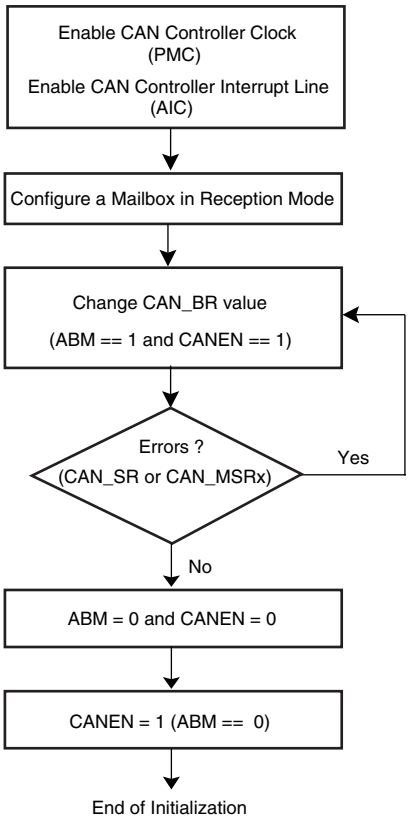
The CAN controller must be initialized with the CAN network parameters. The CAN_BR register defines the sampling point in the bit time period. CAN_BR must be set before the CAN controller is enabled by setting the CANEN field in the CAN_MR register.

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 37-10. Possible Initialization Procedure



37.7.2 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.

37.7.3 CAN Controller Message Handling

37.7.3.1 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.

37.7.3.2 Simple Receive Mailbox

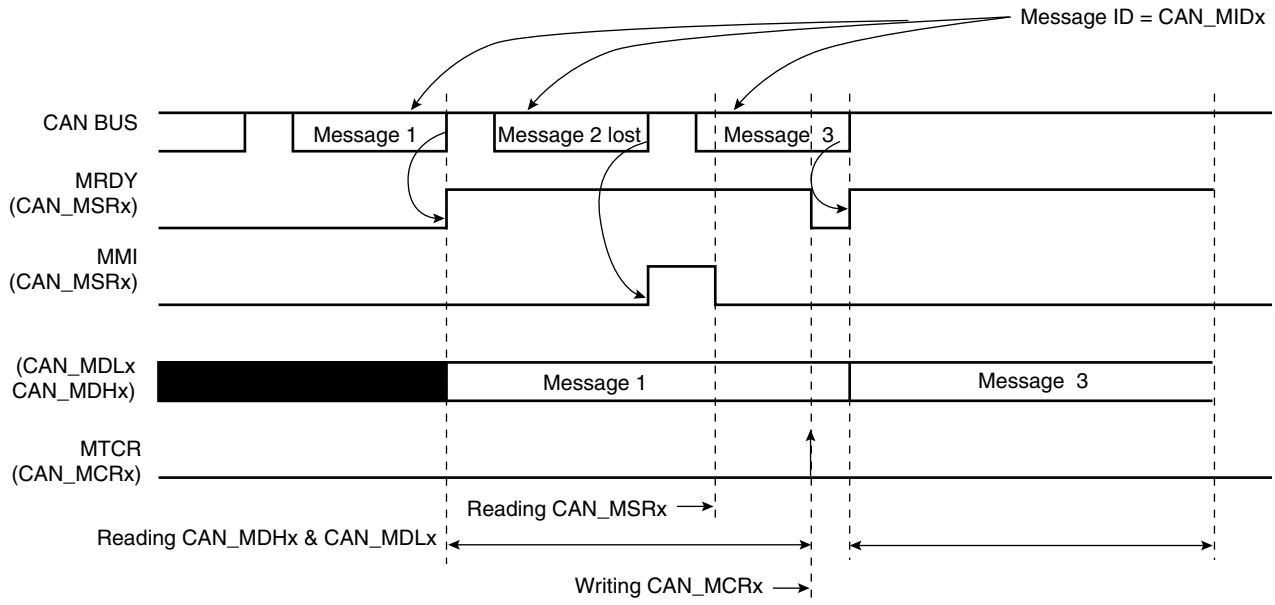
A mailbox is in Receive Mode once the MOT field in the CAN_MMRx register has been configured. Message ID and Message Acceptance Mask must be set before the Receive Mode is enabled.

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 37-11.

Figure 37-11. Receive Mailbox



Note: In the case of ARM architecture, CAN_MSRx, CAN_MDLx, CAN_MDHx can be read using an optimized ldm assembler instruction.

37.7.3.3 Receive with Overwrite Mailbox

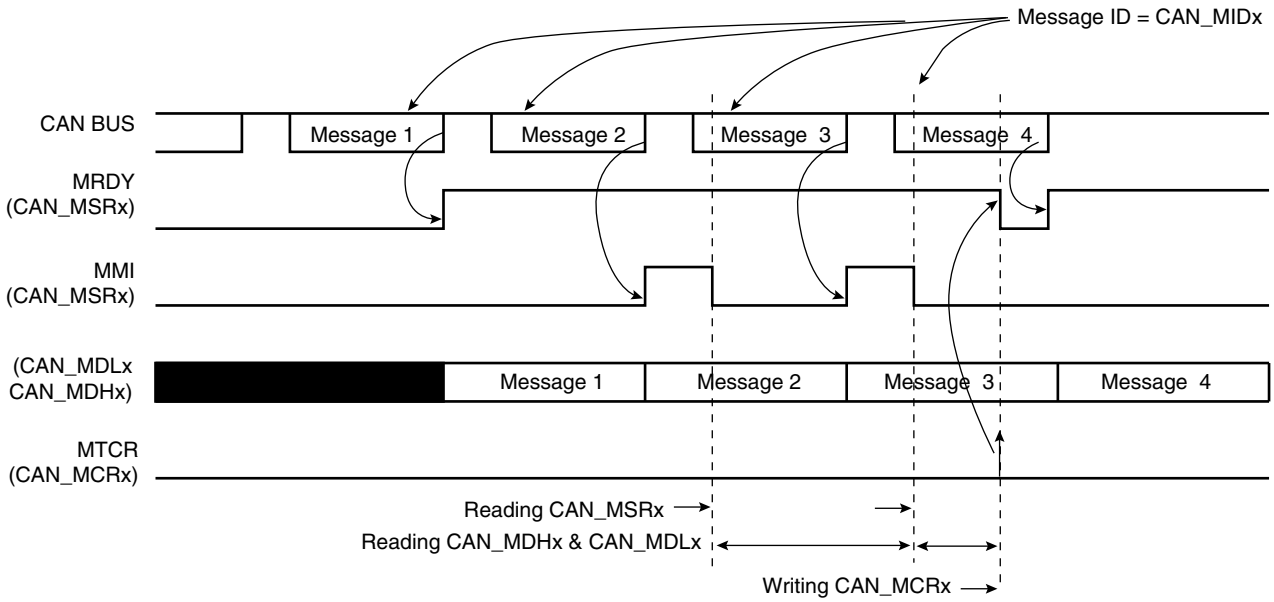
A mailbox is in Receive with Overwrite Mode once the MOT field in the CAN_MMRx register has been configured. Message ID and Message Acceptance masks must be set before Receive Mode is enabled.

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 37-12).

Figure 37-12. Receive with Overwrite Mailbox

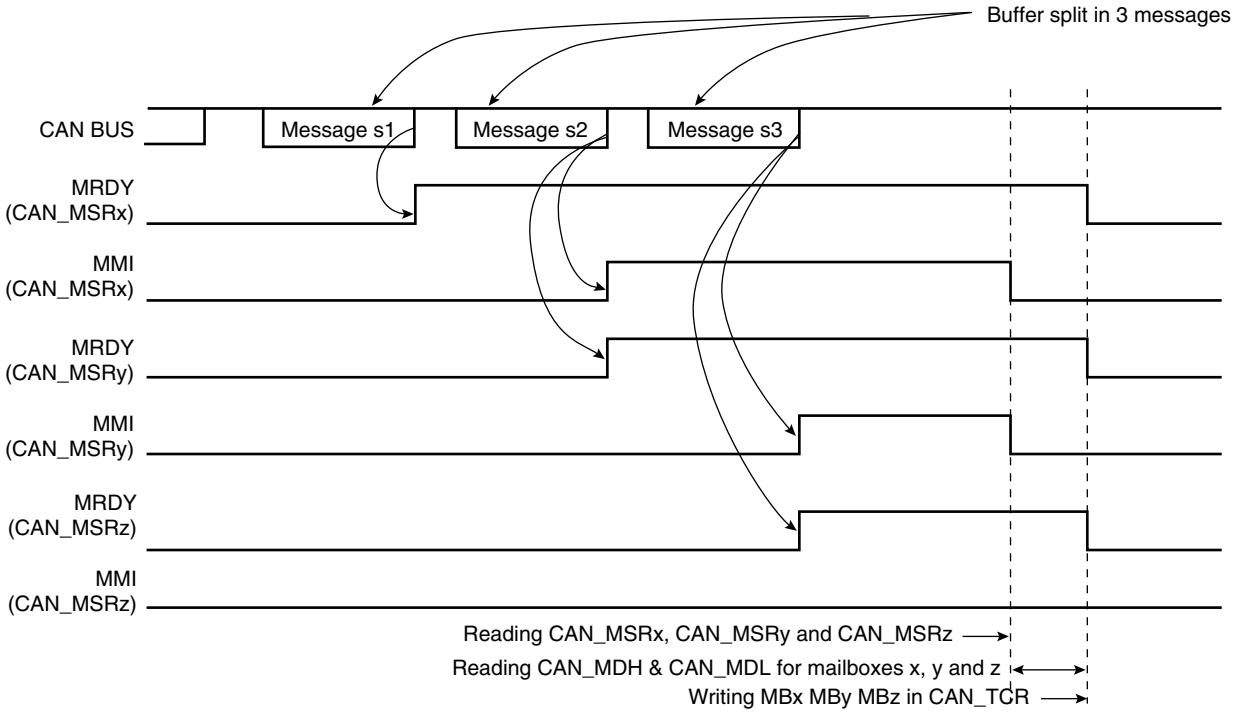


37.7.3.4 Chaining Mailboxes

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.

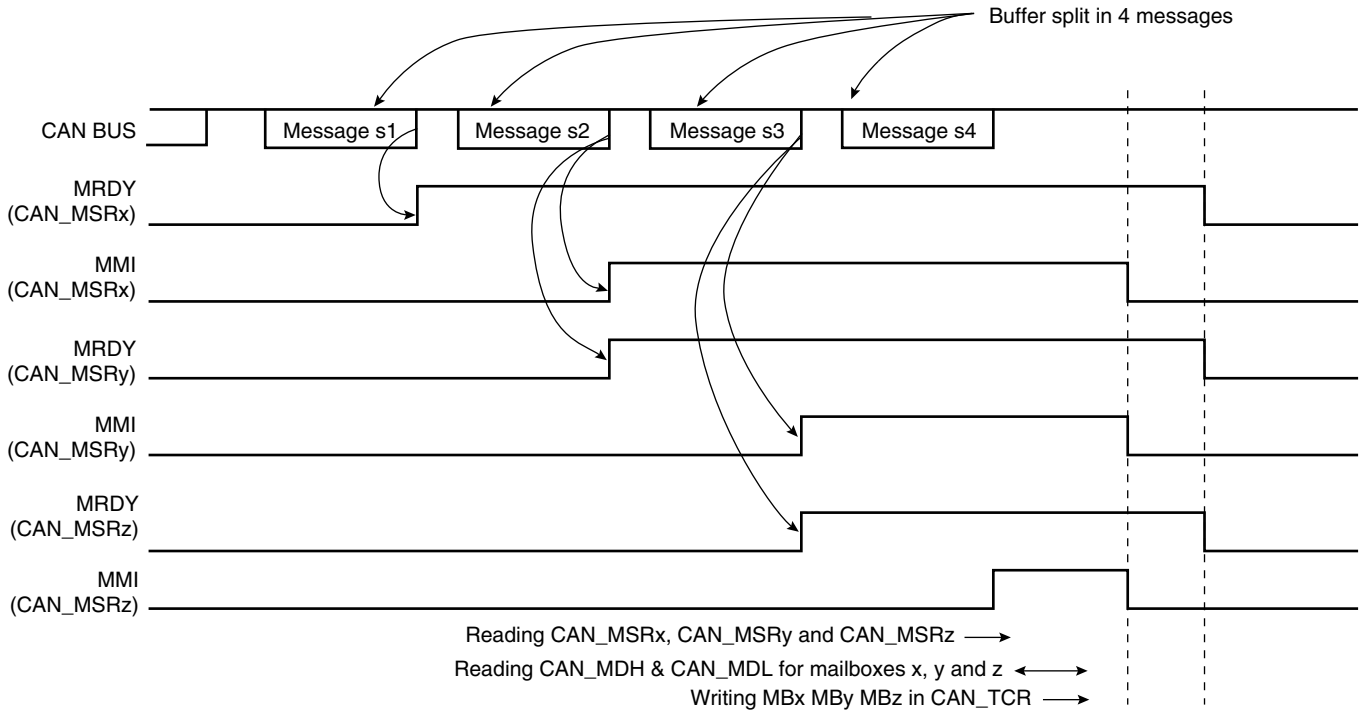
If several mailboxes are chained to receive a buffer split into several messages, all mailboxes except the last one (with the highest number) must be configured in Receive Mode. The first message received is handled by the first mailbox, the second one is refused by the first mailbox and accepted by the second mailbox, the last message is accepted by the last mailbox and refused by previous ones (see Figure 37-13).

Figure 37-13. 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 37-14).

Figure 37-14. Chaining Three Mailboxes to Receive a Buffer Split into Four Messages



37.7.3.5 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 671.

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.

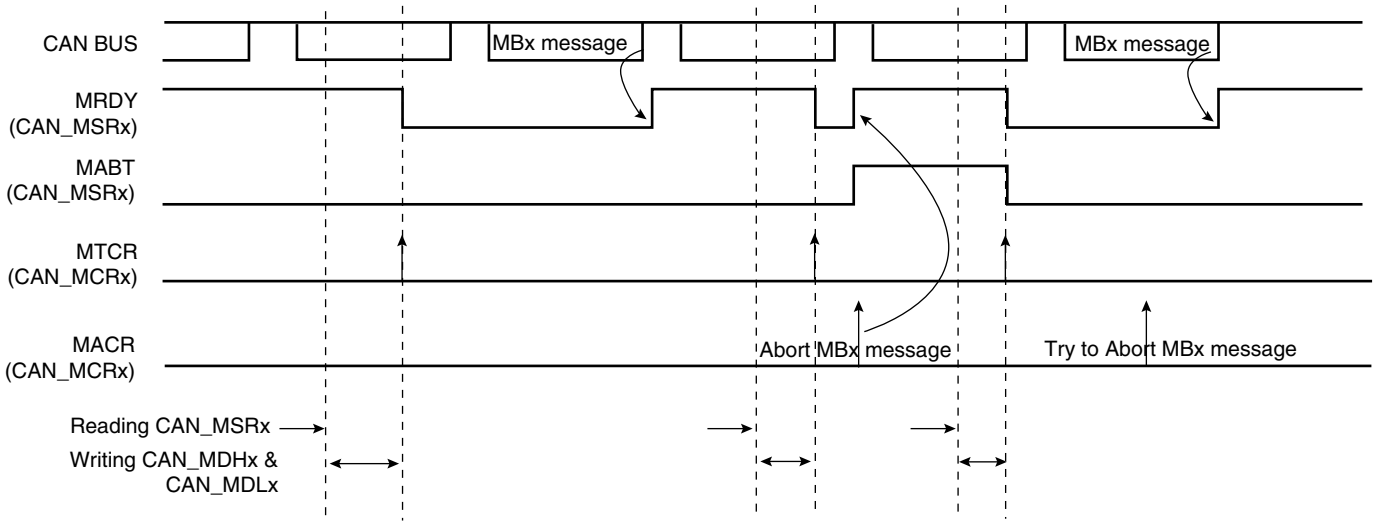
Setting the MACR bit in the CAN_MCRx register aborts the transmission. Transmission for several mailboxes can be aborted by writing MBx fields in the CAN_MACR register. If the message is being sent when the abort command is set, then the application is notified by the MRDY bit set and not the MABT in 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 37-15 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.

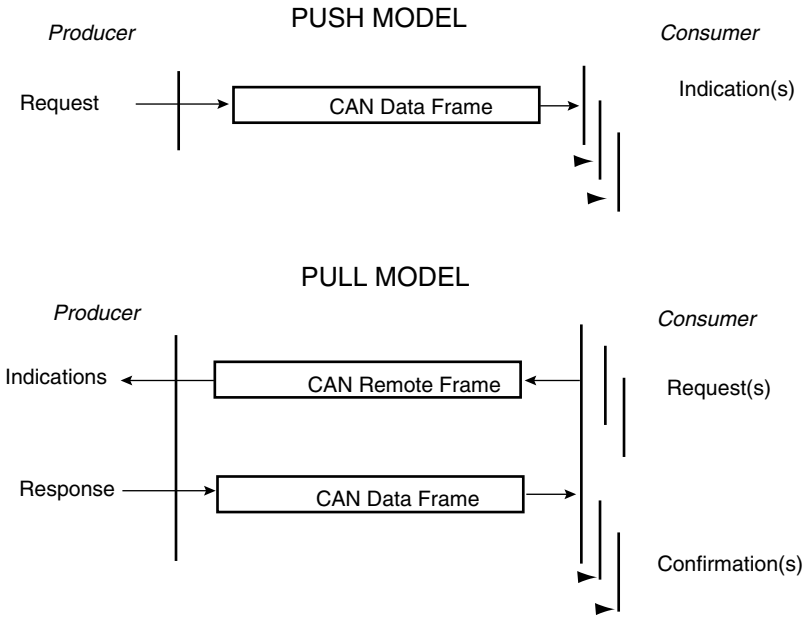
Figure 37-15. Transmitting Messages



37.7.3.6 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 37-16. 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.

37.7.3.7 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.

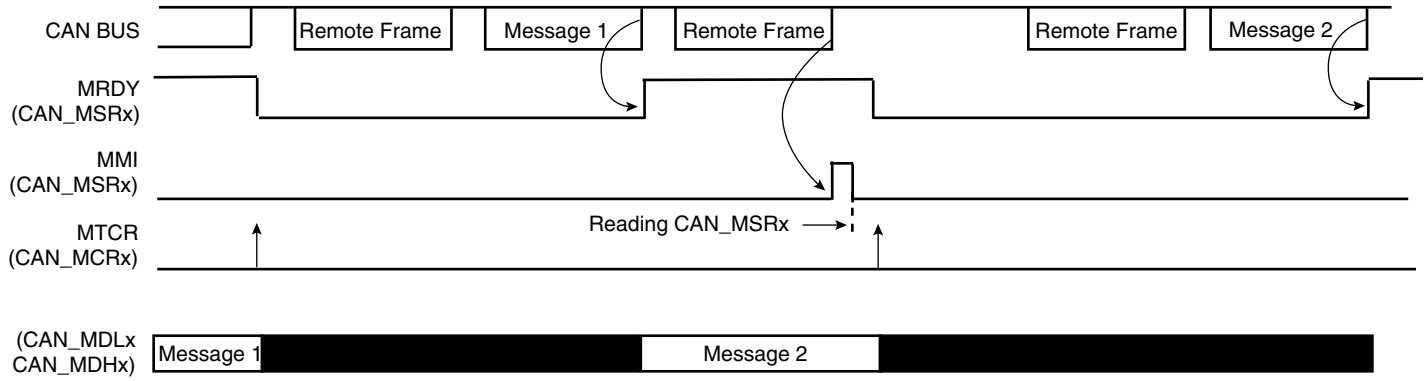
The MRDY flag remains at zero as long as the message has not been sent or aborted. No access to the mailbox data register can be done while MRDY flag is cleared. An interrupt is pending for the mailbox while the MRDY flag is set. This interrupt can be masked according to the mailbox flag in the CAN_IMR 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 670.

Figure 37-17. Producer Handling



37.7.3.8 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.

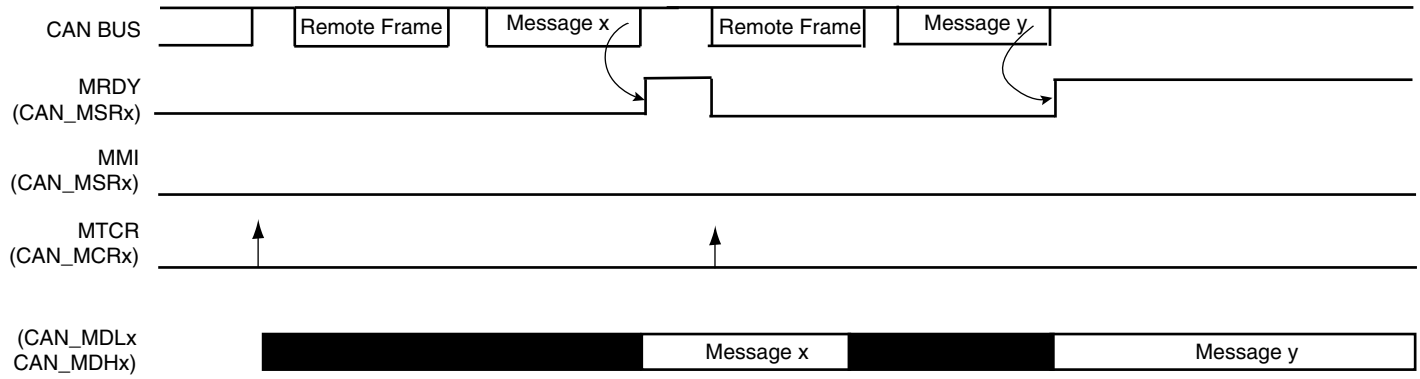
After Consumer Mode is enabled, the MRDY flag in the CAN_MSR register is automatically cleared until the first transfer request command. The software application sends a remote frame by setting the MTCR bit in the CAN_MCRx register or the MBx bit in the global CAN_TCR register. The application is notified 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 37-18. Consumer Handling



37.7.4 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:

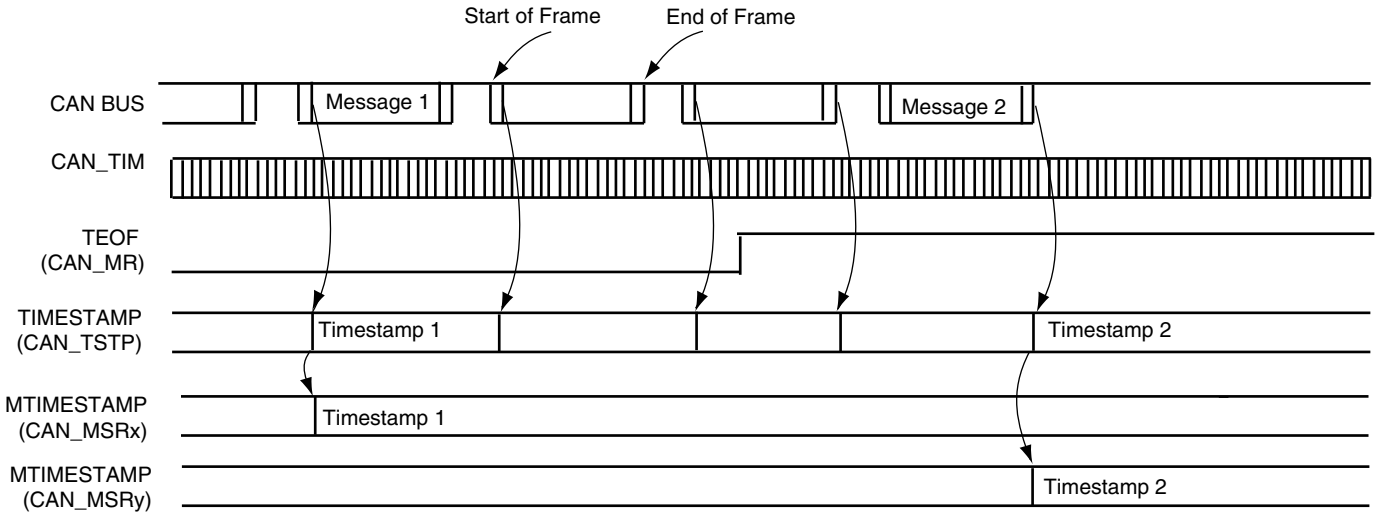
- Timestamping Mode: The value of the internal timer is captured at each Start Of Frame or each End Of Frame.
- Time Triggered Mode: The mailbox transfer operation is triggered when the internal timer reaches the mailbox trigger.

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.

37.7.4.1 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.

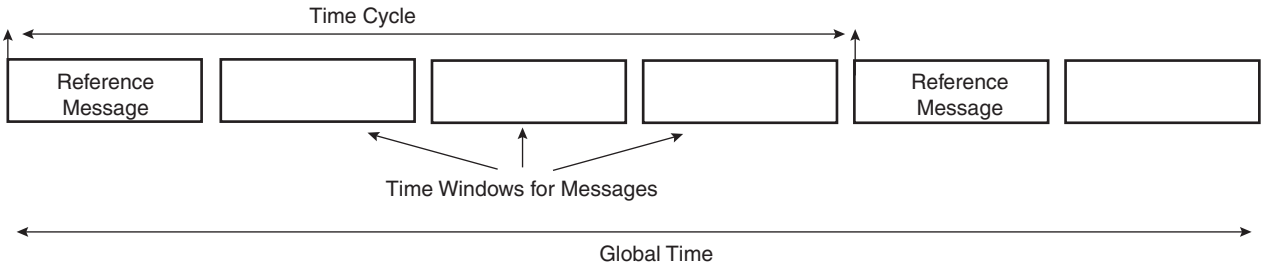
Figure 37-19. Mailbox Timestamp



37.7.4.2 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 37-20. 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.

37.7.4.3 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.

37.7.4.4 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.

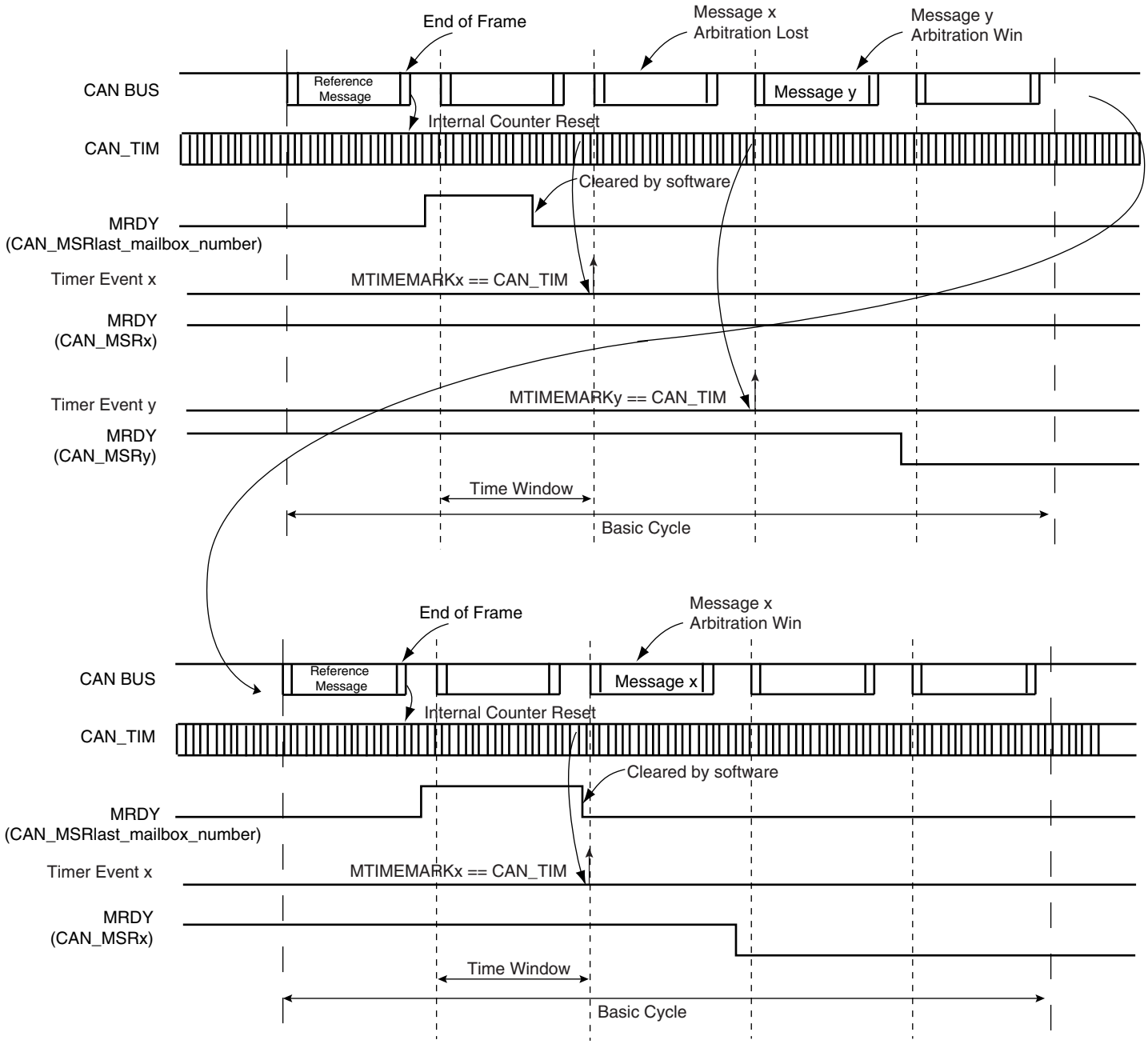
If the transmit operation is failed, i.e., the message loses the bus arbitration and the next transmit attempt is delayed until the next internal time trigger event. This prevents overlapping the next time window, but the message is still pending and is retried in the next time window when CAN_TIM value equals the MTIMEMARK value. It is also possible to prevent a retry by setting the DRPT field in the CAN_MR register.

37.7.4.5 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 37-21. Time Triggered Operations



37.8 Controller Area Network (CAN) User Interface

Table 37-2. 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.

37.8.1 CAN Mode Register

Name: CAN_MR

Address: 0xFFFAC000

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
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 675.

- **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.

37.8.2 CAN Interrupt Enable Register

Name: CAN_IER

Address: 0xFFFAC004

Access Type: 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.

37.8.3 CAN Interrupt Disable Register

Name: CAN_IDR

Address: 0xFFFAC008

Access Type: 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.

37.8.4 CAN Interrupt Mask Register

Name: CAN_IMR

Address: 0xFFFAC00C

Access Type: 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.

37.8.5 CAN Status Register

Name: CAN_SR

Address: 0xFFFAC010

Access Type: 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 37.6.4.6 “Error Interrupt Handler” on page 661 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 37.6.4.6 “Error Interrupt Handler” on page 661 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 37.6.4.6 “Error Interrupt Handler” on page 661 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 37.6.4.6 “Error Interrupt Handler” on page 661 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 bit 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.

37.8.6 CAN Baudrate Register

Name: CAN_BR

Address: 0xFFFAC014

Access Type: 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 37.6.4.1 “CAN Bit Timing Configuration” on page 656.

- **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.

37.8.7 CAN Timer Register

Name: CAN_TIM

Address: 0xFFFAC018

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
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.

37.8.8 CAN Timestamp Register

Name: CAN_TIMESTP

Address: 0xFFFAC01C

Access Type: Read-only

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

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

15	14	13	12	11	10	9	8
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.

37.8.9 CAN Error Counter Register

Name: CAN_ECR

Address: 0xFFFAC020

Access Type: 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.

37.8.10 CAN Transfer Command Register

Name: CAN_TCR

Address: 0xFFFAC024

Access Type: 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.

37.8.11 CAN Abort Command Register

Name: CAN_ACR

Address: 0xFFFAC028

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
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.

37.8.12 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 Type: 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 675.

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.

MOT			Mailbox Object Type
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

37.8.13 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 Type: 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.

37.8.14 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 Type: 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**

37.8.15 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 Type: 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
```

37.8.16 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 Type: 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.

37.8.17 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 Type: 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]

37.8.18 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 Type: 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]

37.8.19 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 Type: 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.

38. Pulse Width Modulation Controller (PWM)

38.1 Overview

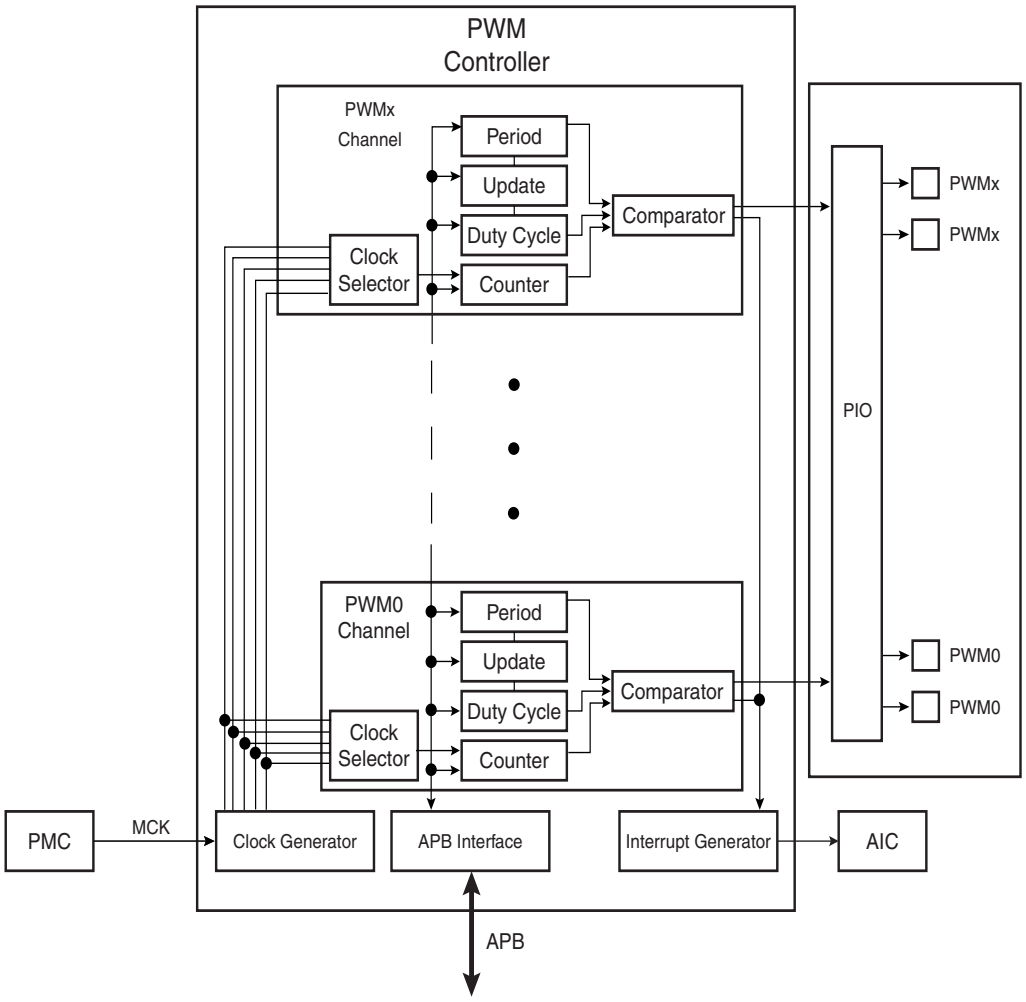
The PWM macrocell controls several channels independently. Each channel controls one square output waveform. Characteristics of the output waveform such as period, duty-cycle and polarity are configurable through the user interface. Each channel selects and uses one of the clocks provided by the clock 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.

38.2 Block Diagram

Figure 38-1. Pulse Width Modulation Controller Block Diagram



38.3 I/O Lines Description

Each channel outputs one waveform on one external I/O line.

Table 38-1. I/O Line Description

Name	Description	Type
PWMx	PWM Waveform Output for channel x	Output

38.4 Product Dependencies

38.4.1 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.

38.4.2 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.

38.4.3 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.

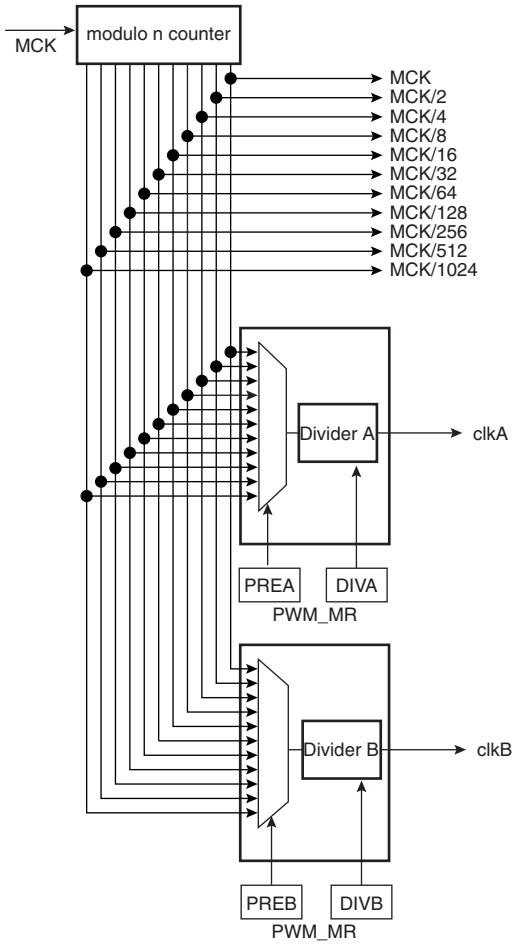
38.5 Functional Description

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.

38.5.1 PWM Clock Generator

Figure 38-2. 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).

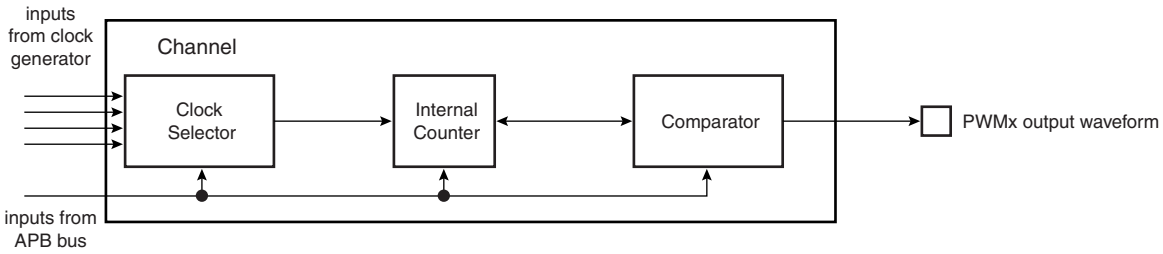
After a reset of the PWM controller, DIVA (DIVB) and PREA (PREB) in the PWM Mode register are set to 0. This implies that after reset clkA (clkB) are turned off.

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.

38.5.2 PWM Channel

38.5.2.1 Block Diagram

Figure 38-3. 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 38.5.1 “PWM Clock Generator” on page 711.
- An internal counter clocked by the output of the clock selector. This internal counter is incremented or decremented according to the channel configuration and comparators events. The size of the internal counter is 16 bits.
- A comparator used to generate events according to the internal counter value. It also computes the PWMx output waveform according to the configuration.

38.5.2.2 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:

$$\frac{(CRPD \times DIVA)}{MCK} \text{ or } \frac{(CRPD \times DIVAB)}{MCK}$$

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:

$$\frac{(2 \times CPRD \times DIVA)}{MCK} \text{ or } \frac{(2 \times CPRD \times DIVB)}{MCK}$$

- 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:

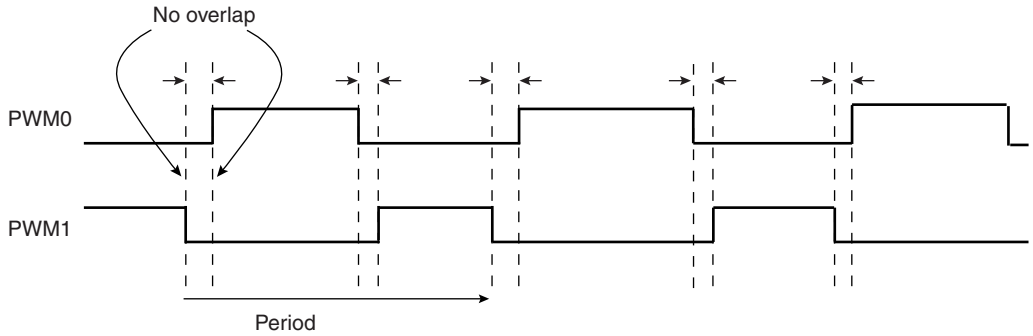
$$\text{duty cycle} = \frac{(period - 1 / \text{ fchannel_x_clock} \times CDTY)}{period}$$

  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 38-4. Non Overlapped Center Aligned Waveforms



Note: 1. See Figure 38-5 on page 715 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:

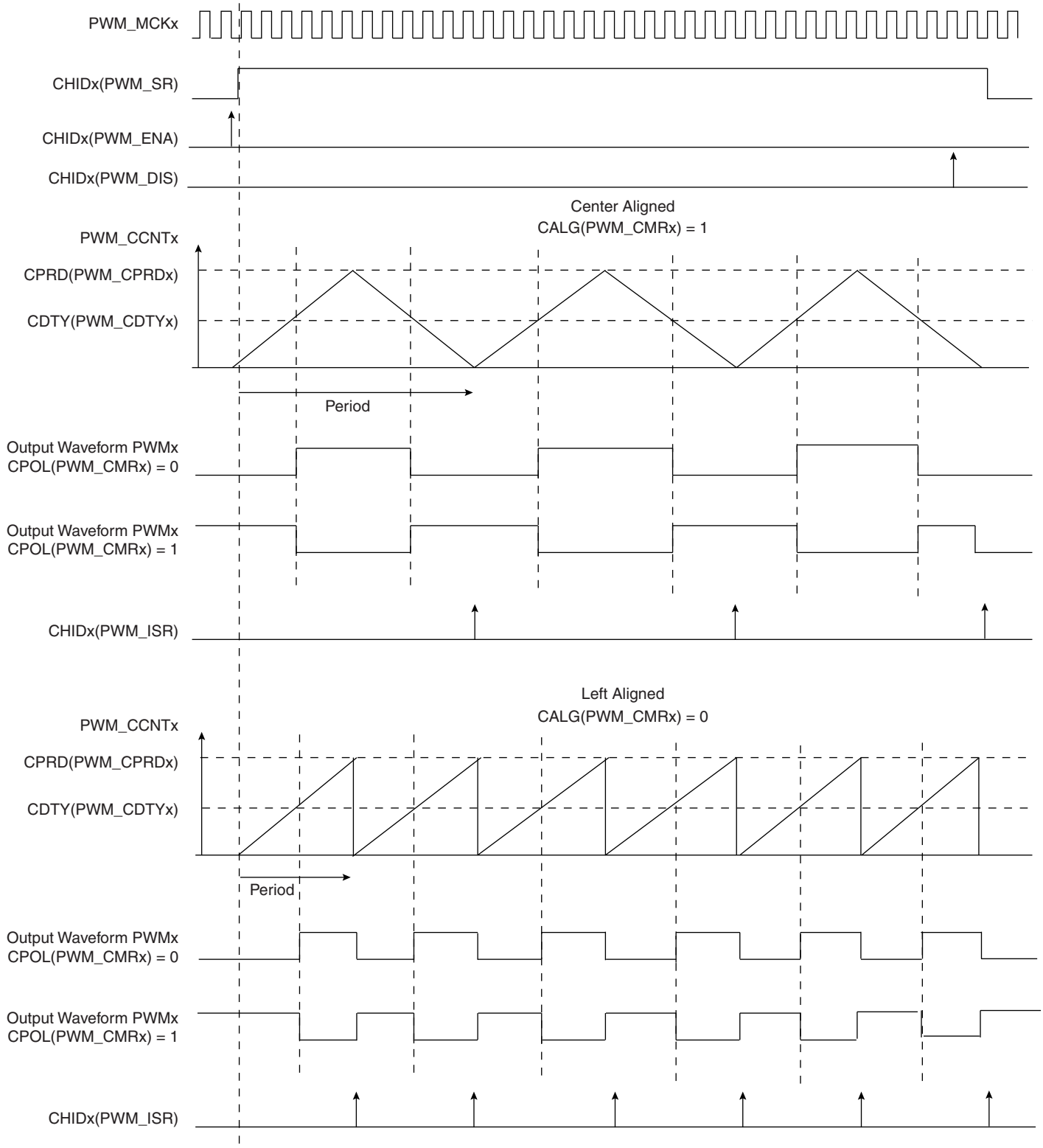
- CDTY = CPRD and CPOL = 0
- CDTY = 0 and CPOL = 1

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 38-5. Waveform Properties



38.5.3 PWM Controller Operations

38.5.3.1 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.

38.5.3.2 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.

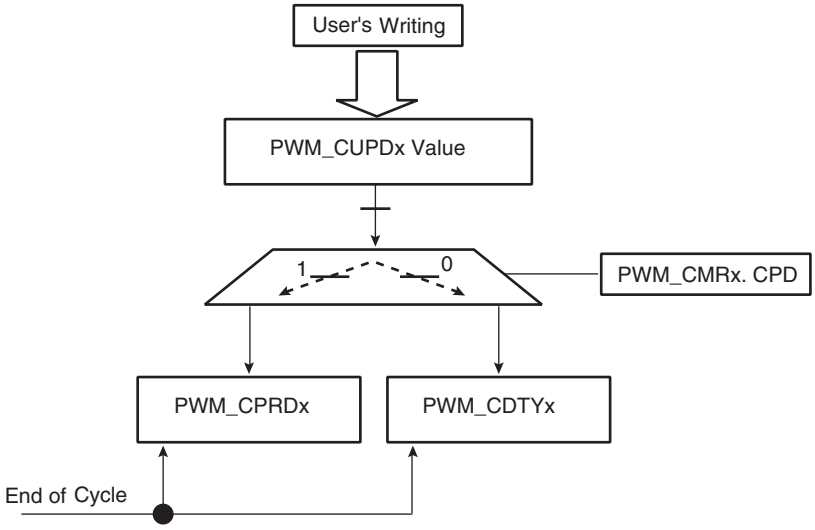
For example, if the user sets 15 (in decimal) in PWM_CPRDx, the user is able to set a value between 1 up to 14 in PWM_CDTYx Register. The resulting duty cycle quantum cannot be lower than 1/15 of the PWM period.

38.5.3.3 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 38-6. Synchronized Period or Duty Cycle Update



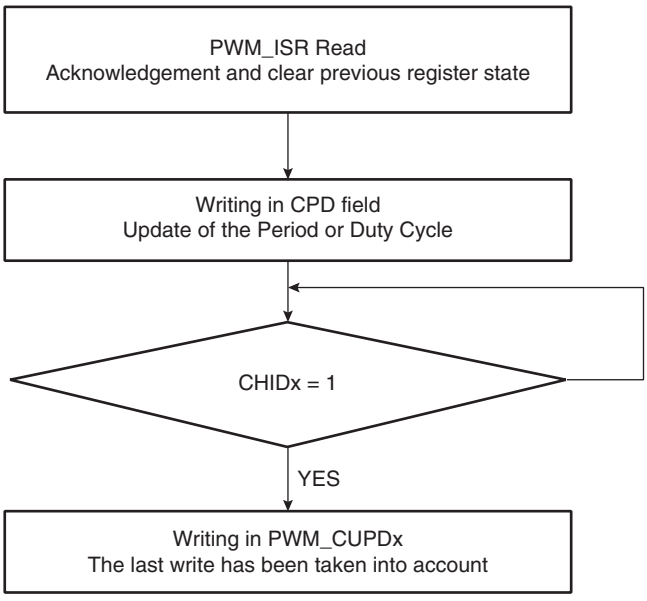
To prevent overwriting the PWM_CUPDx by software, the user can use status events in order to synchronize his software. Two methods are possible. In both, the user must enable the dedicated interrupt in PWM_IER at PWM Controller level.

The first method (polling method) consists of reading the relevant status bit in PWM_ISR Register according to the enabled channel(s). See Figure 38-7.

The second method uses an Interrupt Service Routine associated with the PWM channel.

Note: Reading the PWM_ISR register automatically clears CHIDx flags.

Figure 38-7. Polling Method



Note: Polarity and alignment can be modified only when the channel is disabled.

38.5.3.4 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.

38.6 Pulse Width Modulation Controller (PWM) User Interface

Table 38-2. Register Mapping

Offset[1]	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
0x4C - 0xFC	Reserved	–	–	–
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	-

Note: 1. Some registers are indexed with “ch_num” index ranging from 0 to 3.

38.6.1 PWM Mode Register

Register Name: PWM_MR

Address: 0xFFFB8000

Access Type: Read/Write

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

23	22	21	20	19	18	17	16
DIVB							

15	14	13	12	11	10	9	8
–	–	–	–	PREA			

7	6	5	4	3	2	1	0
DIVA							

- **DIVA, DIVB: CLKA, CLKB Divide Factor**

DIVA, DIVB	CLKA, CLKB
0	CLKA, CLKB clock is turned off
1	CLKA, CLKB clock is clock selected by PREA, PREB
2-255	CLKA, CLKB clock is clock selected by PREA, PREB divided by DIVA, DIVB factor.

- **PREA, PREB**

PREA, PREB				Divider Input Clock
0	0	0	0	MCK.
0	0	0	1	MCK/2
0	0	1	0	MCK/4
0	0	1	1	MCK/8
0	1	0	0	MCK/16
0	1	0	1	MCK/32
0	1	1	0	MCK/64
0	1	1	1	MCK/128
1	0	0	0	MCK/256
1	0	0	1	MCK/512
1	0	1	0	MCK/1024
Other				Reserved

38.6.2 PWM Enable Register

Register Name: PWM_ENA

Address: 0xFFFB8004

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = No effect.

1 = Enable PWM output for channel x.

38.6.3 PWM Disable Register

Register Name: PWM_DIS

Address: 0xFFFB8008

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = No effect.

1 = Disable PWM output for channel x.

38.6.4 PWM Status Register

Register Name: PWM_SR

Address: 0xFFFB800C

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = PWM output for channel x is disabled.

1 = PWM output for channel x is enabled.

38.6.5 PWM Interrupt Enable Register

Register Name: PWM_IER

Address: 0xFFFB8010

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID.**

0 = No effect.

1 = Enable interrupt for PWM channel x.

38.6.6 PWM Interrupt Disable Register

Register Name: PWM_IDR

Address: 0xFFFB8014

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID.**

0 = No effect.

1 = Disable interrupt for PWM channel x.

38.6.7 PWM Interrupt Mask Register

Register Name: PWM_IMR

Address: 0xFFFB8018

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID.**

0 = Interrupt for PWM channel x is disabled.

1 = Interrupt for PWM channel x is enabled.

38.6.8 PWM Interrupt Status Register

Register Name: PWM_ISR

Address: 0xFFFB801C

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: Channel ID**

0 = No new channel period has been achieved since the last read of the PWM_ISR register.

1 = At least one new channel period has been achieved since the last read of the PWM_ISR register.

Note: Reading PWM_ISR automatically clears CHIDx flags.

38.6.9 PWM Channel Mode Register

Register Name: PWM_CMR[0..3]

Addresses: 0xFFFB8200 [0], 0xFFFB8220 [1], 0xFFFB8240 [2], 0xFFFB8260 [3]

Access Type: Read/Write

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

- **CPRE: Channel Pre-scaler**

CPRE				Channel Pre-scaler
0	0	0	0	MCK
0	0	0	1	MCK/2
0	0	1	0	MCK/4
0	0	1	1	MCK/8
0	1	0	0	MCK/16
0	1	0	1	MCK/32
0	1	1	0	MCK/64
0	1	1	1	MCK/128
1	0	0	0	MCK/256
1	0	0	1	MCK/512
1	0	1	0	MCK/1024
1	0	1	1	CLKA
1	1	0	0	CLKB
Other				Reserved

- **CALG: Channel Alignment**

0 = The period is left aligned.

1 = The period is center aligned.

- **CPOL: Channel Polarity**

0 = The output waveform starts at a low level.

1 = The output waveform starts at a high level.

- **CPD: Channel Update Period**

0 = Writing to the PWM_CUPDx will modify the duty cycle at the next period start event.

1 = Writing to the PWM_CUPDx will modify the period at the next period start event.

38.6.10 PWM Channel Duty Cycle Register

Register Name:PWM_CDTY[0..3]

Addresses: 0xFFFB8204 [0], 0xFFFB8224 [1], 0xFFFB8244 [2], 0xFFFB8264 [3]

Access Type: Read/Write

31	30	29	28	27	26	25	24
CDTY							

23	22	21	20	19	18	17	16
CDTY							

15	14	13	12	11	10	9	8
CDTY							

7	6	5	4	3	2	1	0
CDTY							

Only the first 16 bits (internal channel counter size) are significant.

- **CDTY: Channel Duty Cycle**

Defines the waveform duty cycle. This value must be defined between 0 and CPRD (PWM_CPRx).

38.6.11 PWM Channel Period Register

Register Name: PWM_CPRD[0..3]

Addresses: 0xFFFB8208 [0], 0xFFFB8228 [1], 0xFFFB8248 [2], 0xFFFB8268 [3]

Access Type: Read/Write

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

Only the first 16 bits (internal channel counter size) are significant.

- **CPRD: Channel Period**

If the waveform is left-aligned, then the output waveform period depends on the counter source clock and can be calculated:

– By using the Master Clock (MCK) divided by an X given prescaler value (with X being 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024). The resulting period formula will be:

$$\frac{(X \times CPRD)}{MCK}$$

– By using a Master Clock divided by one of both DIVA or DIVB divider, the formula becomes, respectively:

$$\frac{(CRPD \times DIVA)}{MCK} \text{ or } \frac{(CRPD \times DIVAB)}{MCK}$$

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:

$$\frac{(2 \times CPRD \times DIVA)}{MCK} \text{ or } \frac{(2 \times CPRD \times DIVB)}{MCK}$$

38.6.12 PWM Channel Counter Register

Register Name: PWM_CCNT[0..3]

Addresses: 0xFFFB820C [0], 0xFFFB822C [1], 0xFFFB824C [2], 0xFFFB826C [3]

Access Type: Read-only

31	30	29	28	27	26	25	24
CNT							

23	22	21	20	19	18	17	16
CNT							

15	14	13	12	11	10	9	8
CNT							

7	6	5	4	3	2	1	0
CNT							

- **CNT: Channel Counter Register**

Internal counter value. This register is reset when:

- the channel is enabled (writing CHIDx in the PWM_ENA register).
- the counter reaches CPRD value defined in the PWM_CPRDx register if the waveform is left aligned.

38.6.13 PWM Channel Update Register

Register Name: PWM_CUPD[0..3]

Addresses: 0xFFFB8210 [0], 0xFFFB8230 [1], 0xFFFB8250 [2], 0xFFFB8270 [3]

Access Type: Write-only

31	30	29	28	27	26	25	24
CUPD							

23	22	21	20	19	18	17	16
CUPD							

15	14	13	12	11	10	9	8
CUPD							

7	6	5	4	3	2	1	0
CUPD							

This register acts as a double buffer for the period or the duty cycle. This prevents an unexpected waveform when modifying the waveform period or duty-cycle.

Only the first 16 bits (internal channel counter size) are significant.

CPD (PWM_CMRx Register)	
0	The duty-cycle (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.

39. Timer Counter

39.1 Overview

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	SLCK

39.2 Block Diagram

Figure 39-1. Timer Counter Block Diagram

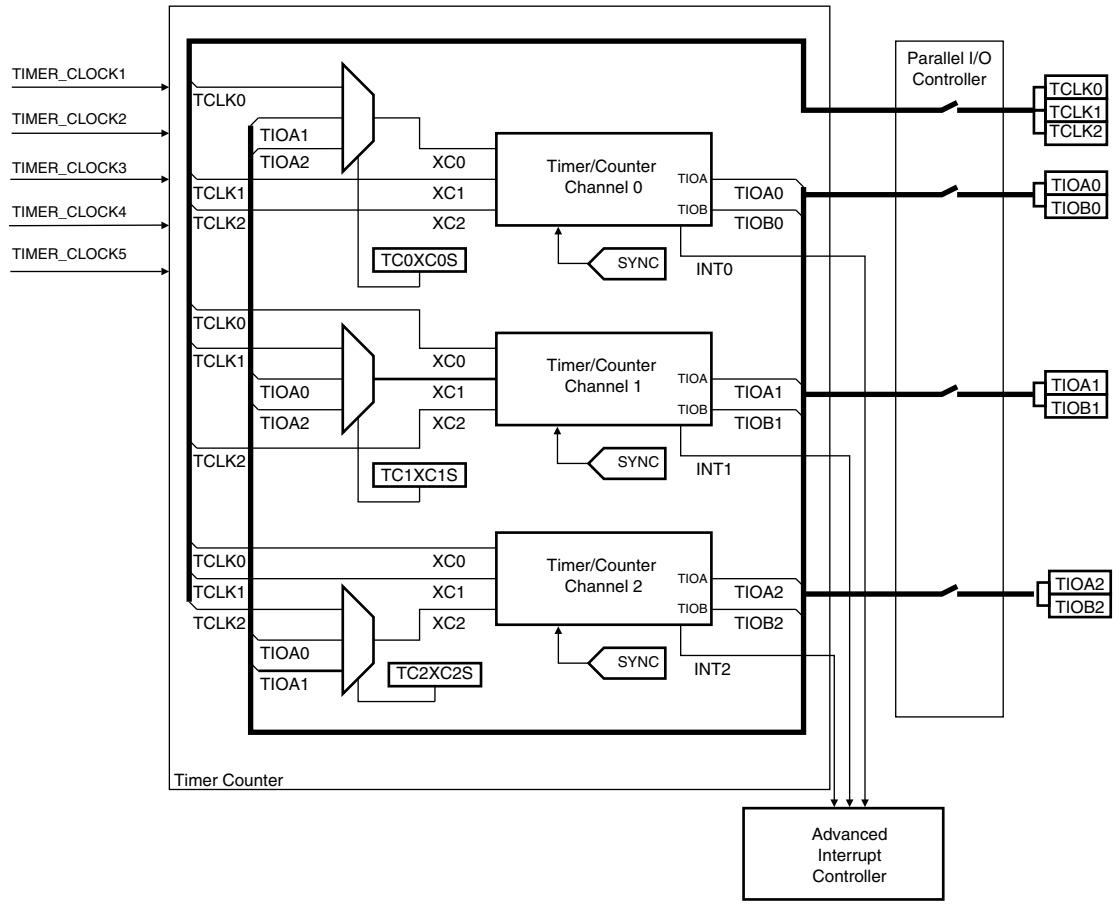


Table 39-2. Signal Name Description

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.

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 39-4 on page 749.

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 737.

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 737

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

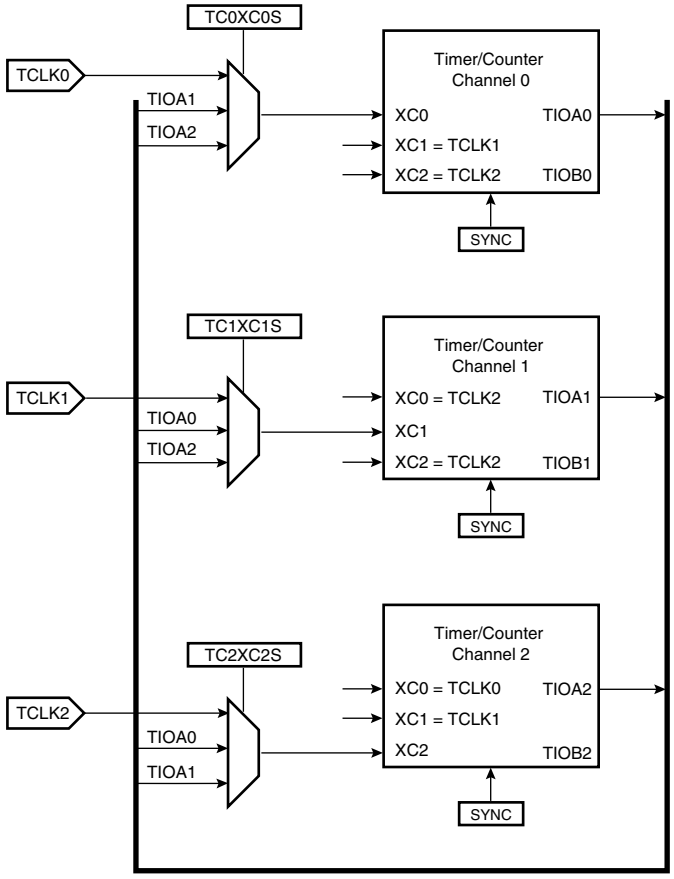
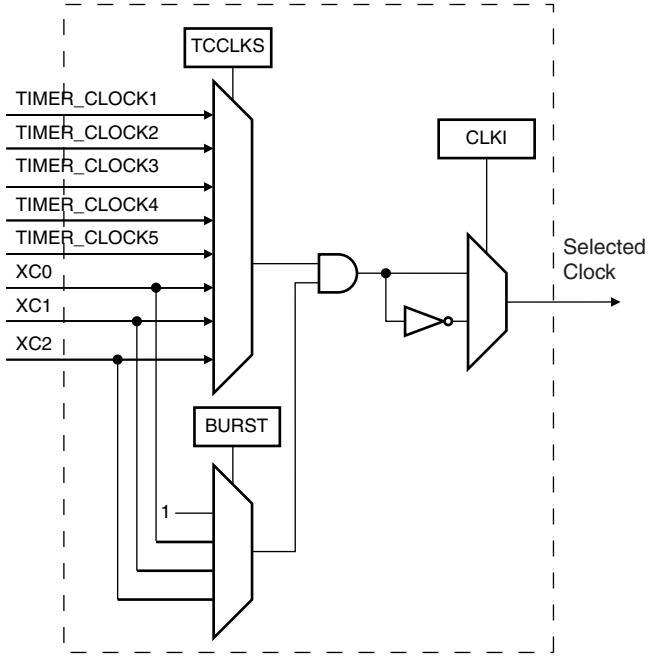


Figure 39-3. Clock Selection

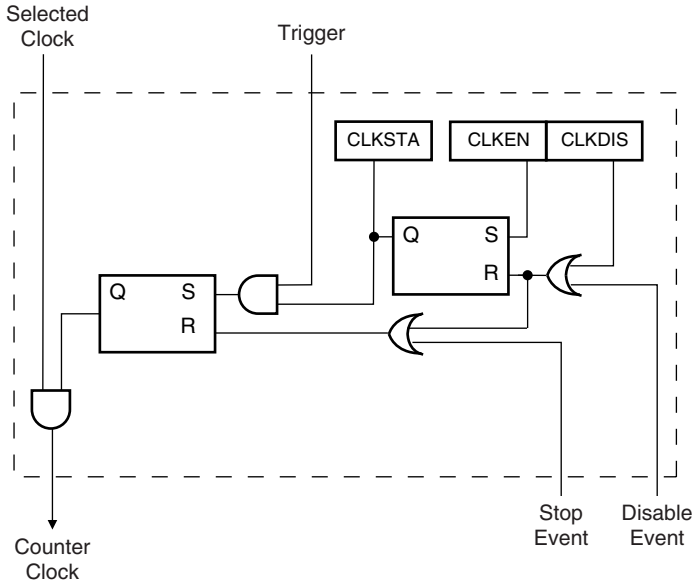


39.5.4 Clock Control

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.

The following triggers are common to both modes:

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

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

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

Regardless of the trigger used, it will be taken into account at the following active edge of the selected clock. This means that the counter value can be read differently from zero just after a trigger, especially when a low frequency signal is selected as the clock.

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.

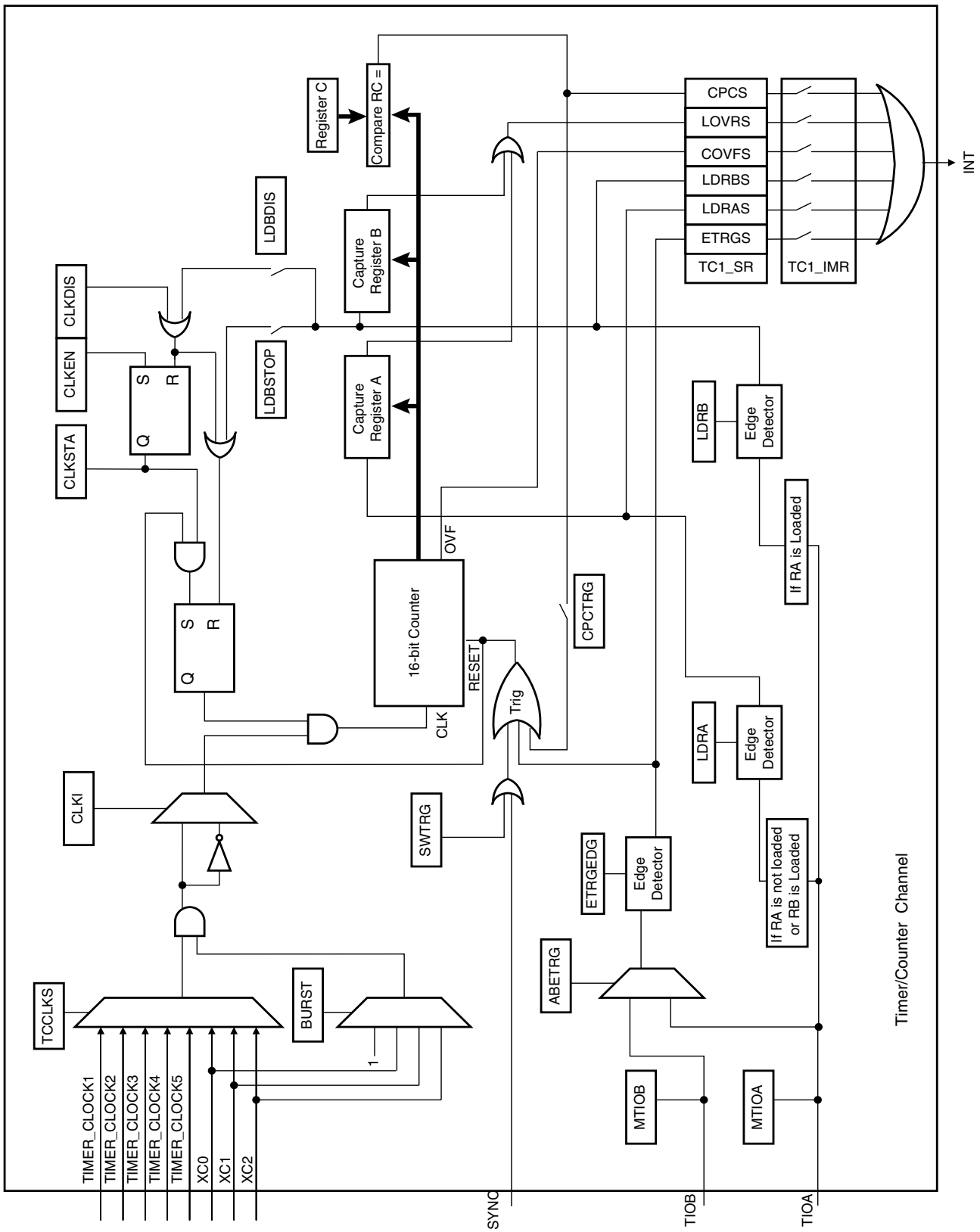
Loading RA or RB before the read of the last value loaded sets the Overrun Error Flag (LOVRS) in TC_SR (Status Register). In this case, the old value is overwritten.

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.

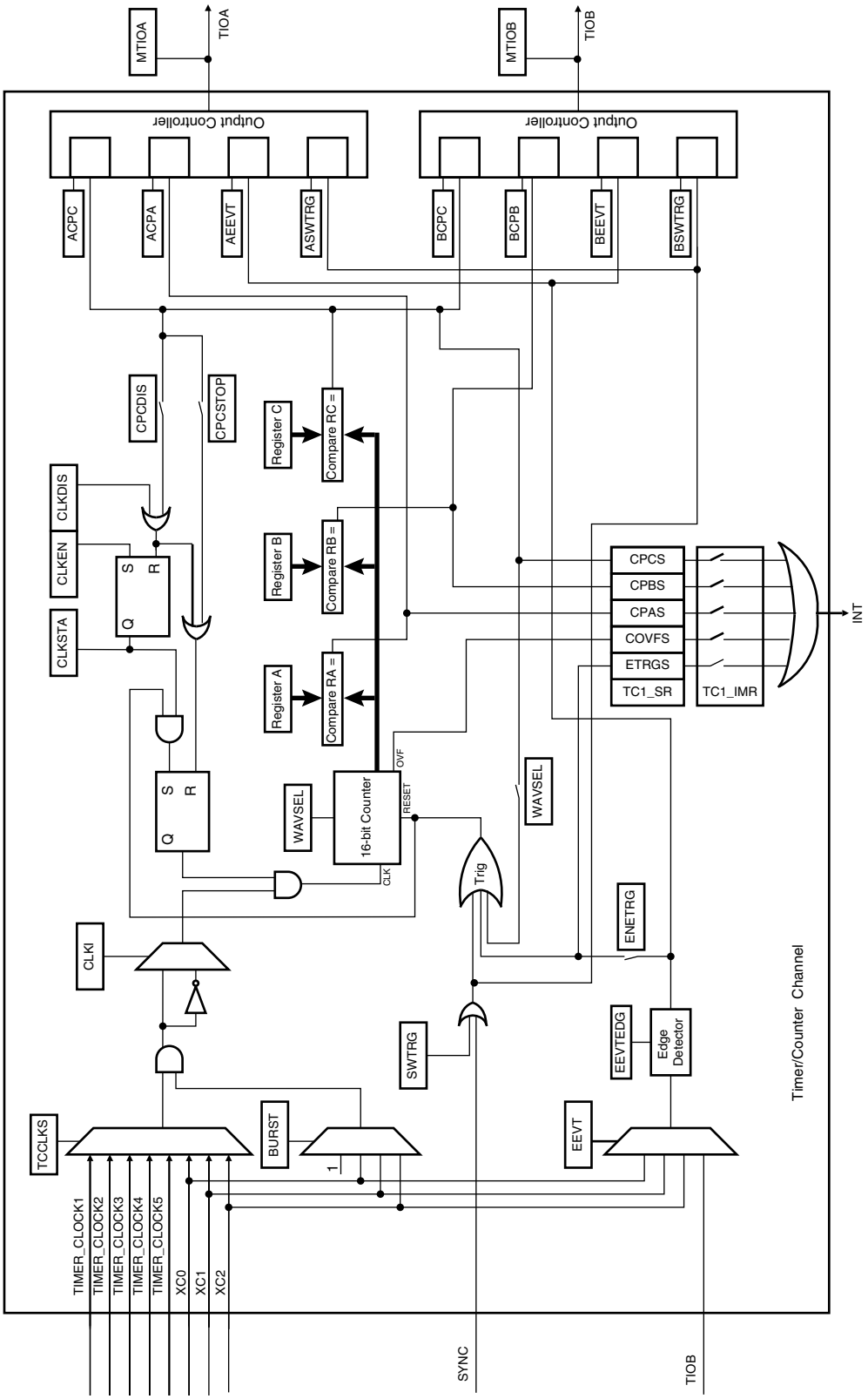
39.5.11 Waveform Selection

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

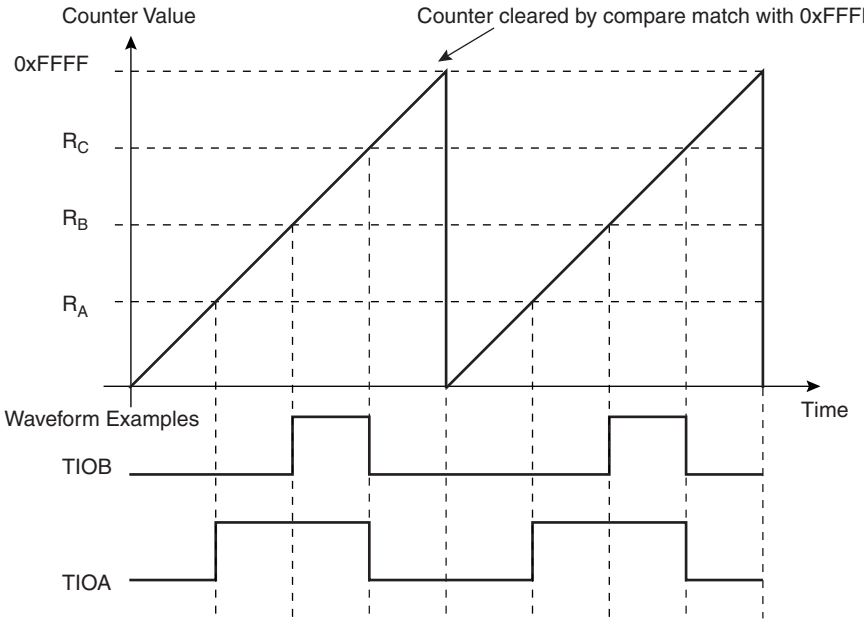
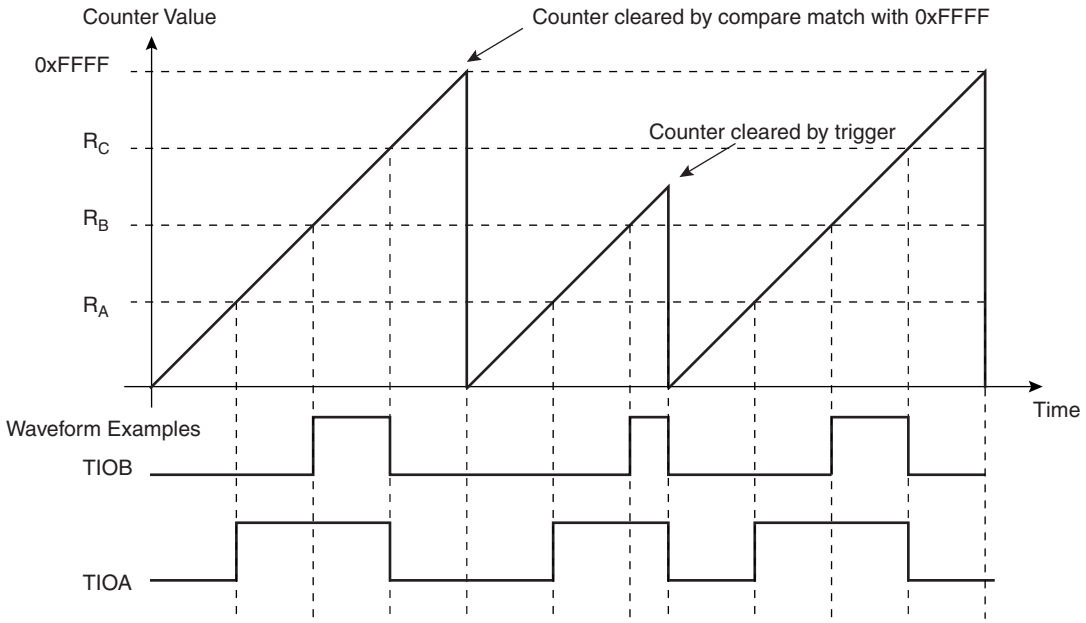


Figure 39-8. WAVSEL= 00 with trigger



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

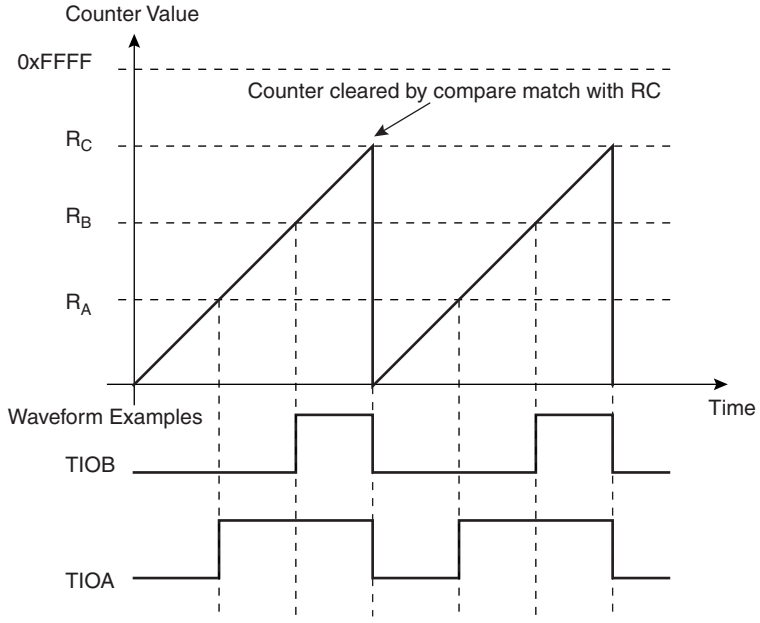
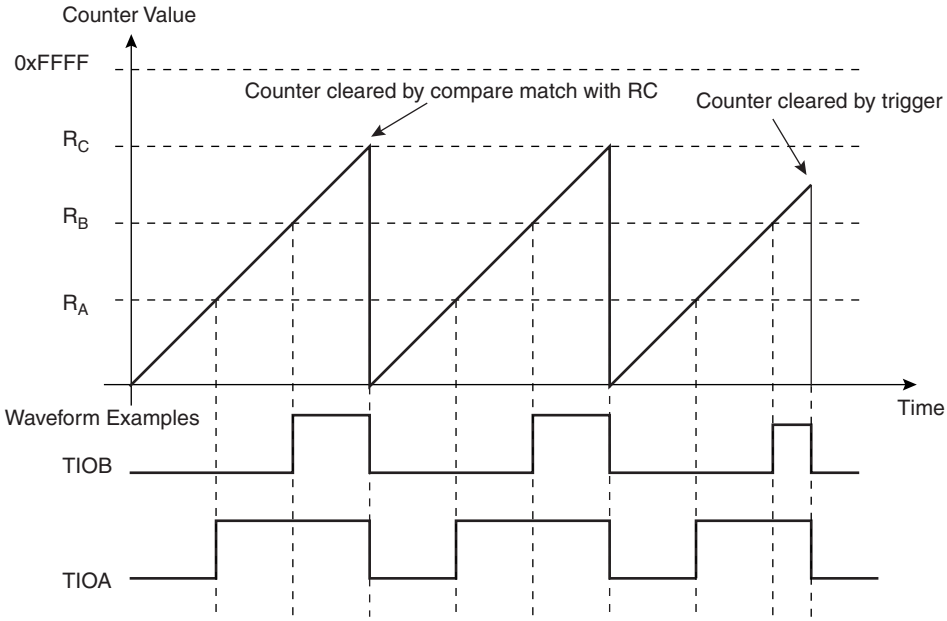


Figure 39-10. WAVSEL = 10 With Trigger



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

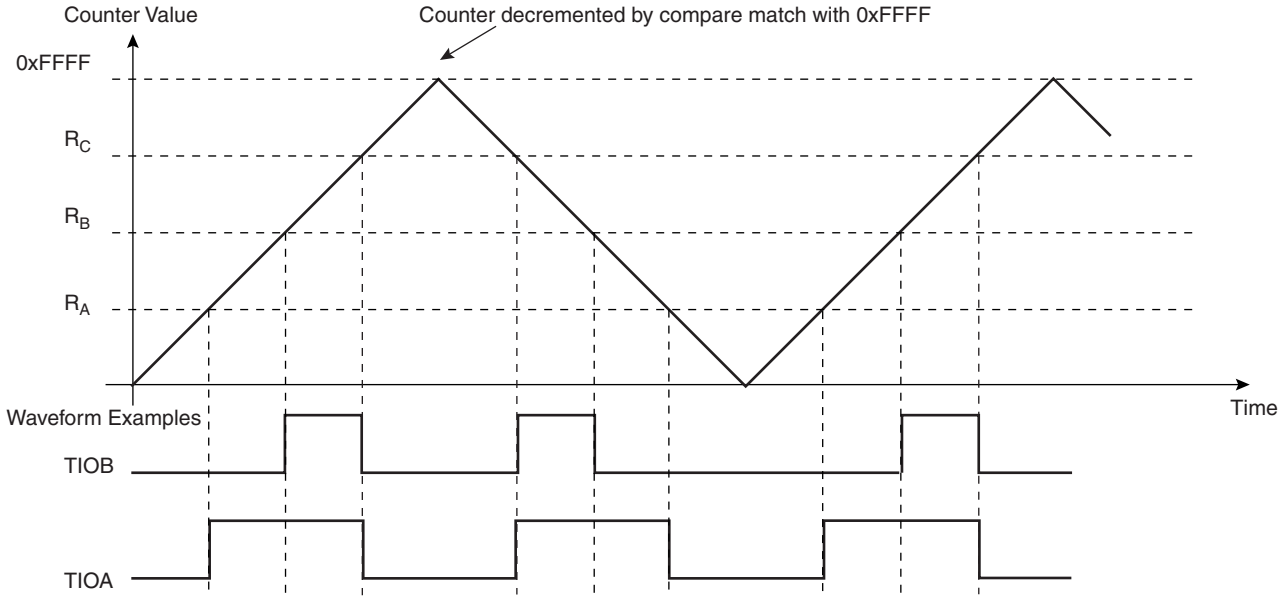
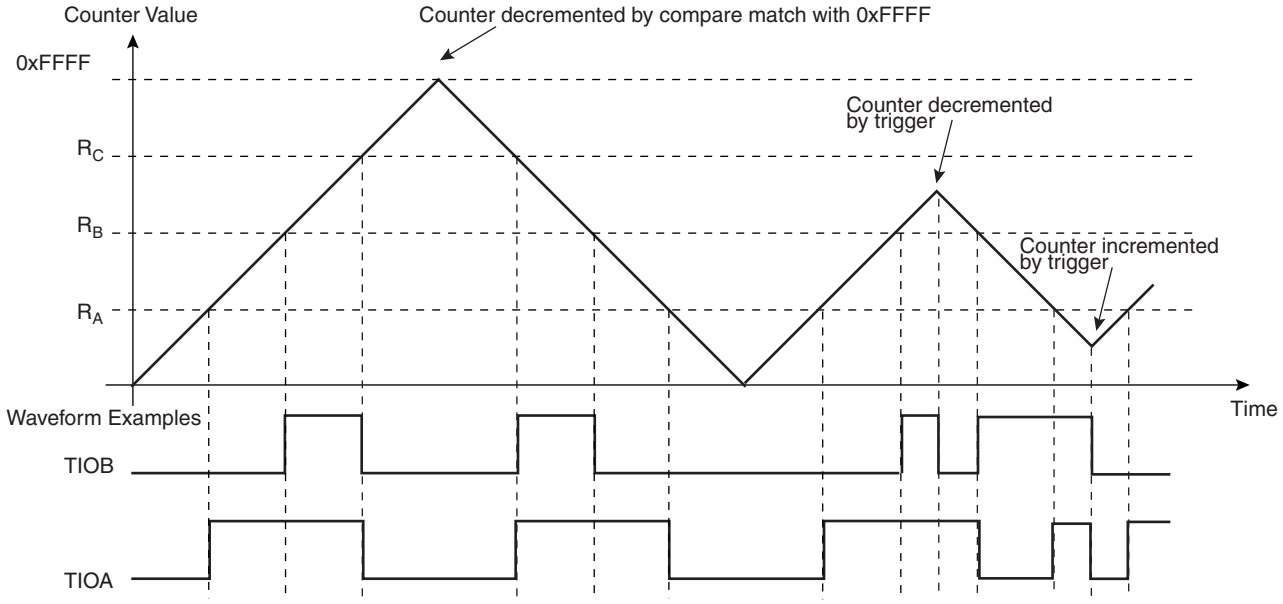


Figure 39-12. WAVSEL = 01 With Trigger



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

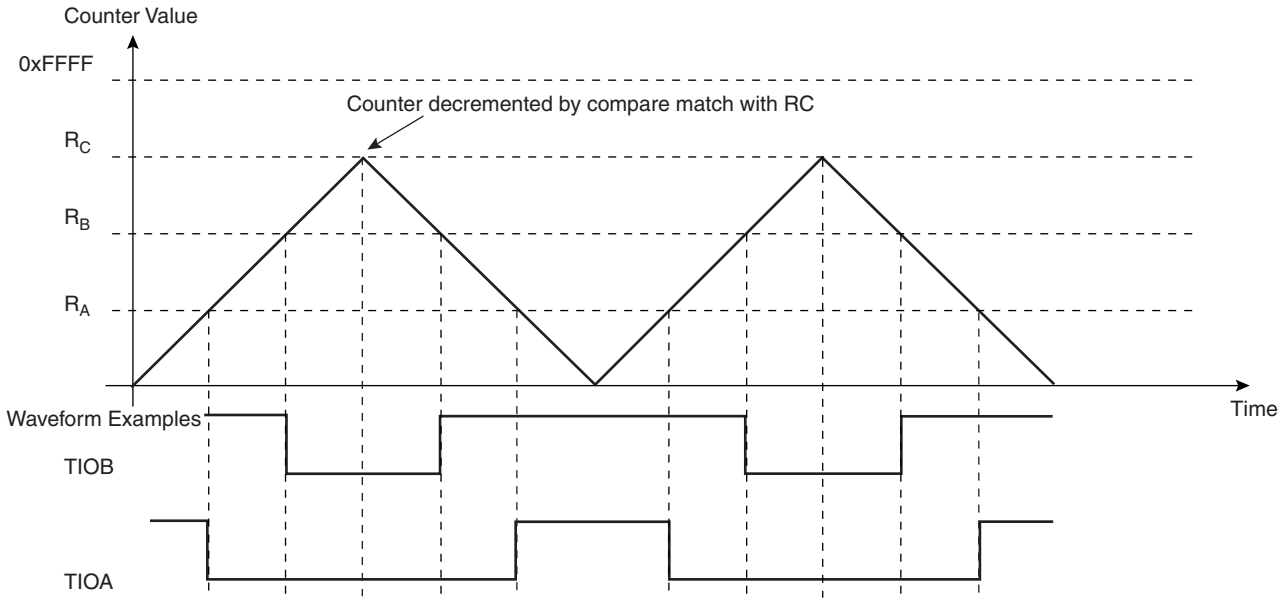
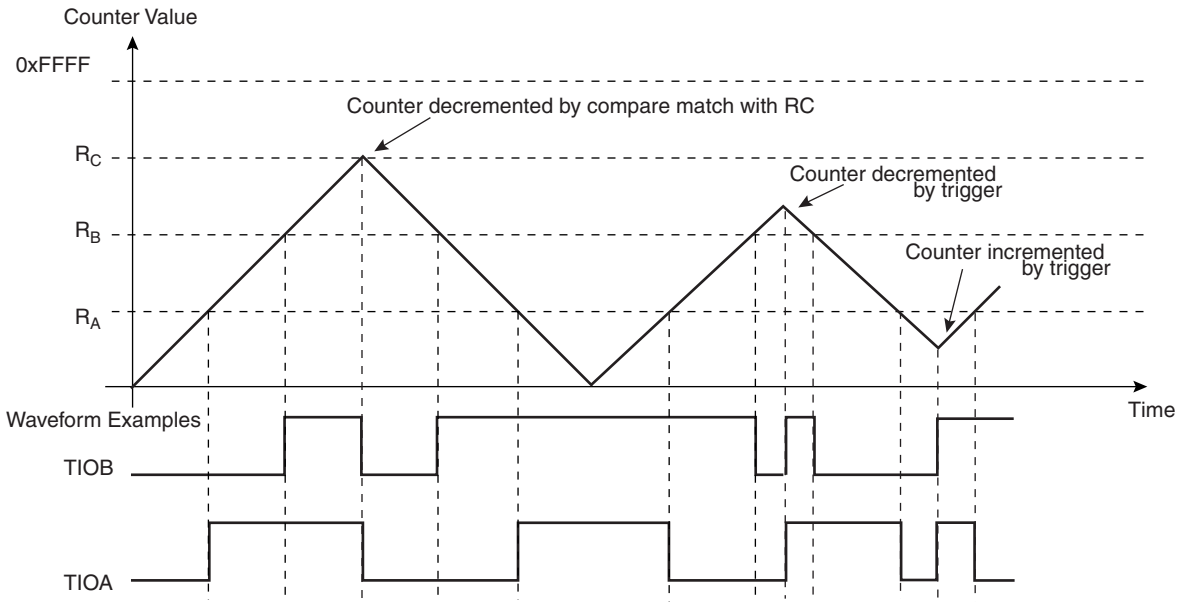


Figure 39-14. WAVSEL = 11 With Trigger



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 39-4. 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

Register Name: TC_BCR

Address: 0xFFF7C0C0

Access Type: Write-only

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

- **SYNC: Synchro Command**

0 = No effect.

1 = Asserts the SYNC signal which generates a software trigger simultaneously for each of the channels.

39.6.2 TC Block Mode Register

Register Name: TC_BMR

Address: 0xFFF7C0C4

Access Type: Read-write

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

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

TC0XC0S		Signal Connected to XC0
0	0	TCLK0
0	1	none
1	0	TIOA1
1	1	TIOA2

- **TC1XC1S: External Clock Signal 1 Selection**

TC1XC1S		Signal Connected to XC1
0	0	TCLK1
0	1	none
1	0	TIOA0
1	1	TIOA2

- **TC2XC2S: External Clock Signal 2 Selection**

TC2XC2S		Signal Connected to XC2
0	0	TCLK2
0	1	none
1	0	TIOA0
1	1	TIOA1

39.6.3 TC Channel Control Register

Register Name:TC_CCRx [x=0..2]

Addresses: 0xFFF7C000 (0)[0], 0xFFF7C040 (0)[1], 0xFFF7C080 (0)[2]

Access Type: Write-only

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

- **CLKEN: Counter Clock Enable Command**

0 = No effect.

1 = Enables the clock if CLKDIS is not 1.

- **CLKDIS: Counter Clock Disable Command**

0 = No effect.

1 = Disables the clock.

- **SWTRG: Software Trigger Command**

0 = No effect.

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

39.6.4 TC Channel Mode Register: Capture Mode

Register Name: TC_CMRx [x=0..2] (WAVE = 0)

Addresses: 0xFFF7C004 (0)[0], 0xFFF7C044 (0)[1], 0xFFF7C084 (0)[2]

Access Type: Read-write

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

Register Name: TC_CMRx [x=0..2] (WAVE = 1)

Addresses: 0xFFF7C004 (0)[0], 0xFFF7C044 (0)[1], 0xFFF7C084 (0)[2]

Access Type: Read-write

31	30	29	28	27	26	25	24
BSWTRG		BEEVT		BCPC		BCPB	
23	22	21	20	19	18	17	16
ASWTRG		AEEVT		ACPC		ACPA	
15	14	13	12	11	10	9	8
WAVE	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

Register Name:TC_CVx [x=0..2]

Addresses: 0xFFF7C010 (0)[0], 0xFFF7C050 (0)[1], 0xFFF7C090 (0)[2]

Access Type: Read-only

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

- **CV: Counter Value**

CV contains the counter value in real time.

39.6.7 TC Register A

Register Name:TC_RAx [x=0..2]

Addresses: 0xFFF7C014 (0)[0], 0xFFF7C054 (0)[1], 0xFFF7C094 (0)[2]

Access Type: Read-only if WAVE = 0, Read-write if WAVE = 1

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

- **RA: Register A**

RA contains the Register A value in real time.

39.6.8 TC Register B

Register Name:TC_RBx [x=0..2]

Addresses: 0xFFF7C018 (0)[0], 0xFFF7C058 (0)[1], 0xFFF7C098 (0)[2]

Access Type: Read-only if WAVE = 0, Read-write if WAVE = 1

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

- **RB: Register B**

RB contains the Register B value in real time.

39.6.9 TC Register C

Register Name:TC_RCx [x=0..2]

Addresses: 0xFFF7C01C (0)[0], 0xFFF7C05C (0)[1], 0xFFF7C09C (0)[2]

Access Type: Read-write

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

- **RC: Register C**

RC contains the Register C value in real time.

39.6.10 TC Status Register

Register Name:TC_SRx [x=0..2]

Addresses: 0xFFF7C020 (0)[0], 0xFFF7C060 (0)[1], 0xFFF7C0A0 (0)[2]

Access Type: Read-only

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

- **COVFS: Counter Overflow Status**

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

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

- **LOVRS: Load Overrun Status**

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

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

- **CPAS: RA Compare Status**

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

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

- **CPBS: RB Compare Status**

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

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

- **CPCS: RC Compare Status**

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

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

- **LDRAS: RA Loading Status**

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

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

- **LDRBS: RB Loading Status**

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

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

- **ETRGS: External Trigger Status**

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

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

- **CLKSTA: Clock Enabling Status**

0 = Clock is disabled.

1 = Clock is enabled.

- **MTIOA: TIOA Mirror**

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

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

- **MTIOB: TIOB Mirror**

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

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

39.6.11 TC Interrupt Enable Register

Register Name:TC_IERx [x=0..2]

Addresses: 0xFFF7C024 (0)[0], 0xFFF7C064 (0)[1], 0xFFF7C0A4 (0)[2]

Access Type: Write-only

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

- **COVFS: Counter Overflow**

0 = No effect.

1 = Enables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun**

0 = No effect.

1 = Enables the Load Overrun Interrupt.

- **CPAS: RA Compare**

0 = No effect.

1 = Enables the RA Compare Interrupt.

- **CPBS: RB Compare**

0 = No effect.

1 = Enables the RB Compare Interrupt.

- **CPCS: RC Compare**

0 = No effect.

1 = Enables the RC Compare Interrupt.

- **LDRAS: RA Loading**

0 = No effect.

1 = Enables the RA Load Interrupt.

- **LDRBS: RB Loading**

0 = No effect.

1 = Enables the RB Load Interrupt.

- **ETRGS: External Trigger**

0 = No effect.

1 = Enables the External Trigger Interrupt.

39.6.12 TC Interrupt Disable Register

Register Name:TC_IDRx [x=0..2]

Addresses: 0xFFF7C028 (0)[0], 0xFFF7C068 (0)[1], 0xFFF7C0A8 (0)[2]

Access Type: Write-only

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

- **COVFS: Counter Overflow**

0 = No effect.

1 = Disables the Counter Overflow Interrupt.

- **LOVRS: Load Overrun**

0 = No effect.

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

- **CPAS: RA Compare**

0 = No effect.

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

- **CPBS: RB Compare**

0 = No effect.

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

- **CPCS: RC Compare**

0 = No effect.

1 = Disables the RC Compare Interrupt.

- **LDRAS: RA Loading**

0 = No effect.

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

- **LDRBS: RB Loading**

0 = No effect.

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

- **ETRGS: External Trigger**

0 = No effect.

1 = Disables the External Trigger Interrupt.

39.6.13 TC Interrupt Mask Register

Register Name:TC_IMRx [x=0..2]

Addresses: 0xFFF7C02C (0)[0], 0xFFF7C06C (0)[1], 0xFFF7C0AC (0)[2]

Access Type: Read-only

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

- **COVFS: Counter Overflow**

0 = The Counter Overflow Interrupt is disabled.

1 = The Counter Overflow Interrupt is enabled.

- **LOVRS: Load Overrun**

0 = The Load Overrun Interrupt is disabled.

1 = The Load Overrun Interrupt is enabled.

- **CPAS: RA Compare**

0 = The RA Compare Interrupt is disabled.

1 = The RA Compare Interrupt is enabled.

- **CPBS: RB Compare**

0 = The RB Compare Interrupt is disabled.

1 = The RB Compare Interrupt is enabled.

- **CPCS: RC Compare**

0 = The RC Compare Interrupt is disabled.

1 = The RC Compare Interrupt is enabled.

- **LDRAS: RA Loading**

0 = The Load RA Interrupt is disabled.

1 = The Load RA Interrupt is enabled.

- **LDRBS: RB Loading**

0 = The Load RB Interrupt is disabled.

1 = The Load RB Interrupt is enabled.

- **ETRGS: External Trigger**

0 = The External Trigger Interrupt is disabled.

1 = The External Trigger Interrupt is enabled.

40. MultiMedia Card Interface (MCI)

40.1 Overview

The MultiMedia Card Interface (MCI) supports the MultiMedia Card (MMC) Specification V3.31, the SDIO Specification V1.1 and the SD Memory Card Specification V1.0.

The MCI includes a command register, response registers, data registers, timeout counters and error detection logic that automatically handle the transmission of commands and, when required, the reception of the associated responses and data with a limited processor overhead.

The MCI supports stream, block and multi-block data read and write, and is compatible with the Peripheral DMA Controller (PDC) channels, minimizing processor intervention for large buffer transfers.

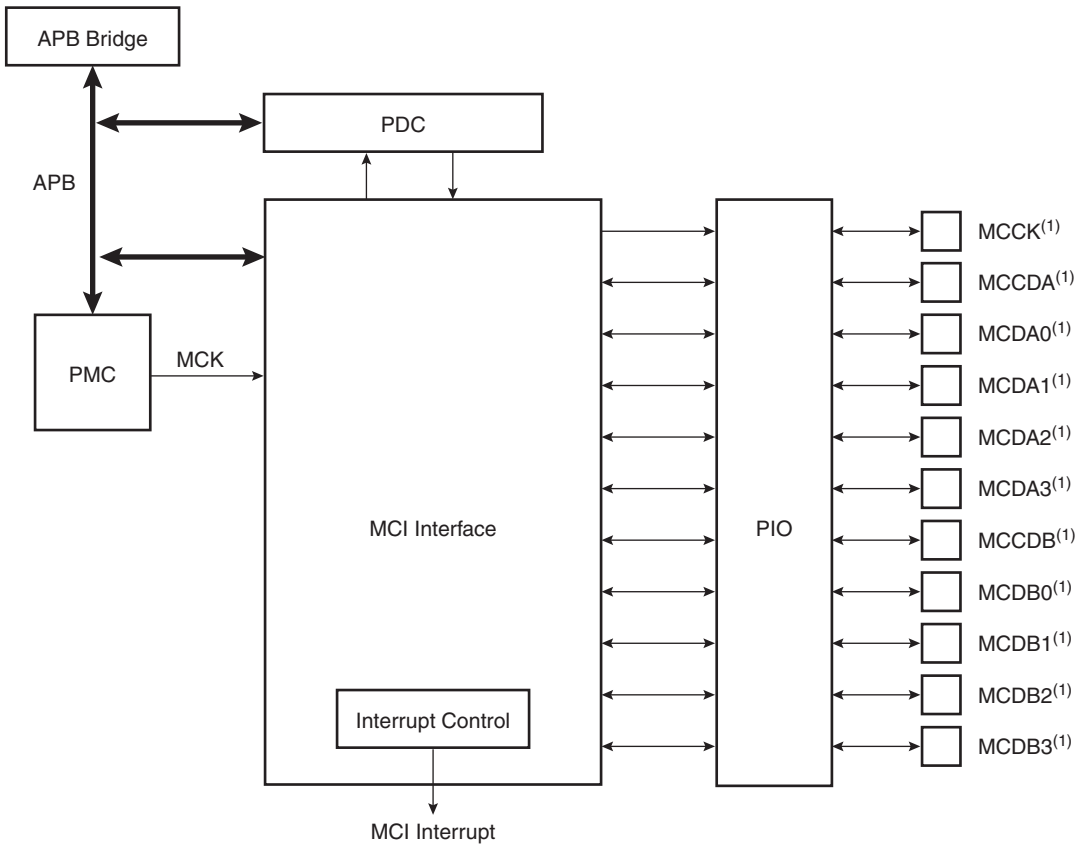
The MCI operates at a rate of up to Master Clock divided by 2 and supports the interfacing of 2 slot(s). Each slot may be used to interface with a MultiMediaCard bus (up to 30 Cards) or with a SD Memory Card. Only one slot can be selected at a time (slots are multiplexed). A bit field in the SD Card Register performs this selection.

The SD Memory Card communication is based on a 9-pin interface (clock, command, four data and three power lines) and the MultiMedia Card on a 7-pin interface (clock, command, one data, three power lines and one reserved for future use).

The SD Memory Card interface also supports MultiMedia Card operations. The main differences between SD and MultiMedia Cards are the initialization process and the bus topology.

40.2 Block Diagram

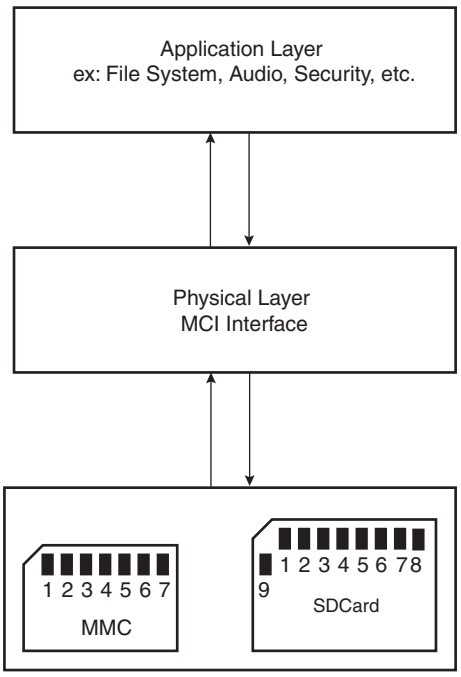
Figure 40-1. Block Diagram



Note: 1. When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx_CK, MCCDA to MCIx_CDA, MCCDB to MCIx_CDB,MCDAy to MCIx_DAy, MCDBy to MCIx_DBy.

40.3 Application Block Diagram

Figure 40-2. Application Block Diagram



40.4 Pin Name List

Table 40-1. I/O Lines Description

Pin Name[2]	Pin Description	Type[1]	Comments
MCCDA/MCCDB	Command/response	I/O/PP/OD	CMD of an MMC or SDCard/SDIO
MCCK	Clock	I/O	CLK of an MMC or SD Card/SDIO
MCDA0 - MCDA3	Data 0..3 of Slot A	I/O/PP	DAT0 of an MMC DAT[0..3] of an SD Card/SDIO
MCDB0 - MCDB3	Data 0..3 of Slot B	I/O/PP	DAT0 of an MMC DAT[0..3] of an SD Card/SDIO

Notes: 1. I: Input, O: Output, PP: Push/Pull, OD: Open Drain.

2. When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx_CK, MCCDA to MCIx_CDA, MCCDB to MCIx_CDB, MCDAy to MCIx_DAy, MCDBy to MCIx_DBy.

40.5 Product Dependencies

40.5.1 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.

40.5.2 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.

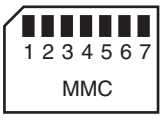
40.5.3 Interrupt

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.

40.6 Bus Topology

Figure 40-3. 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.

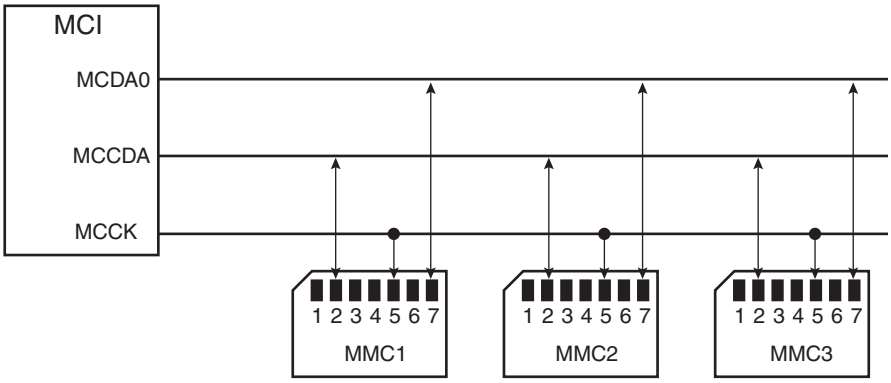
Table 40-2. Bus Topology

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.

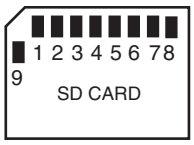
2. When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx_CK, MCCDA to MCIx_CDA, MCCDB to MCIx_CDB, MCDAy to MCIx_DAy, MCDBy to MCIx_DBy.

Figure 40-4. 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 40-5. SD Memory Card Bus Topology



The SD Memory Card bus includes the signals listed in Table 40-3.

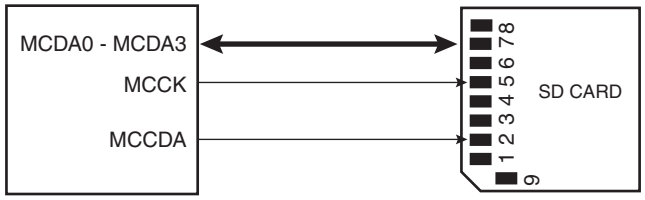
Table 40-3. SD Memory Card Bus Signals

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.

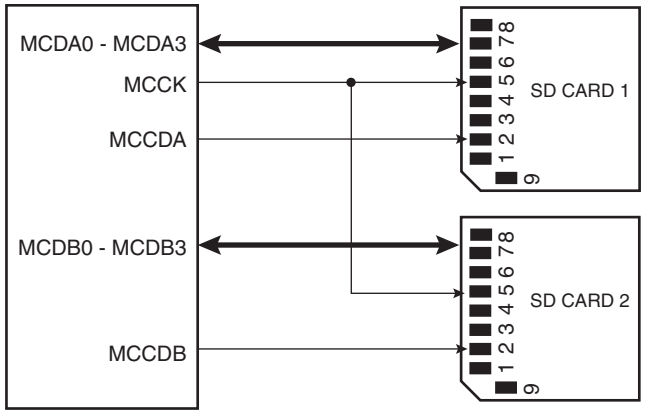
2. When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx_CK, MCCDA to MCIx_CDA, MCCDB to MCIx_CDB, MCDAy to MCIx_DAy, MCDBy to MCIx_DBy.

Figure 40-6. SD Card Bus Connections with One Slot



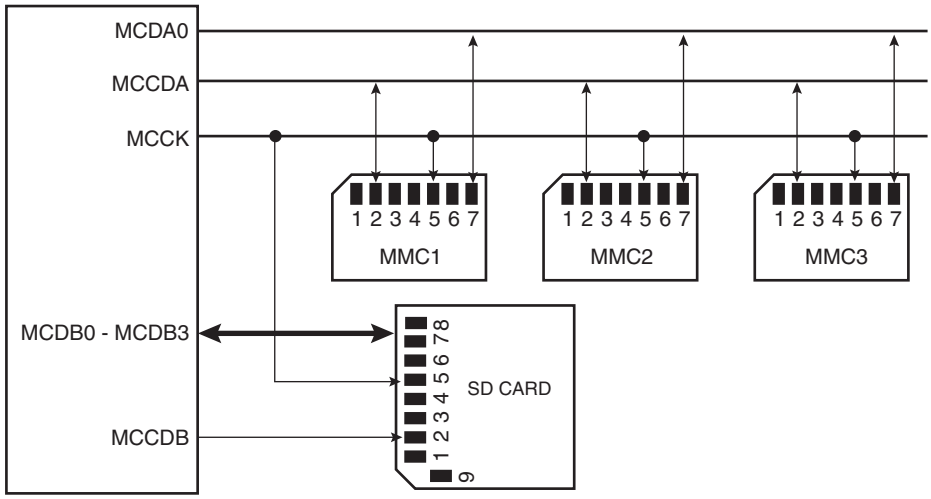
Note: When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx_CK, MCCDA to MCIx_CDA MCDAy to MCIx_DAy.

Figure 40-7. SD Card Bus Connections with Two Slots



Note: When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx_CK,MCCDA to MCIx_CDA, MCDAy to MCIx_DAy, MCCDB to MCIx_CDB, MCDBy to MCIx_DBy.

Figure 40-8. Mixing MultiMedia and SD Memory Cards with Two Slots



Note: When several MCI (x MCI) are embedded in a product, MCCK refers to MCIx_CK, MCCDA to MCIx_CDA, MCDAy to MCIx_DAy, MCCDB to MCIx_CDB, MCDBy to MCIx_DBy.

When the MCI is configured to operate with SD memory cards, the width of the data bus can be selected in the MCI_SDCR register. Clearing the SDCBUS bit in this register means that the width is one bit; setting it means that the width is four bits. In the case of multimedia cards, only the data line 0 is used. The other data lines can be used as independent PIOs.

40.7 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 40-4 on page 774.

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 775.).

The MCI provides a set of registers to perform the entire range of MultiMedia Card operations.

40.7.1 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 40-4 and Table 40-5.

Table 40-4. ALL_SEND_CID Command Description

CMD Index	Type	Argument	Resp	Abbreviation	Command Description
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line

Note: bcr means broadcast command with response.

Table 40-5. Fields and Values for MCI_CMDR Command Register

Field	Value
CMDNB (command number)	2 (CMD2)
RSPTYP (response type)	2 (R2: 136 bits response)
SPCMD (special command)	0 (not a special command)
OPCMD (open drain command)	1
MAXLAT (max latency for command to response)	0 (NID cycles ==> 5 cycles)
TRCMD (transfer command)	0 (No transfer)
TRDIR (transfer direction)	X (available only in transfer command)
TRTYP (transfer type)	X (available only in transfer command)
IOSPCMD (SDIO special command)	0 (not a special command)

The MCI_ARGR contains the argument field of the command.

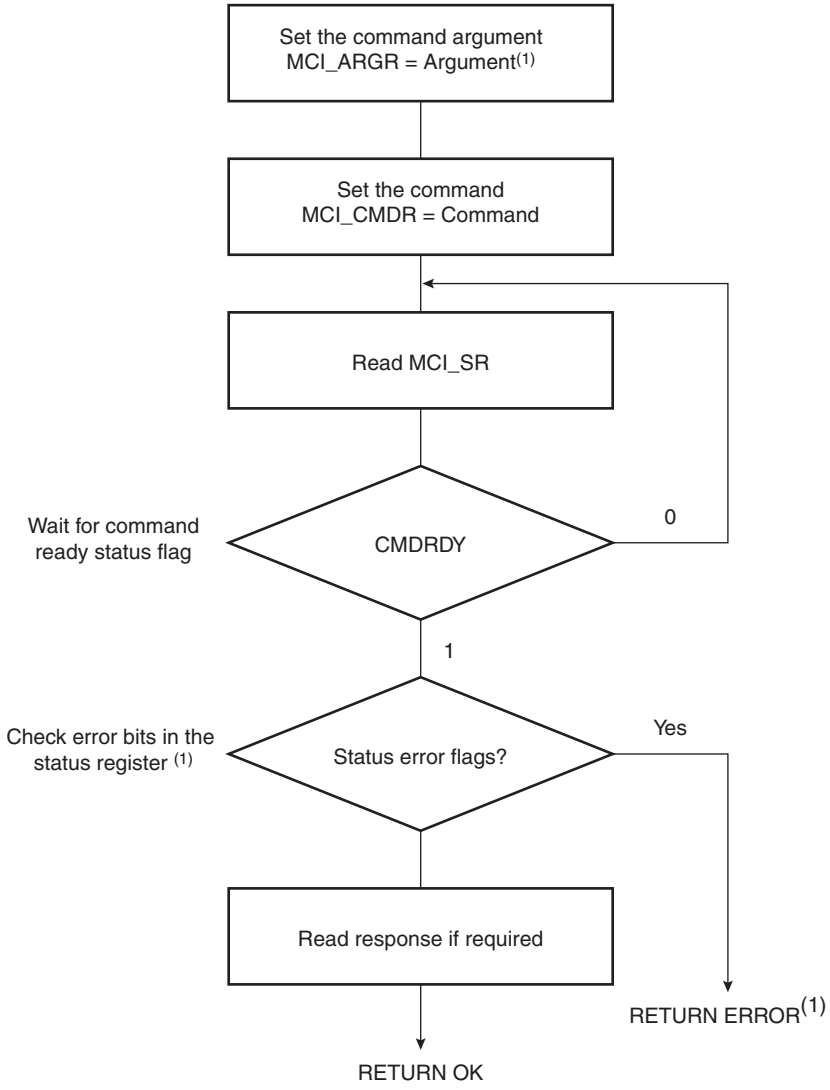
To send a command, the user must perform the following steps:

- Fill the argument register (MCI_ARGR) with the command argument.
- Set the command register (MCI_CMDR) (see Table 40-5).

The command is sent immediately after writing the command register. The status bit CMDRDY in the status register (MCI_SR) is asserted when the command is completed. If the command requires a response, it can be read in the MCI response register (MCI_RSPR). The response size can be from 48 bits up to 136 bits depending on the command. The MCI embeds an error detection to prevent any corrupted data during the transfer.

The following flowchart shows how to send a command to the card and read the response if needed. In this example, the status register bits are polled but setting the appropriate bits in the interrupt enable register (MCI_IER) allows using an interrupt method.

Figure 40-9. Command/Response Functional Flow Diagram



Note: If the command is SEND_OP_COND, the CRC error flag is always present (refer to R3 response in the MultiMedia Card specification).

40.7.2 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.

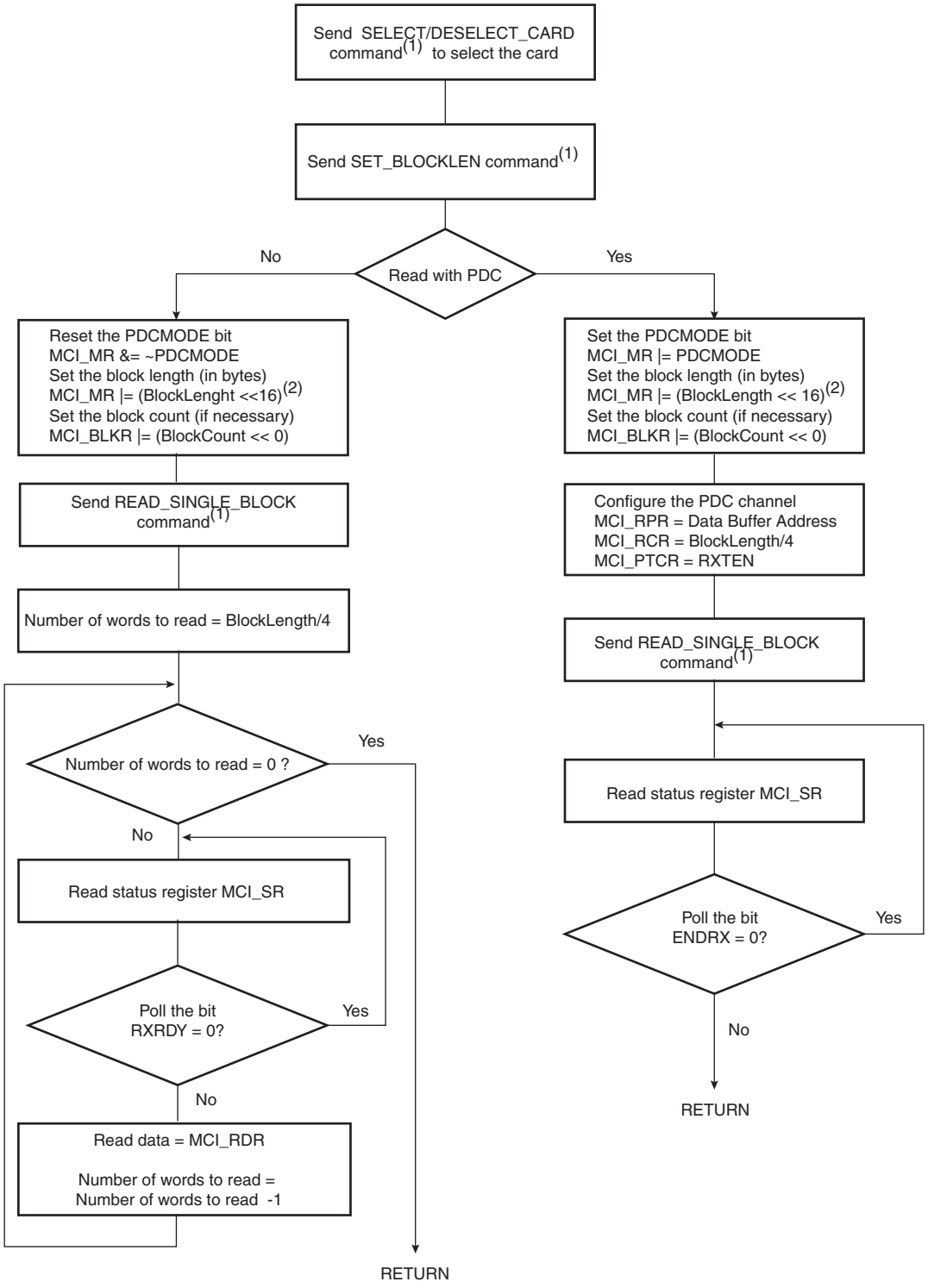
- Multiple block read (or write) with pre-defined block count (since version 3.1 and higher):

  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.

40.7.3 Read Operation

The following flowchart shows how to read a single block with or without use of PDC facilities. In this example (see Figure 40-10), a polling method is used to wait for the end of read. Similarly, the user can configure the interrupt enable register (MCI_IER) to trigger an interrupt at the end of read.

Figure 40-10. Read Functional Flow Diagram



Notes: 1. It is assumed that this command has been correctly sent (see Figure 40-9).
2. This field is also accessible in the MCI Block Register (MCI_BLKR).

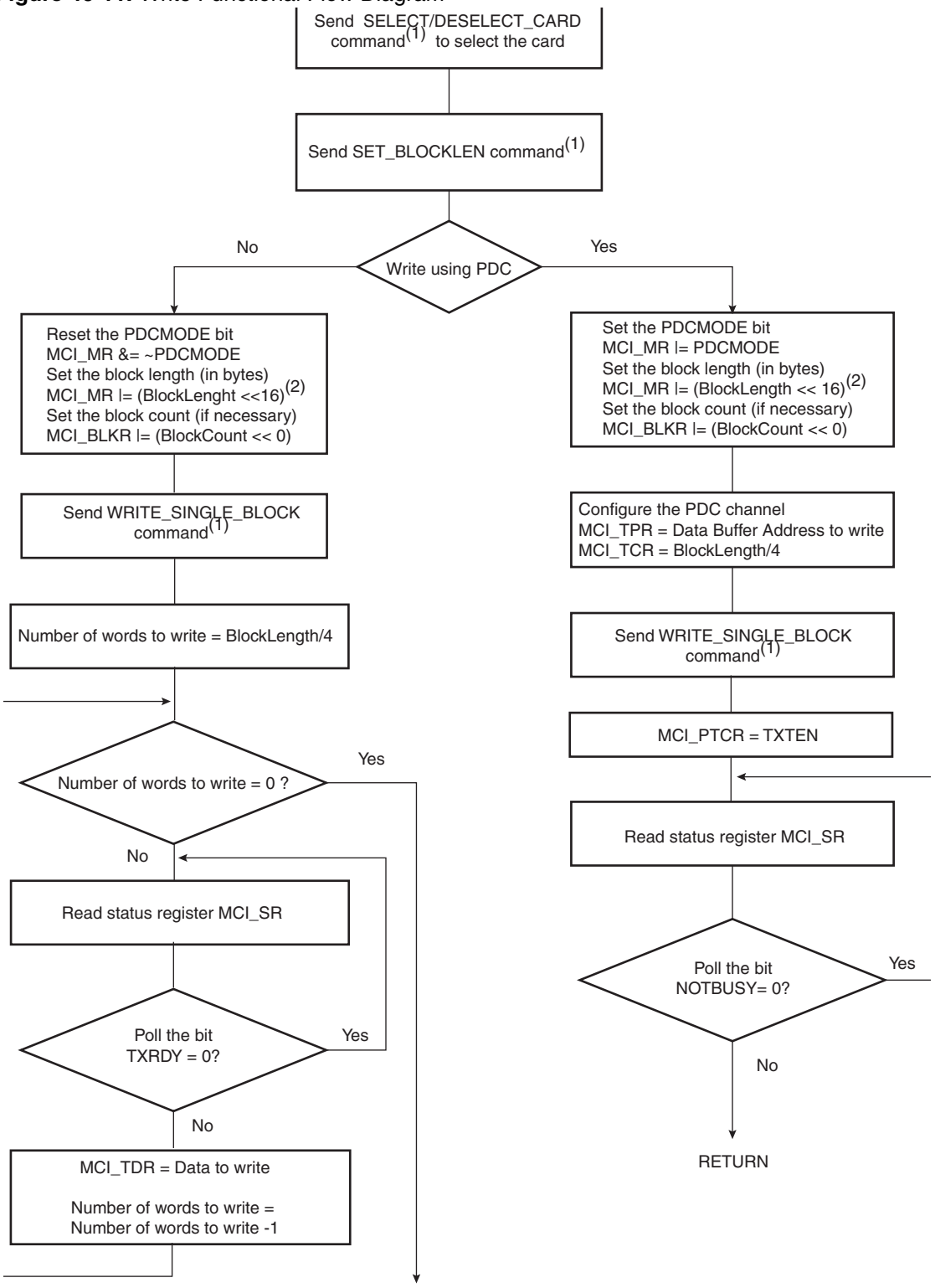
40.7.4 Write Operation

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 40-11). Polling or interrupt method can be used to wait for the end of write according to the contents of the Interrupt Mask Register (MCI_IMR).

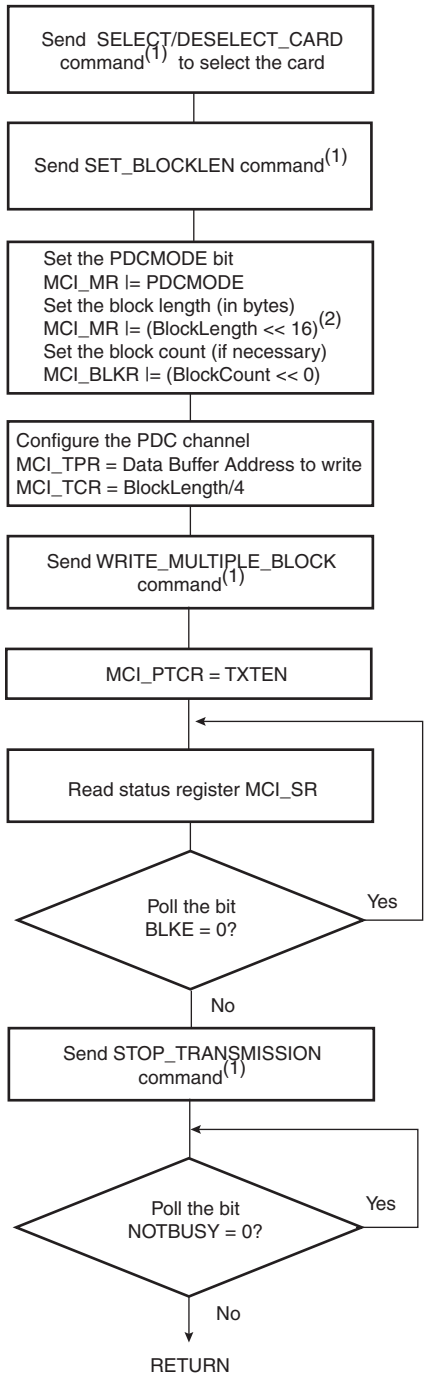
Figure 40-11. Write Functional Flow Diagram



Notes: 1. It is assumed that this command has been correctly sent (see Figure 40-9).
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 40-12). Polling or interrupt method can be used to wait for the end of write according to the contents of the Interrupt Mask Register (MCI_IMR).

Figure 40-12. Multiple Write Functional Flow Diagram



Notes:
1. It is assumed that this command has been correctly sent (see Figure 40-9).
2. This field is also accessible in the MCI Block Register (MCI_BLKR).

40.8 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).

40.8.1 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.

40.8.2 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.

40.9 MultiMedia Card Interface (MCI) User Interface

Table 40-6. 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.

40.9.1 MCI Control Register

Name: MCI_CR

Addresses: 0xFFF80000 (0), 0xFFF84000 (1)

Access Type: Write-only

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

- **MCIEN: Multi-Media Interface Enable**

0 = No effect.

1 = Enables the Multi-Media Interface if MCDIS is 0.

- **MCIDIS: Multi-Media Interface Disable**

0 = No effect.

1 = Disables the Multi-Media Interface.

- **PWSEN: Power Save Mode Enable**

0 = No effect.

1 = Enables the Power Saving Mode if PWSDIS is 0.

Warning: Before enabling this mode, the user must set a value different from 0 in the PWSDIV field (Mode Register MCI_MR).

- **PWSDIS: Power Save Mode Disable**

0 = No effect.

1 = Disables the Power Saving Mode.

- **SWRST: Software Reset**

0 = No effect.

1 = Resets the MCI. A software triggered hardware reset of the MCI interface is performed.

40.9.2 MCI Mode Register

Name: MCI_MR

Addresses: 0xFFF80004 (0), 0xFFF84004 (1)

Access Type: Read-write

31	30	29	28	27	26	25	24
BLKLEN							
23	22	21	20	19	18	17	16
BLKLEN							
15	14	13	12	11	10	9	8
PDCMODE	PDCPADV	PDCFBYTE	WRPROOF	RDPROOF	PWSDIV		
7	6	5	4	3	2	1	0
CLKDIV							

- **CLKDIV: Clock Divider**

Multimedia Card Interface clock (MCCK or MCI_CK) is Master Clock (MCK) divided by (2*(CLKDIV+1)).

- **PWSDIV: Power Saving Divider**

Multimedia Card Interface clock is divided by $2^{(PWSDIV)}$ + 1 when entering Power Saving Mode.

Warning: This value must be different from 0 before enabling the Power Save Mode in the MCI_CR (MCI_PWSEN bit).

- **RDPROOF Read Proof Enable**

Enabling Read Proof allows to stop the MCI Clock during read access if the internal FIFO is full. This 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.

Warning: BLKLEN value depends on PDCFBYTE.

0 = Disables PDC Force Byte Transfer. PDC type of transfer are in words.

1 = Enables PDC Force Byte Transfer. PDC type of transfer are in bytes.

- **PDCPADV: PDC Padding Value**

0 = 0x00 value is used when padding data in write transfer (not only PDC transfer).

1 = 0xFF value is used when padding data in write transfer (not only PDC transfer).

- **PDCMODE: PDC-oriented Mode**

0 = Disables PDC transfer

1 = Enables PDC transfer. In this case, UNRE and OVRE flags in the MCI Mode Register (MCI_SR) are deactivated after the PDC transfer has been completed.

- **BLKLEN: Data Block Length**

This field determines the size of the data block.

This field is also accessible in the MCI Block Register (MCI_BLKR).

Bits 16 and 17 must be set to 0 if PDCFBYTE is disabled.

Note: In SDIO Byte mode, BLKLEN field is not used.

40.9.3 MCI Data Timeout Register

Name: MCI_DTOR

Addresses: 0xFFF80008 (0), 0xFFF84008 (1)

Access Type: Read-write

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

40.9.4 MCI SDCard/SDIO Register

Name: MCI_SDCR

Addresses: 0xFFF8000C (0), 0xFFF8400C (1)

Access Type: Read-write

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

- **SDCSEL: SDCard/SDIO Slot**

SDCSEL		SDCard/SDIO Slot
0	0	Slot A is selected.
0	1	Slot B selected
1	0	Reserved
1	1	Reserved

- **SDCBUS: SDCard/SDIO Bus Width**

0 = 1-bit data bus

1 = 4-bit data bus

40.9.5 MCI Argument Register

Name: MCI_ARGR

Addresses: 0xFFF80010 (0), 0xFFF84010 (1)

Access Type: 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**

40.9.6 MCI Command Register

Name: MCI_CMDR

Addresses: 0xFFF80014 (0), 0xFFF84014 (1)

Access Type: 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 writable 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

40.9.7 MCI Block Register

Name: MCI_BLKR

Addresses: 0xFFF80018 (0), 0xFFF84018 (1)

Access Type: Read-write

31	30	29	28	27	26	25	24
BLKLEN							
23	22	21	20	19	18	17	16
BLKLEN							
15	14	13	12	11	10	9	8
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 65536: Value 0 corresponds to an infinite block transfer.
1	0	0	SDIO Byte	From 1 to 512 bytes: Value 0 corresponds to a 512-byte transfer. Values from 0x200 to 0xFFFF are forbidden.
1	0	1	SDIO Block	From 1 to 511 blocks: Value 0 corresponds to an infinite block transfer. Values from 0x200 to 0xFFFF are forbidden.
Other values			-	Reserved.

Warning: In SDIO Byte and Block modes, writing to the 7 last bits of BCNT field, is forbidden and may lead to unpredictable results.

- **BLKLEN: Data Block Length**

This field determines the size of the data block.

This field is also accessible in the MCI Mode Register (MCI_MR).

Bits 16 and 17 must be set to 0 if PDCFBYTE is disabled.

Note: In SDIO Byte mode, BLKLEN field is not used.

40.9.8 MCI Response Register

Name: MCI_RSPR

Addresses: 0xFFF80020 (0), 0xFFF84020 (1)

Access Type: 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.

40.9.9 MCI Receive Data Register

Name: MCI_RDR

Addresses: 0xFFF80030 (0), 0xFFF84030 (1)

Access Type: Read-only

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

- **DATA: Data to Read**

40.9.10 MCI Transmit Data Register

Name: MCI_TDR

Addresses: 0xFFF80034 (0), 0xFFF84034 (1)

Access Type: Write-only

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

- **DATA: Data to Write**

40.9.11 MCI Status Register

Name: MCI_SR

Addresses: 0xFFF80040 (0), 0xFFF84040 (1)

Access Type: Read-only

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
TXBUFE	RXBUFF	–	–	–	–	SDIOIRQB	SDIOIRQA
7	6	5	4	3	2	1	0
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

- **CMDRDY: Command Ready**

0 = A command is in progress.

1 = The last command has been sent. Cleared when writing in the MCI_CMDR.

- **RXRDY: Receiver Ready**

0 = Data has not yet been received since the last read of MCI_RDR.

1 = Data has been received since the last read of MCI_RDR.

- **TXRDY: Transmit Ready**

0= The last data written in MCI_TDR has not yet been transferred in the Shift Register.

1= The last data written in MCI_TDR has been transferred in the Shift Register.

- **BLKE: Data Block Ended**

This flag must be used only for Write Operations.

0 = A data block transfer is not yet finished. Cleared when reading the MCI_SR.

1 = A data block transfer has ended, including the CRC16 Status transmission.
In PDC mode (PDCMODE=1), the flag is set when the CRC Status of the last block has been transmitted (TXBUFE already set).
Otherwise (PDCMODE=0), the flag is set for each transmitted CRC Status.

Refer to the MMC or SD Specification for more details concerning the CRC Status.

- **DTIP: Data Transfer in Progress**

0 = No data transfer in progress.

1 = The current data transfer is still in progress, including CRC16 calculation. Cleared at the end of the CRC16 calculation.

- **NOTBUSY: MCI Not Busy**

This flag must be used only for Write Operations.

A block write operation uses a simple busy signalling of the write operation duration on the data (DAT0) line: during a data transfer block, if the card does not have a free data receive buffer, the card indicates this condition by pulling down the data

line (DAT0) to LOW. The card stops pulling down the data line as soon as at least one receive buffer for the defined data transfer block length becomes free.

The NOTBUSY flag allows to deal with these different states.

0 = The MCI is not ready for new data transfer. Cleared at the end of the card response.

1 = The MCI is ready for new data transfer. Set when the busy state on the data line has ended. This corresponds to a free internal data receive buffer of the card.

Refer to the MMC or SD Specification for more details concerning the busy behavior.

- **ENDRX: End of RX Buffer**

0 = The Receive Counter Register has not reached 0 since the last write in MCI_RCR or MCI_RNCR.

1 = The Receive Counter Register has reached 0 since the last write in MCI_RCR or MCI_RNCR.

- **ENDTX: End of TX Buffer**

0 = The Transmit Counter Register has not reached 0 since the last write in MCI_TCR or MCI_TNCR.

1 = The Transmit Counter Register has reached 0 since the last write in MCI_TCR or MCI_TNCR.

Note: BLKE and NOTBUSY flags can be used to check that the data has been successfully transmitted on the data lines and not only transferred from the PDC to the MCI Controller.

- **RXBUFF: RX Buffer Full**

0 = MCI_RCR or MCI_RNCR has a value other than 0.

1 = Both MCI_RCR and MCI_RNCR have a value of 0.

- **TXBUFE: TX Buffer Empty**

0 = MCI_TCR or MCI_TNCR has a value other than 0.

1 = Both MCI_TCR and MCI_TNCR have a value of 0.

Note: BLKE and NOTBUSY flags can be used to check that the data has been successfully transmitted on the data lines and not only transferred from the PDC to the MCI Controller.

- **RINDE: Response Index Error**

0 = No error.

1 = A mismatch is detected between the command index sent and the response index received. Cleared when writing in the MCI_CMDR.

- **RDIRE: Response Direction Error**

0 = No error.

1 = The direction bit from card to host in the response has not been detected.

- **RCRCE: Response CRC Error**

0 = No error.

1 = A CRC7 error has been detected in the response. Cleared when writing in the MCI_CMDR.

- **RENDE: Response End Bit Error**

0 = No error.

1 = The end bit of the response has not been detected. Cleared when writing in the MCI_CMDR.

- **RTOE: Response Time-out Error**

0 = No error.

1 = The response time-out set by MAXLAT in the MCI_CMDR has been exceeded. Cleared when writing in the MCI_CMDR.

- **DCRCE: Data CRC Error**

0 = No error.

1 = A CRC16 error has been detected in the last data block. 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.

- **SDIOIRQB: SDIO Interrupt for Slot B**

0 = No interrupt detected on SDIO Slot B.

1 = A SDIO Interrupt on Slot B has reached. Cleared when reading the MCI_SR.

- **RXBUFF: RX Buffer Full**

0 = MCI_RCR or MCI_RNCR has a value other than 0.

1 = Both MCI_RCR and MCI_RNCR have a value of 0.

- **TXBUFE: TX Buffer Empty**

0 = MCI_TCR or MCI_TNCR has a value other than 0.

1 = Both MCI_TCR and MCI_TNCR have a value of 0.

40.9.12 MCI Interrupt Enable Register

Name: MCI_IER

Addresses: 0xFFF80044 (0), 0xFFF84044 (1)

Access Type: Write-only

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
TXBUFE	RXBUFF	–	–	–	–	SDIOIRQB	SDIOIRQA
7	6	5	4	3	2	1	0
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

- **CMDRDY: Command Ready Interrupt Enable**
- **RXRDY: Receiver Ready Interrupt Enable**
- **TXRDY: Transmit Ready Interrupt Enable**
- **BLKE: Data Block Ended Interrupt Enable**
- **DTIP: Data Transfer in Progress Interrupt Enable**
- **NOTBUSY: Data Not Busy Interrupt Enable**
- **ENDRX: End of Receive Buffer Interrupt Enable**
- **ENDTX: End of Transmit Buffer Interrupt Enable**
- **SDIOIRQA: SDIO Interrupt for Slot A Interrupt Enable**
- **SDIOIRQB: SDIO Interrupt for Slot B Interrupt Enable**
- **RXBUFF: Receive Buffer Full Interrupt Enable**
- **TXBUFE: Transmit Buffer Empty Interrupt Enable**
- **RINDE: Response Index Error Interrupt Enable**
- **RDIRE: Response Direction Error Interrupt Enable**
- **RCRCE: Response CRC Error Interrupt Enable**
- **RENDE: Response End Bit Error Interrupt Enable**
- **RTOE: Response Time-out Error Interrupt Enable**
- **DCRCE: Data CRC Error Interrupt Enable**
- **DTOE: Data Time-out Error Interrupt Enable**

- **OVRE: Overrun Interrupt Enable**
- **UNRE: UnderRun Interrupt Enable**

0 = No effect.

1 = Enables the corresponding interrupt.

40.9.13 MCI Interrupt Disable Register

Name: MCI_IDR

Addresses: 0xFFF80048 (0), 0xFFF84048 (1)

Access Type: Write-only

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
TXBUFE	RXBUFF	–	–	–	–	SDIOIRQB	SDIOIRQA
7	6	5	4	3	2	1	0
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

- **CMDRDY: Command Ready Interrupt Disable**
- **RXRDY: Receiver Ready Interrupt Disable**
- **TXRDY: Transmit Ready Interrupt Disable**
- **BLKE: Data Block Ended Interrupt Disable**
- **DTIP: Data Transfer in Progress Interrupt Disable**
- **NOTBUSY: Data Not Busy Interrupt Disable**
- **ENDRX: End of Receive Buffer Interrupt Disable**
- **ENDTX: End of Transmit Buffer Interrupt Disable**
- **SDIOIRQA: SDIO Interrupt for Slot A Interrupt Disable**
- **SDIOIRQB: SDIO Interrupt for Slot B Interrupt Disable**
- **RXBUFF: Receive Buffer Full Interrupt Disable**
- **TXBUFE: Transmit Buffer Empty Interrupt Disable**
- **RINDE: Response Index Error Interrupt Disable**
- **RDIRE: Response Direction Error Interrupt Disable**
- **RCRCE: Response CRC Error Interrupt Disable**
- **RENDE: Response End Bit Error Interrupt Disable**
- **RTOE: Response Time-out Error Interrupt Disable**
- **DCRCE: Data CRC Error Interrupt Disable**
- **DTOE: Data Time-out Error Interrupt Disable**

- **OVRE: Overrun Interrupt Disable**
- **UNRE: UnderRun Interrupt Disable**

0 = No effect.

1 = Disables the corresponding interrupt.

40.9.14 MCI Interrupt Mask Register

Name: MCI_IMR

Addresses: 0xFFF8004C (0), 0xFFF8404C (1)

Access Type: Read-only

31	30	29	28	27	26	25	24
UNRE	OVRE	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	DTOE	DCRCE	RTOE	RENDE	RCRCE	RDIRE	RINDE
15	14	13	12	11	10	9	8
TXBUFE	RXBUFF	–	–	–	–	SDIOIRQB	SDIOIRQA
7	6	5	4	3	2	1	0
ENDTX	ENDRX	NOTBUSY	DTIP	BLKE	TXRDY	RXRDY	CMDRDY

- **CMDRDY: Command Ready Interrupt Mask**
- **RXRDY: Receiver Ready Interrupt Mask**
- **TXRDY: Transmit Ready Interrupt Mask**
- **BLKE: Data Block Ended Interrupt Mask**
- **DTIP: Data Transfer in Progress Interrupt Mask**
- **NOTBUSY: Data Not Busy Interrupt Mask**
- **ENDRX: End of Receive Buffer Interrupt Mask**
- **ENDTX: End of Transmit Buffer Interrupt Mask**
- **SDIOIRQA: SDIO Interrupt for Slot A Interrupt Mask**
- **SDIOIRQB: SDIO Interrupt for Slot B Interrupt Mask**
- **RXBUFF: Receive Buffer Full Interrupt Mask**
- **TXBUFE: Transmit Buffer Empty Interrupt Mask**
- **RINDE: Response Index Error Interrupt Mask**
- **RDIRE: Response Direction Error Interrupt Mask**
- **RCRCE: Response CRC Error Interrupt Mask**
- **RENDE: Response End Bit Error Interrupt Mask**
- **RTOE: Response Time-out Error Interrupt Mask**
- **DCRCE: Data CRC Error Interrupt Mask**
- **DTOE: Data Time-out Error Interrupt Mask**

- **OVRE: Overrun Interrupt Mask**
- **UNRE: UnderRun Interrupt Mask**

0 = The corresponding interrupt is not enabled.

1 = The corresponding interrupt is enabled.

41. Ethernet MAC 10/100 (EMAC)

41.1 Overview

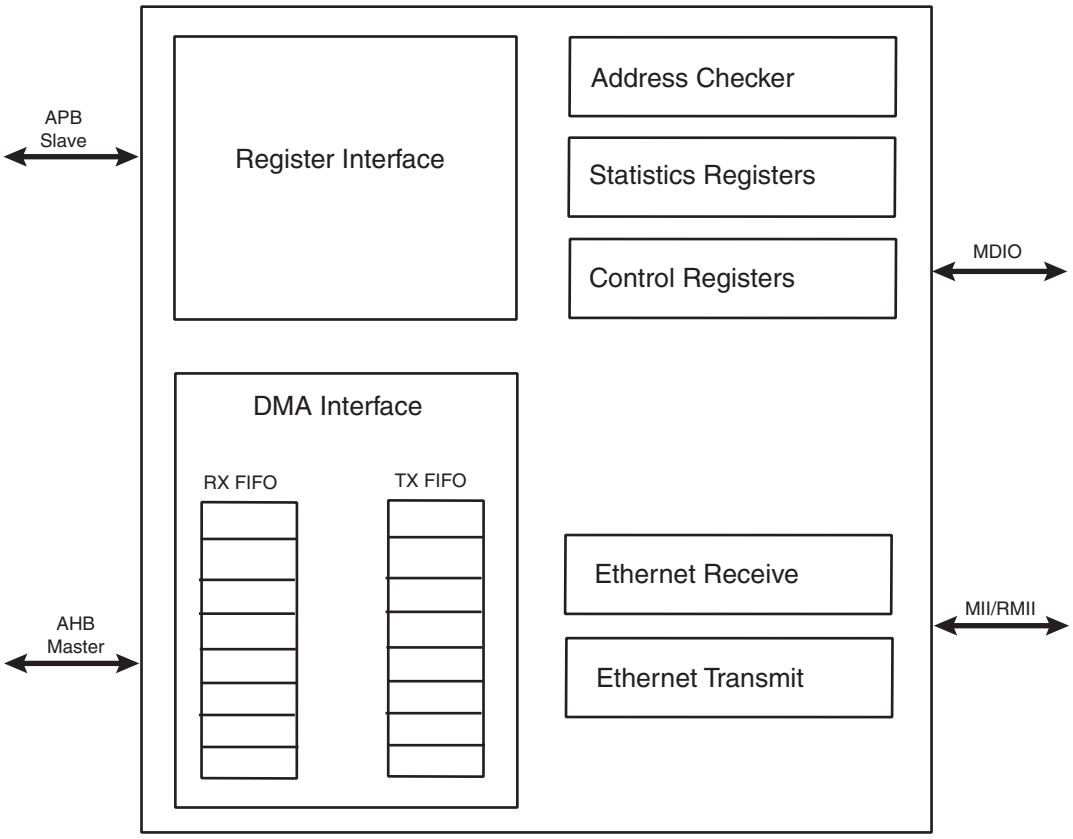
The EMAC module implements a 10/100 Ethernet MAC compatible with the IEEE 802.3 standard using an address checker, statistics and control registers, receive and transmit blocks, and a DMA interface.

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.

41.2 Block Diagram

Figure 41-1. EMAC Block Diagram



41.3 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 41-1 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.

41.3.1 Clock

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.

41.3.2 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

41.3.2.1 FIFO

The FIFO depths are 28 bytes for receive and 28 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 three more. For transmit, a bus request is generated when there is space for four words, or when there is space for two 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 (12 bytes) of data.

At 100 Mbit/s, it takes 960 ns to transmit or receive 12 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 60 MHz master clock this takes 100 ns, making the bus latency requirement 860 ns.

41.3.2.2 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 41-1 for details of the receive buffer descriptor list.

Table 41-1. 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

Table 41-1. Receive Buffer Descriptor Entry (Continued)

Bit	Function
29	Unicast hash match
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 1024th 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 regis-

ter 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.

41.3.2.3 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 41-2 on page 808 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 41-2. Transmit Buffer Descriptor Entry

Bit	Function
Word 0	
31:0	**Byte Address of buffer**
Word 1	

Table 41-2. 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

41.3.3 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.

41.3.4 Pause Frame Support

The start of an 802.3 pause frame is as follows:

Table 41-3. 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).

41.3.5 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.

41.3.6 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

41.3.7 Broadcast Address

The broadcast address of 0xFFFFFFFFFFFF is recognized if the 'no broadcast' bit in the network configuration register is zero.

41.3.8 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.

41.3.9 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.

41.3.10 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.

41.3.11 VLAN Support

An Ethernet encoded 802.1Q VLAN tag looks like this:

Table 41-4. 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

41.3.12 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 820.

41.3.13 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 41-5.

Table 41-5. 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.

41.3.13.1 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.

41.4 Programming Interface

41.4.1 Initialization

41.4.1.1 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:

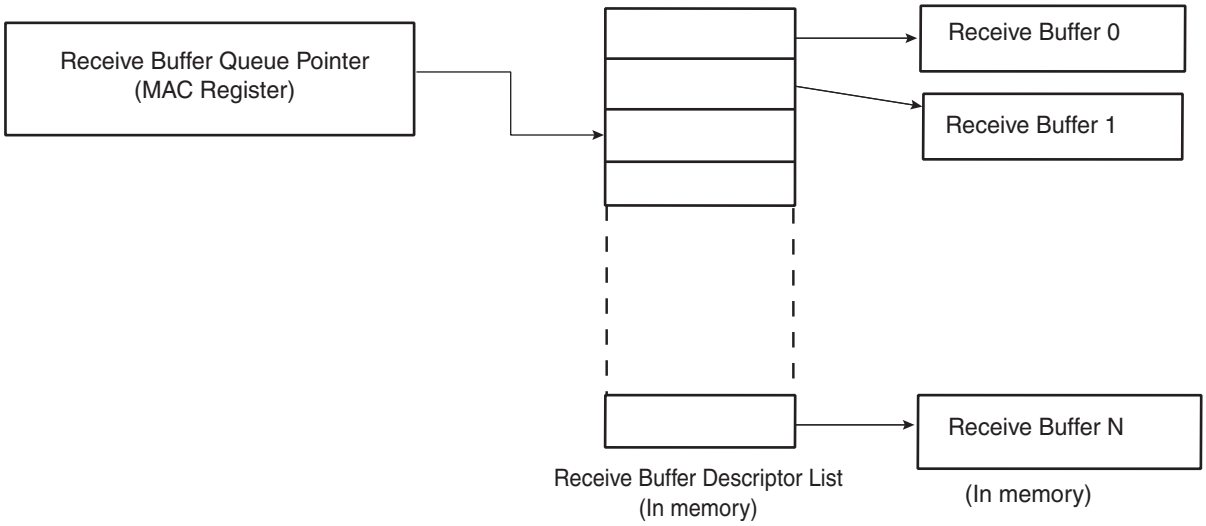
1. Write to network control register to disable transmit and receive circuits.
2. Write to network control register to change loop-back mode.
3. Write to network control register to re-enable transmit or receive circuits.

Note: These writes to network control register cannot be combined in any way.

41.4.1.2 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 805. It points to this data structure.

Figure 41-2. 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.

41.4.1.3 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 41-2 on page 808) 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.

41.4.1.4 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 811. for details of address matching. Each register-pair may be written at any time, regardless of whether the receive circuits are enabled or disabled.

41.4.1.5 Interrupts

There are 14 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.

41.4.1.6 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.

41.4.1.7 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 41-1 on page 805) 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.

41.5 Ethernet MAC 10/100 (EMAC) User Interface

Table 41-6. 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_3FFF
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 41-6. 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
0xC8 - 0xFC	Reserved	–	–	–

41.5.1 Network Control Register

Register Name:EMAC_NCR

Address: 0xFFFBC000

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
–	–	–	–	–	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.

41.5.2 Network Configuration Register

Register Name:EMAC_NCFG

Address: 0xFFFBC004

Access Type: 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.

41.5.3 Network Status Register

Register Name:EMAC_NSR

Address: 0xFFFBC008

Access Type: Read-only

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

41.5.4 Transmit Status Register

Register Name: EMAC_TSR

Address: 0xFFFBC014

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
–	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.

41.5.5 Receive Buffer Queue Pointer Register

Register Name: EMAC_RBQP

Address: 0xFFFBC018

Access Type: 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.

41.5.6 Transmit Buffer Queue Pointer Register

Register Name: EMAC_TBQP

Address: 0xFFFBC01C

Access Type: 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.

41.5.7 Receive Status Register

Register Name: EMAC_RSR

Address: 0xFFFBC020

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
–	–	–	–	–	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.

41.5.8 Interrupt Status Register

Register Name:EMAC_ISR

Address: 0xFFFBC024

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
–	–	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.

41.5.9 Interrupt Enable Register

Register Name:EMAC_IER

Address: 0xFFFBC028

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
–	–	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.

41.5.10 Interrupt Disable Register

Register Name:EMAC_IDR

Address: 0xFFFBC02C

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
–	–	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.

41.5.11 Interrupt Mask Register

Register Name:EMAC_IMR

Address: 0xFFFBC030

Access Type: Read-only

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

41.5.12 PHY Maintenance Register

Register Name: EMAC_MAN

Address: 0xFFFBC034

Access Type: 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.

41.5.13 Pause Time Register

Register Name:EMAC_PTR

Address: 0xFFFBC038

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
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.

41.5.14 Hash Register Bottom

Register Name:EMAC_HRB

Address: 0xFFFBC090

Access Type: 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 812.

41.5.15 Hash Register Top

Register Name:EMAC_HRT

Address: 0xFFFBC094

Access Type: 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 812.

41.5.16 Specific Address 1 Bottom Register

Register Name:EMAC_SA1B

Address: 0xFFFBC098

Access Type: 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.

41.5.17 Specific Address 1 Top Register

Register Name:EMAC_SA1T

Address: 0xFFFBC09C

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
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.

41.5.18 Specific Address 2 Bottom Register

Register Name:EMAC_SA2B

Address: 0xFFFBC0A0

Access Type: 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.

41.5.19 Specific Address 2 Top Register

Register Name:EMAC_SA2T

Address: 0xFFFBC0A4

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
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.

41.5.20 Specific Address 3 Bottom Register

Register Name:EMAC_SA3B

Address: 0xFFFBC0A8

Access Type: 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.

41.5.21 Specific Address 3 Top Register

Register Name:EMAC_SA3T

Address: 0xFFFBC0AC

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
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.

41.5.22 Specific Address 4 Bottom Register

Register Name:EMAC_SA4B

Address: 0xFFFBC0B0

Access Type: 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.

41.5.23 Specific Address 4 Top Register

Register Name:EMAC_SA4T

Address: 0xFFFBC0B4

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
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.

41.5.24 Type ID Checking Register

Register Name:EMAC_TID

Address: 0xFFFBC0B8

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
TID							

7	6	5	4	3	2	1	0
TID							

- **TID: Type ID checking**

For use in comparisons with received frames TypeID/Length field.

41.5.25 User Input/Output Register

Register Name:EMAC_USRIO

Address: 0xFFFBC0C0

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
–	–	–	–	–	–	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.

41.5.26 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.

41.5.26.1 Pause Frames Received Register

Register Name:EMAC_PFR

Address: 0xFFFBC03C

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
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.

41.5.26.2 Frames Transmitted OK Register

Register Name:EMAC_FTO

Address: 0xFFFBC040

Access Type: 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.

41.5.26.3 Single Collision Frames Register

Register Name:EMAC_SCF

Address: 0xFFFBC044

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
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.

41.5.26.4 Multicollision Frames Register

Register Name:EMAC_MCF

Address: 0xFFFBC048

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
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.

41.5.26.5 Frames Received OK Register

Register Name:EMAC_FRO

Address: 0xFFFBC04C

Access Type: 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.

41.5.26.6 Frames Check Sequence Errors Register

Register Name:EMAC_FCSE

Address: 0xFFFBC050

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
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.

41.5.26.7 Alignment Errors Register

Register Name:EMAC_ALE

Address: 0xFFFBC054

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
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.

41.5.26.8 Deferred Transmission Frames Register

Register Name:EMAC_DTF

Address: 0xFFFBC058

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
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.

41.5.26.9 Late Collisions Register

Register Name:EMAC_LCOL

Address: 0xFFFBC05C

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
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.

41.5.26.10 Excessive Collisions Register

Register Name:EMAC_ECOL

Address: 0xFFFBC060

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
EXCOL							

- **EXCOL: Excessive Collisions**

An 8-bit register counting the number of frames that failed to be transmitted because they experienced 16 collisions.

41.5.26.11 Transmit Underrun Errors Register

Register Name:EMAC_TUND

Address: 0xFFFBC064

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
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.

41.5.26.12 Carrier Sense Errors Register

Register Name:EMAC_CSE

Address: 0xFFFBC068

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
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.

41.5.26.13 Receive Resource Errors Register

Register Name:EMAC_RRE

Address: 0xFFFBC06C

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
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.

41.5.26.14 Receive Overrun Errors Register

Register Name:EMAC_ROV

Address: 0xFFFBC070

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
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.

41.5.26.15 Receive Symbol Errors Register

Register Name:EMAC_RSE

Address: 0xFFFBC074

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
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.

41.5.26.16 Excessive Length Errors Register

Register Name:EMAC_ELE

Address: 0xFFFBC078

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
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.

41.5.26.17 Receive Jabbers Register

Register Name:EMAC_RJA

Address: 0xFFFBC07C

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
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.

41.5.26.18 Undersize Frames Register

Register Name:EMAC_USF

Address: 0xFFFBC080

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
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.

41.5.26.19 SQE Test Errors Register

Register Name:EMAC_STE

Address: 0xFFFBC084

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
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.

41.5.26.20 Received Length Field Mismatch Register

Register Name:EMAC_RLE

Address: 0xFFFBC088

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
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.

42. USB Host Port (UHP)

42.1 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.

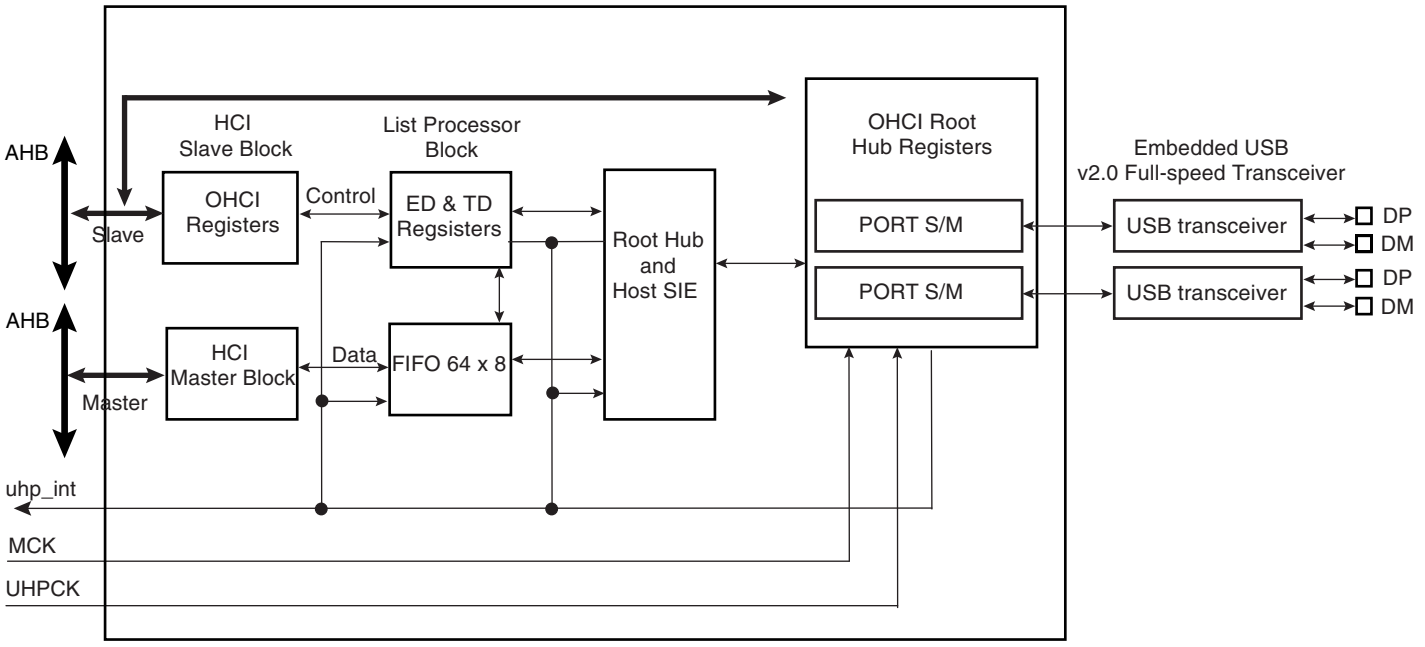
The USB Host Port integrates a root hub and transceivers on downstream ports. It provides several high-speed half-duplex serial communication ports at a baud rate of 12 Mbit/s. Up to 127 USB devices (printer, camera, mouse, keyboard, disk, etc.) and the USB hub can be connected to the USB host in the 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.

42.2 Block Diagram

Figure 42-1. 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 “UnrecoverableError” 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.

Warning: A pull-down must be connected to DP on the board. Otherwise the USB host will permanently detect a device connection on this port.

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.

42.3 Product Dependencies

42.3.1 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.

42.3.2 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).

42.3.3 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.

42.4 Functional Description

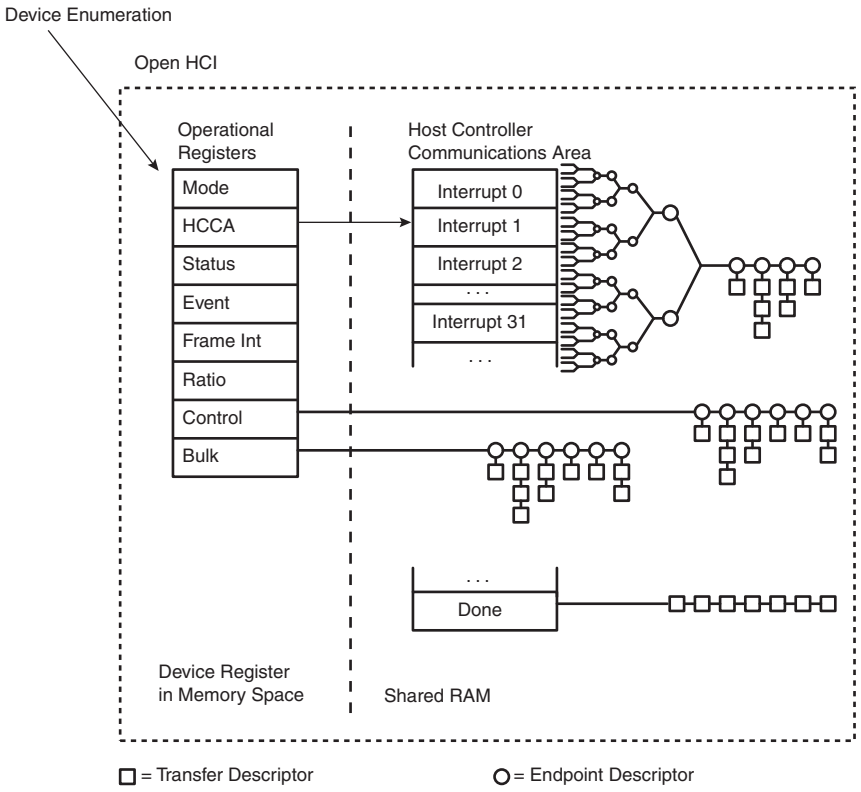
Please refer to the Open Host Controller Interface Specification for USB Release 1.0.a.

42.4.1 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.

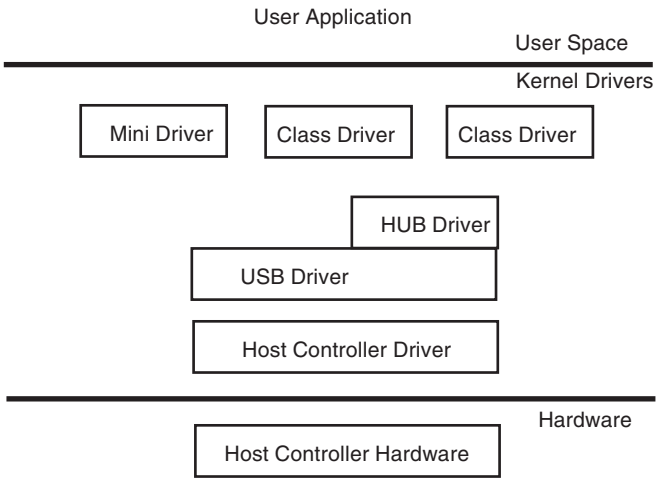
The basic building blocks for communication across the interface are Endpoint Descriptors (ED, 4 double words) and Transfer Descriptors (TD, 4 or 8 double words). The host controller assigns an Endpoint Descriptor to each endpoint in the system. A queue of Transfer Descriptors is linked to the Endpoint Descriptor for the specific endpoint.

Figure 42-2. USB Host Communication Channels



42.4.2 Host Controller Driver

Figure 42-3. USB Host Drivers

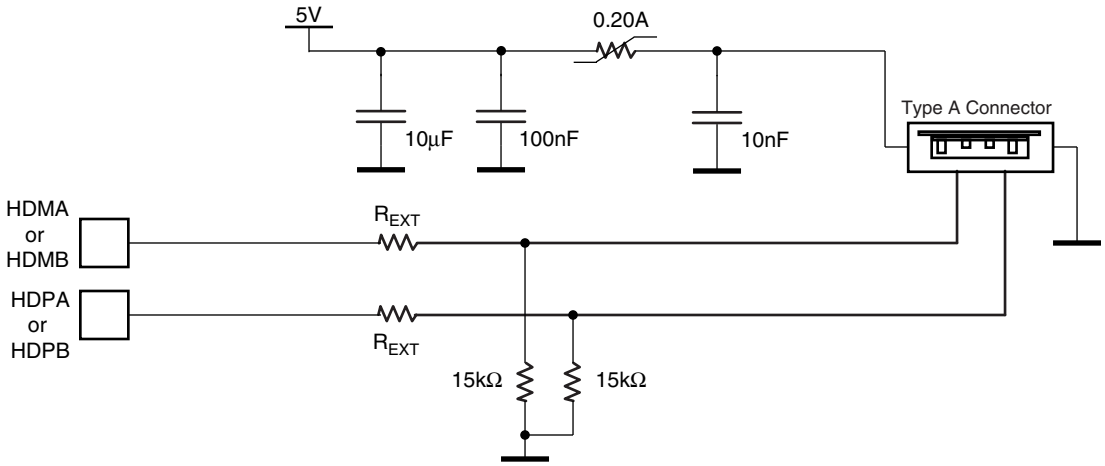


USB Handling is done through several layers as follows:

- 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.

42.5 Typical Connection

Figure 42-4. Board Schematic to Interface UHP Device Controller



As device connection is automatically detected by the USB host port logic, a pull-down must be connected on DP and DM on the board. Otherwise the USB host permanently detects a device connection on this port.

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}).

43. USB Device Port (UDP)

43.1 Overview

The USB Device Port (UDP) is compliant with the Universal Serial Bus (USB) V2.0 full-speed device specification.

Each endpoint can be configured in one of several USB transfer types. It can be associated with one or two banks of a dual-port RAM used to store the current data payload. If two banks are used, one DPR bank is read or written by the processor, while the other is read or written by the USB device peripheral. This feature is mandatory for isochronous endpoints. Thus the device maintains the maximum bandwidth (1M bytes/s) by working with endpoints with two banks of DPR.

Table 43-1. USB Endpoint Description

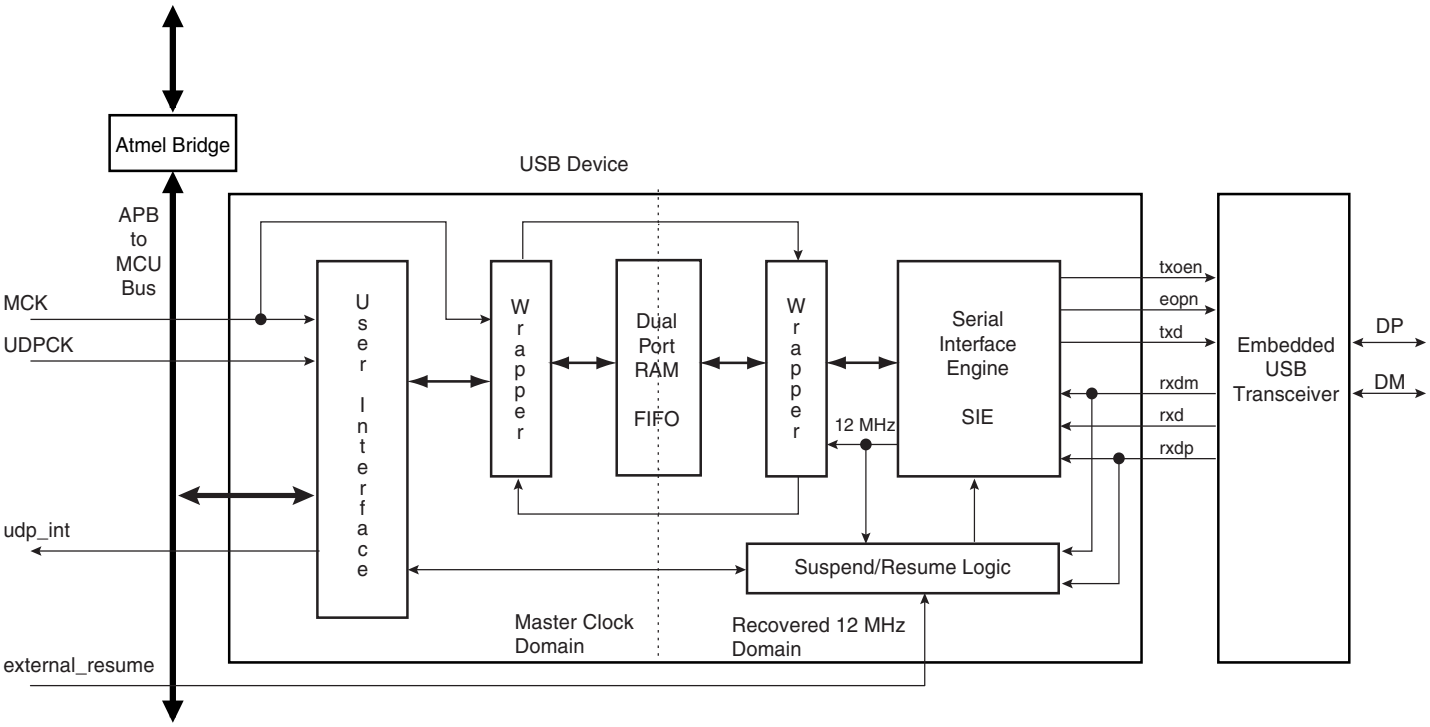
Endpoint Number	Mnemonic	Dual-Bank[1]	Max. Endpoint Size	Endpoint Type
0	EP0	No	64	Control/Bulk/Interrupt
1	EP1	Yes	64	Bulk/Iso/Interrupt
2	EP2	Yes	64	Bulk/Iso/Interrupt
3	EP3	No	64	Control/Bulk/Interrupt
4	EP4	Yes	256	Bulk/Iso/Interrupt
5	EP5	Yes	256	Bulk/Iso/Interrupt

Note: 1. The Dual-Bank function provides two banks for an endpoint. This feature is used for ping-pong mode.

Suspend and resume are automatically detected by the USB device, which notifies the processor by raising an interrupt. Depending on the product, an external signal can be used to send a wake up to the USB host controller.

43.2 Block Diagram

Figure 43-1. Block Diagram



Access to the UDP is via the APB bus interface. Read and write to the data FIFO are done by reading and writing 8-bit values to APB registers.

The UDP peripheral requires two clocks: one peripheral clock used by the Master Clock domain (MCK) and a 48 MHz clock (UDPCK) used by the 12 MHz domain.

A USB 2.0 full-speed pad is embedded and controlled by the Serial Interface Engine (SIE).

The signal external_resume is optional. It allows the UDP peripheral to wake up once in system mode. The host is then notified that the device asks for a resume. This optional feature must be also negotiated with the host during the enumeration.

43.3 Product Dependencies

For further details on the USB Device hardware implementation, see the specific Product Properties document.

The USB physical transceiver is integrated into the product. The bidirectional differential signals DP and DM are available from the product boundary.

One I/O line may be used by the application to check that VBUS is still available from the host. Self-powered devices may use this entry to be notified that the host has been powered off. In this case, the pullup on DP must be disabled in order to prevent feeding current to the host. The application should disconnect the transceiver, then remove the pullup.

43.3.1 I/O Lines

DP and DM are not controlled by any PIO controllers. The embedded USB physical transceiver is controlled by the USB device peripheral.

To reserve an I/O line to check VBUS, the programmer must first program the PIO controller to assign this I/O in input PIO mode.

43.3.2 Power Management

The USB device peripheral requires a 48 MHz clock. This clock must be generated by a PLL with an accuracy of ± 0.25%.

Thus, the USB device receives two clocks from the Power Management Controller (PMC): the master clock, MCK, used to drive the peripheral user interface, and the UDPCK, used to interface with the bus USB signals (recovered 12 MHz domain).

WARNING: The UDP peripheral clock in the Power Management Controller (PMC) must be enabled before any read/write operations to the UDP registers including the UDP_TXCV register.

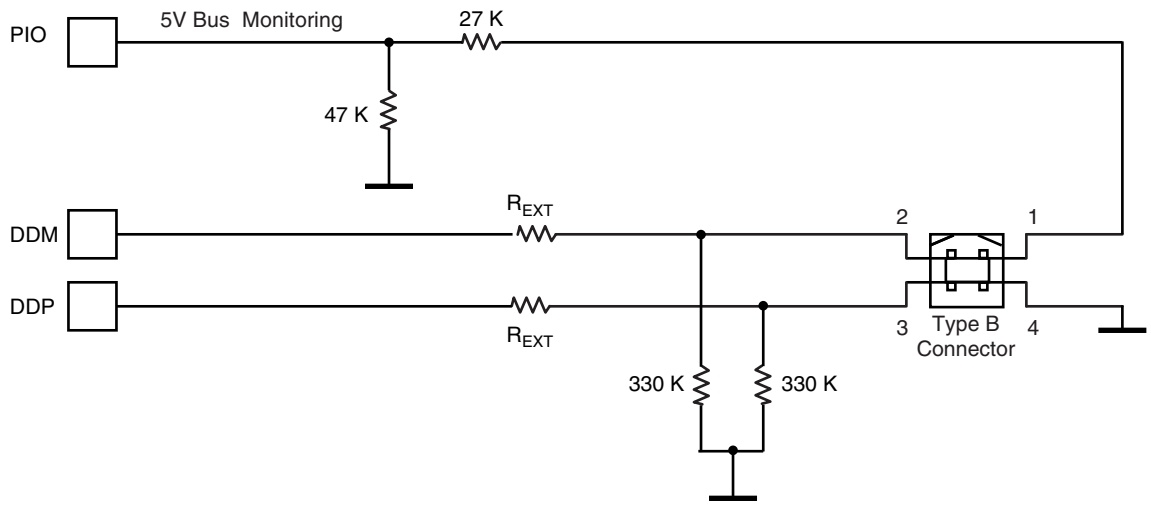
43.3.3 Interrupt

The USB device interface has an interrupt line connected to the Advanced Interrupt Controller (AIC).

Handling the USB device interrupt requires programming the AIC before configuring the UDP.

43.4 Typical Connection

Figure 43-2. Board Schematic to Interface Device Peripheral



43.4.1 USB Device Transceiver

The USB device transceiver is embedded in the product. A few discrete components are required as follows:

- the application detects all device states as defined in chapter 9 of the USB specification;
  - VBUS monitoring
- to reduce power consumption the host is disconnected
- for line termination.

43.4.2 VBUS Monitoring

VBUS monitoring is required to detect host connection. VBUS monitoring is done using a standard PIO with internal pullup disabled. When the host is switched off, it should be considered as a disconnect, the pullup must be disabled in order to prevent powering the host through the pullup resistor.

When the host is disconnected and the transceiver is enabled, then DDP and DDM are floating. This may lead to over consumption. A solution is to connect 330 KΩ pulldowns on DP and DM. These pulldowns do not alter DDP and DDM signal integrity.

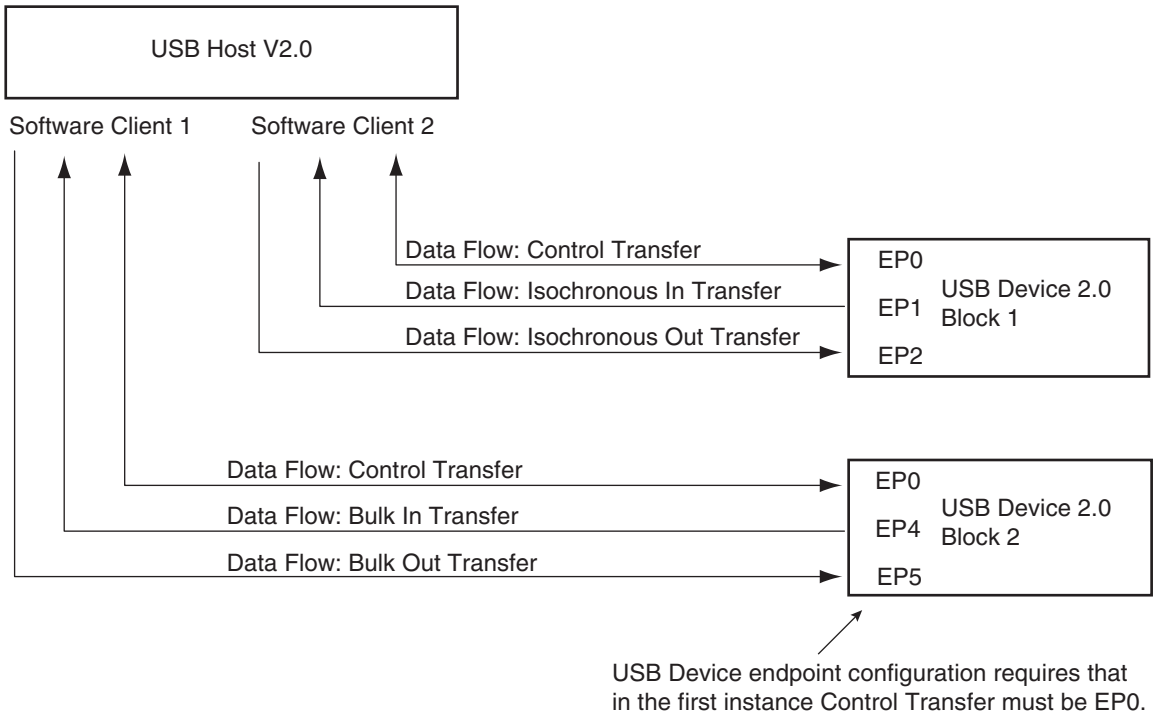
A termination serial resistor must be connected to DP and DM. The resistor value is defined in the electrical specification of the product (R_{EXT}).

43.5 Functional Description

43.5.1 USB V2.0 Full-speed Introduction

The USB V2.0 full-speed provides communication services between host and attached USB devices. Each device is offered with a collection of communication flows (pipes) associated with each endpoint. Software on the host communicates with a USB device through a set of communication flows.

Figure 43-3. Example of USB V2.0 Full-speed Communication Control



The Control Transfer endpoint EP0 is always used when a USB device is first configured (USB v. 2.0 specifications).

43.5.1.1 USB V2.0 Full-speed Transfer Types

A communication flow is carried over one of four transfer types defined by the USB device.

Table 43-2. USB Communication Flow

Transfer	Direction	Bandwidth	Supported Endpoint Size	Error Detection	Retrying
Control	Bidirectional	Not guaranteed	8, 16, 32, 64	Yes	Automatic
Isochronous	Unidirectional	Guaranteed	256	Yes	No
Interrupt	Unidirectional	Not guaranteed	≤64	Yes	Yes
Bulk	Unidirectional	Not guaranteed	8, 16, 32, 64	Yes	Yes

43.5.1.2 USB Bus Transactions

Each transfer results in one or more transactions over the USB bus. There are three kinds of transactions flowing across the bus in packets:

1. Setup Transaction
2. Data IN Transaction
3. Data OUT Transaction

43.5.1.3 *USB Transfer Event Definitions*

As indicated below, transfers are sequential events carried out on the USB bus.

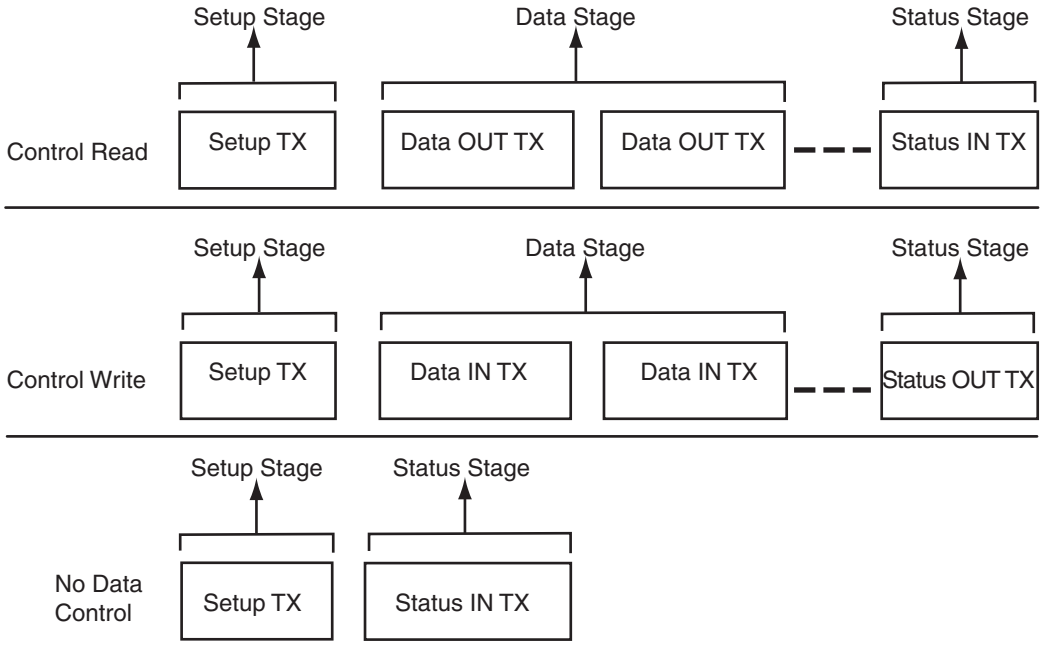
Table 43-3. USB Transfer Events

Control Transfers[1] [3]	• Setup transaction > Data IN transactions > Status OUT transaction • Setup transaction > Data OUT transactions > Status IN transaction • Setup transaction > Status IN transaction
Interrupt IN Transfer (device toward host)	• Data IN transaction > Data IN transaction
Interrupt OUT Transfer (host toward device)	• Data OUT transaction > Data OUT transaction
Isochronous IN Transfer[2] (device toward host)	• Data IN transaction > Data IN transaction
Isochronous OUT Transfer[2] (host toward device)	• Data OUT transaction > Data OUT transaction
Bulk IN Transfer (device toward host)	• Data IN transaction > Data IN transaction
Bulk OUT Transfer (host toward device)	• Data OUT transaction > Data OUT transaction

Notes:
1. Control transfer must use endpoints with no ping-pong attributes.
2. Isochronous transfers must use endpoints with ping-pong attributes.
3. Control transfers can be aborted using a stall handshake.

A status transaction is a special type of host-to-device transaction used only in a control transfer. The control transfer must be performed using endpoints with no ping-pong attributes. According to the control sequence (read or write), the USB device sends or receives a status transaction.

Figure 43-4. Control Read and Write Sequences



Notes:
1. During the Status IN stage, the host waits for a zero length packet (Data IN transaction with no data) from the device using DATA1 PID. Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev. 2.0,* for more information on the protocol layer.
2. During the Status OUT stage, the host emits a zero length packet to the device (Data OUT transaction with no data).

43.5.2 Handling Transactions with USB V2.0 Device Peripheral

43.5.2.1 Setup Transaction

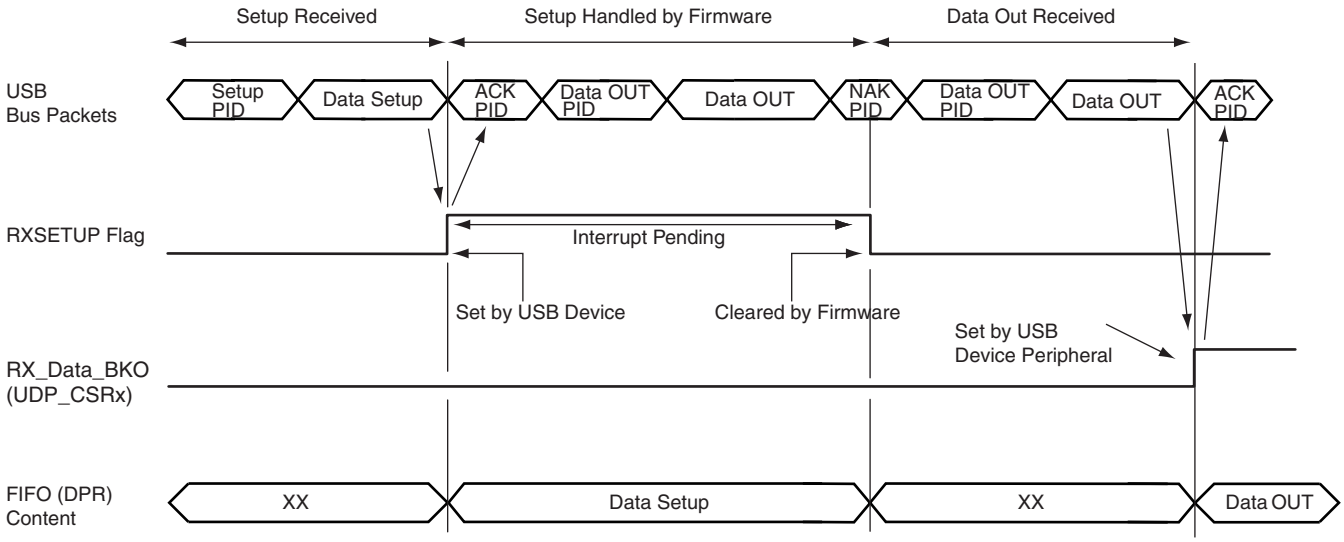
Setup is a special type of host-to-device transaction used during control transfers. Control transfers must be performed using endpoints with no ping-pong attributes. A setup transaction needs to be handled as soon as possible by the firmware. It is used to transmit requests from the host to the device. These requests are then handled by the USB device and may require more arguments. The arguments are sent to the device by a Data OUT transaction which follows the setup transaction. These requests may also return data. The data is carried out to the host by the next Data IN transaction which follows the setup transaction. A status transaction ends the control transfer.

When a setup transfer is received by the USB endpoint:

- The USB device automatically acknowledges the setup packet
- RXSETUP is set in the UDP_CSRx register
- An endpoint interrupt is generated while the RXSETUP is not cleared. This interrupt is carried out to the microcontroller if interrupts are enabled for this endpoint.

Thus, firmware must detect the RXSETUP polling the UDP_CSRx or catching an interrupt, read the setup packet in the FIFO, then clear the RXSETUP. RXSETUP cannot be cleared before the setup packet has been read in the FIFO. Otherwise, the USB device would accept the next Data OUT transfer and overwrite the setup packet in the FIFO.

Figure 43-5. Setup Transaction Followed by a Data OUT Transaction



43.5.2.2 *Data IN Transaction*

Data IN transactions are used in control, isochronous, bulk and interrupt transfers and conduct the transfer of data from the device to the host. Data IN transactions in isochronous transfer must be done using endpoints with ping-pong attributes.

43.5.2.3 *Using Endpoints Without Ping-pong Attributes*

To perform a Data IN transaction using a non ping-pong endpoint:

1. The application checks if it is possible to write in the FIFO by polling TXPKTRDY in the endpoint's UDP_CSRx register (TXPKTRDY must be cleared).
2. The application writes the first packet of data to be sent in the endpoint's FIFO, writing zero or more byte values in the endpoint's UDP_FDRx register,
3. The application notifies the USB peripheral it has finished by setting the TXPKTRDY in the endpoint's UDP_CSRx register.
4. The application is notified that the endpoint's FIFO has been released by the USB device when TXCOMP in the endpoint's UDP_CSRx register has been set. Then an interrupt for the corresponding endpoint is pending while TXCOMP is set.
5. The microcontroller writes the second packet of data to be sent in the endpoint's FIFO, writing zero or more byte values in the endpoint's UDP_FDRx register,
6. The microcontroller notifies the USB peripheral it has finished by setting the TXPK-TRDY in the endpoint's UDP_CSRx register.
7. The application clears the TXCOMP in the endpoint's UDP_CSRx.

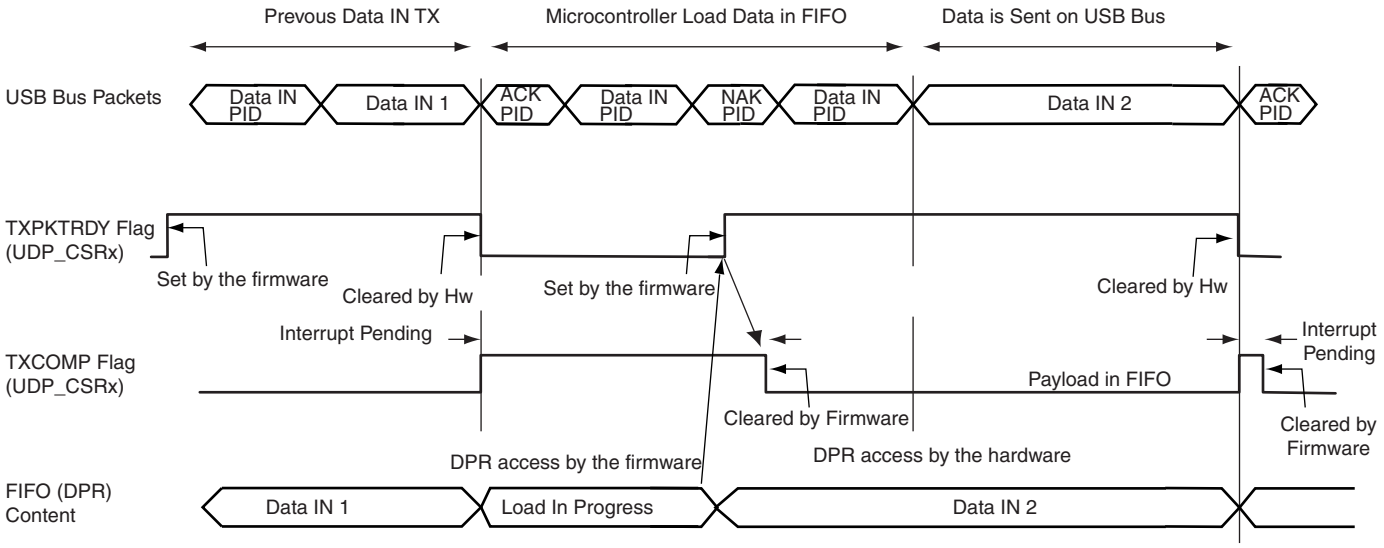
After the last packet has been sent, the application must clear TXCOMP once this has been set.

TXCOMP is set by the USB device when it has received an ACK PID signal for the Data IN packet. An interrupt is pending while TXCOMP is set.

Warning: TX_COMP must be cleared after TX_PKTRDY has been set.

Note: Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev 2.0,* for more information on the Data IN protocol layer.

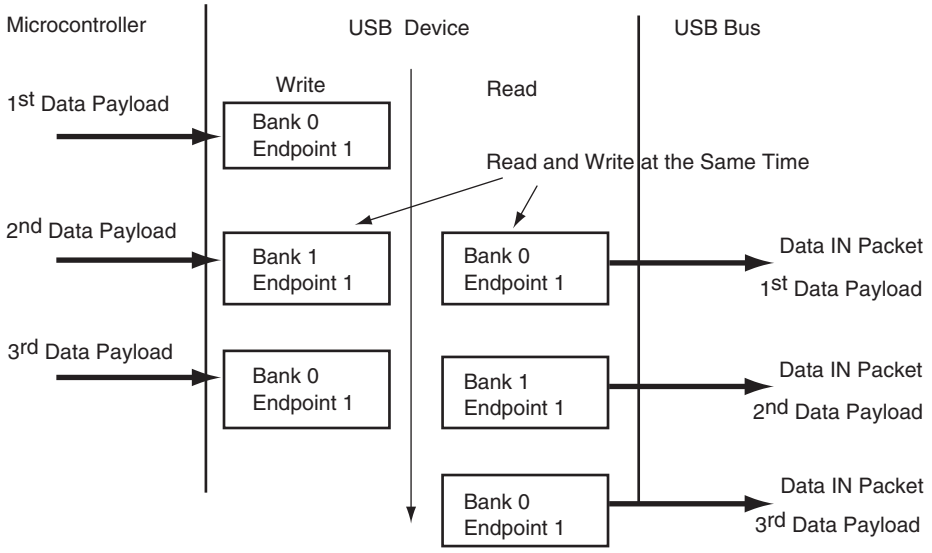
Figure 43-6. Data IN Transfer for Non Ping-pong Endpoint



43.5.2.4 Using Endpoints With Ping-pong Attribute

The use of an endpoint with ping-pong attributes is necessary during isochronous transfer. This also allows handling the maximum bandwidth defined in the USB specification during bulk transfer. To be able to guarantee a constant or the maximum bandwidth, the microcontroller must prepare the next data payload to be sent while the current one is being sent by the USB device. Thus two banks of memory are used. While one is available for the microcontroller, the other one is locked by the USB device.

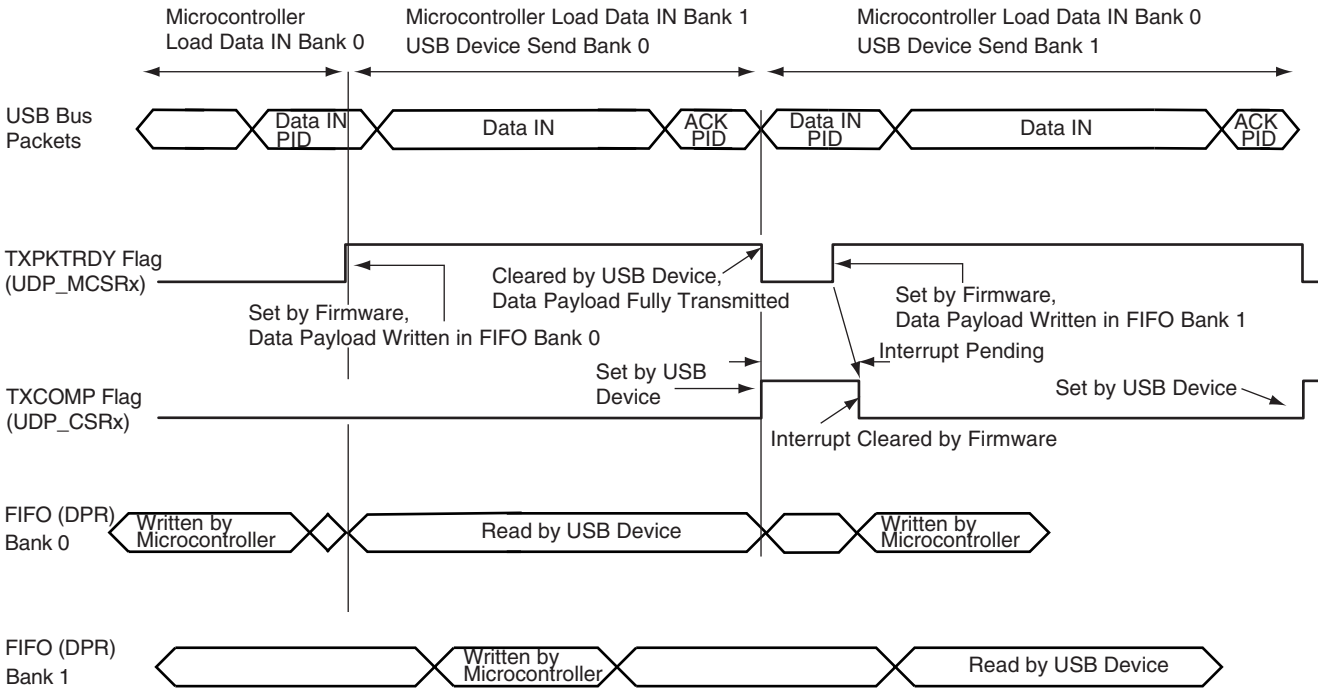
Figure 43-7. Bank Swapping Data IN Transfer for Ping-pong Endpoints



When using a ping-pong endpoint, the following procedures are required to perform Data IN transactions:

1. The microcontroller checks if it is possible to write in the FIFO by polling TXPKTRDY to be cleared in the endpoint's UDP_CSRx register.
2. The microcontroller writes the first data payload to be sent in the FIFO (Bank 0), writing zero or more byte values in the endpoint's UDP_FDRx register.
3. The microcontroller notifies the USB peripheral it has finished writing in Bank 0 of the FIFO by setting the TXPKTRDY in the endpoint's UDP_CSRx register.
4. Without waiting for TXPKTRDY to be cleared, the microcontroller writes the second data payload to be sent in the FIFO (Bank 1), writing zero or more byte values in the endpoint's UDP_FDRx register.
5. The microcontroller is notified that the first Bank has been released by the USB device when TXCOMP in the endpoint's UDP_CSRx register is set. An interrupt is pending while TXCOMP is being set.
6. Once the microcontroller has received TXCOMP for the first Bank, it notifies the USB device that it has prepared the second Bank to be sent, raising TXPKTRDY in the endpoint's UDP_CSRx register.
7. At this step, Bank 0 is available and the microcontroller can prepare a third data payload to be sent.

Figure 43-8. Data IN Transfer for Ping-pong Endpoint



Warning: There is software critical path due to the fact that once the second bank is filled, the driver has to wait for TX_COMP to set TX_PKTRDY. If the delay between receiving TX_COMP is set and TX_PKTRDY is set too long, some Data IN packets may be NACKed, reducing the bandwidth.

Warning: TX_COMP must be cleared after TX_PKTRDY has been set.

43.5.2.5 Data OUT Transaction

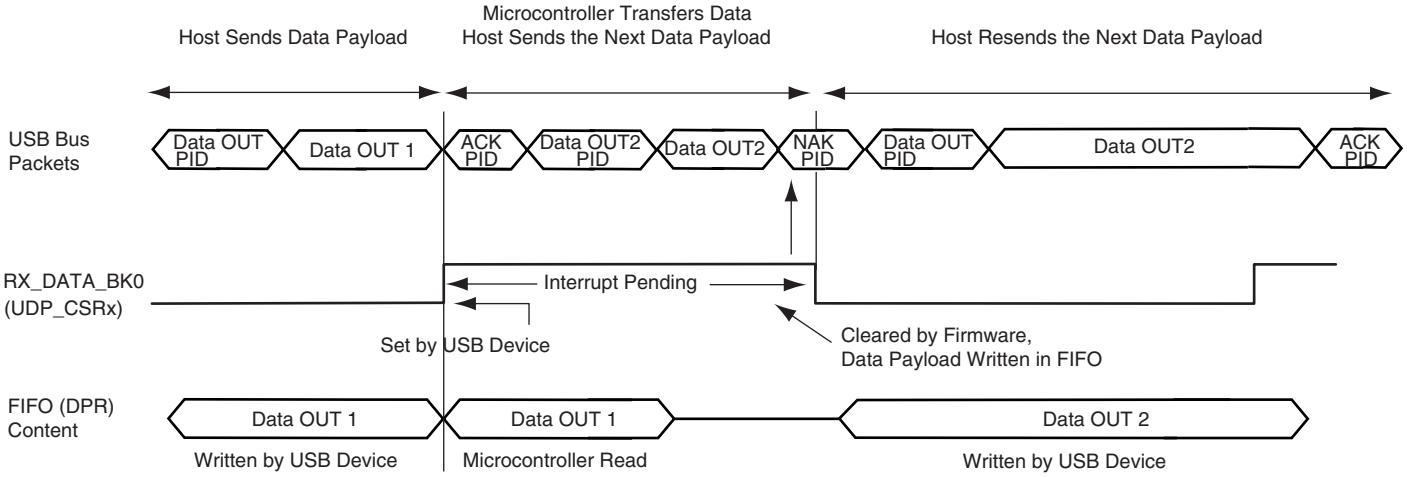
Data OUT transactions are used in control, isochronous, bulk and interrupt transfers and conduct the transfer of data from the host to the device. Data OUT transactions in isochronous transfers must be done using endpoints with ping-pong attributes.

43.5.2.6 Data OUT Transaction Without Ping-pong Attributes

To perform a Data OUT transaction, using a non ping-pong endpoint:

1. The host generates a Data OUT packet.
2. This packet is received by the USB device endpoint. While the FIFO associated to this endpoint is being used by the microcontroller, a NAK PID is returned to the host. Once the FIFO is available, data are written to the FIFO by the USB device and an ACK is automatically carried out to the host.
3. The microcontroller is notified that the USB device has received a data payload polling RX_DATA_BK0 in the endpoint's UDP_CSRx register. An interrupt is pending for this endpoint while RX_DATA_BK0 is set.
4. The number of bytes available in the FIFO is made available by reading RXBYTECNT in the endpoint's UDP_CSRx register.
5. The microcontroller carries out data received from the endpoint's memory to its memory. Data received is available by reading the endpoint's UDP_FDRx register.
6. The microcontroller notifies the USB device that it has finished the transfer by clearing RX_DATA_BK0 in the endpoint's UDP_CSRx register.
7. A new Data OUT packet can be accepted by the USB device.

Figure 43-9. Data OUT Transfer for Non Ping-pong Endpoints



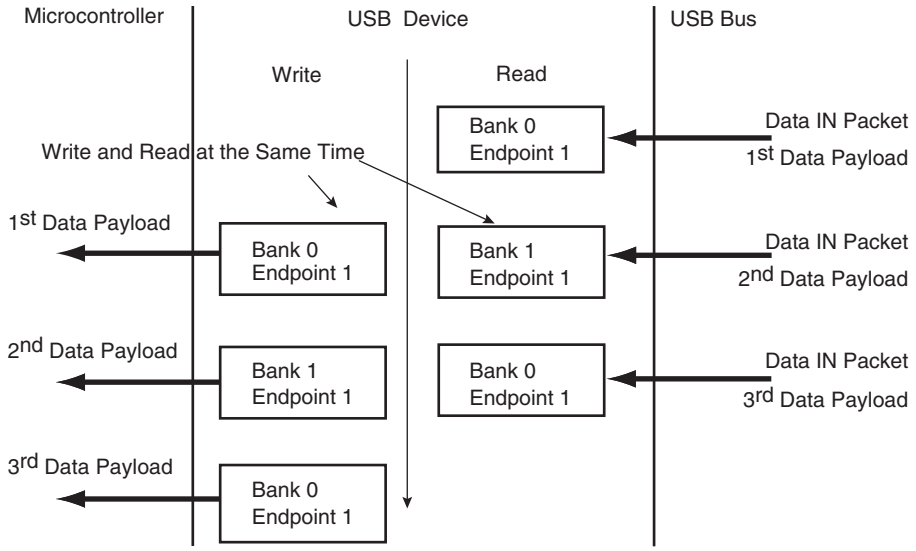
An interrupt is pending while the flag RX_DATA_BK0 is set. Memory transfer between the USB device, the FIFO and microcontroller memory can not be done after RX_DATA_BK0 has been cleared. Otherwise, the USB device would accept the next Data OUT transfer and overwrite the current Data OUT packet in the FIFO.

43.5.2.7 Using Endpoints With Ping-pong Attributes

During isochronous transfer, using an endpoint with ping-pong attributes is obligatory. To be able to guarantee a constant bandwidth, the microcontroller must read the previous data payload sent by the host, while the current data payload is received by the USB device. Thus two

banks of memory are used. While one is available for the microcontroller, the other one is locked by the USB device.

Figure 43-10. Bank Swapping in Data OUT Transfers for Ping-pong Endpoints

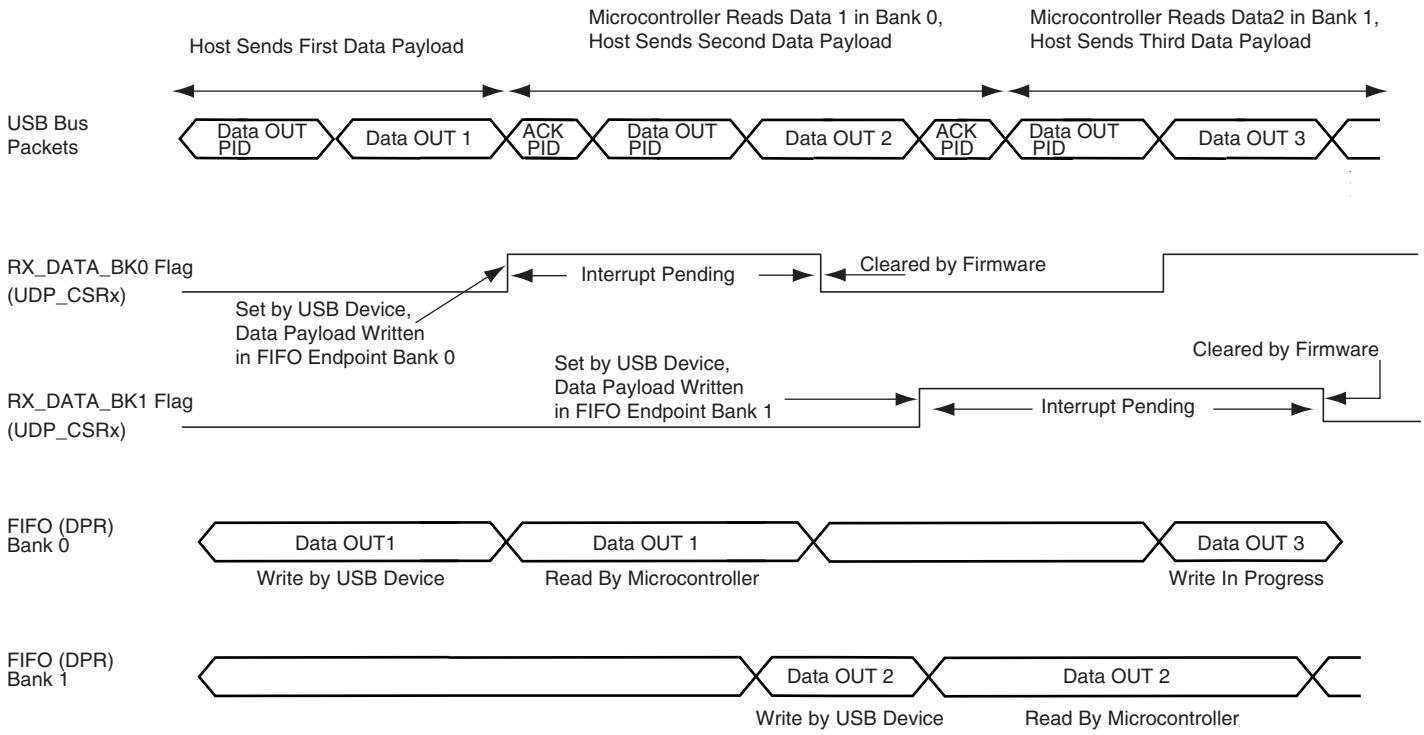


When using a ping-pong endpoint, the following procedures are required to perform Data OUT transactions:

1. The host generates a Data OUT packet.
2. This packet is received by the USB device endpoint. It is written in the endpoint's FIFO Bank 0.
3. The USB device sends an ACK PID packet to the host. The host can immediately send a second Data OUT packet. It is accepted by the device and copied to FIFO Bank 1.
4. The microcontroller is notified that the USB device has received a data payload, polling RX_DATA_BK0 in the endpoint's UDP_CSRx register. An interrupt is pending for this endpoint while RX_DATA_BK0 is set.
5. The number of bytes available in the FIFO is made available by reading RXBYTECNT in the endpoint's UDP_CSRx register.
6. The microcontroller transfers out data received from the endpoint's memory to the microcontroller's memory. Data received is made available by reading the endpoint's UDP_FDRx register.
7. The microcontroller notifies the USB peripheral device that it has finished the transfer by clearing RX_DATA_BK0 in the endpoint's UDP_CSRx register.
8. A third Data OUT packet can be accepted by the USB peripheral device and copied in the FIFO Bank 0.
9. If a second Data OUT packet has been received, the microcontroller is notified by the flag RX_DATA_BK1 set in the endpoint's UDP_CSRx register. An interrupt is pending for this endpoint while RX_DATA_BK1 is set.
10. The microcontroller transfers out data received from the endpoint's memory to the microcontroller's memory. Data received is available by reading the endpoint's UDP_FDRx register.

11. The microcontroller notifies the USB device it has finished the transfer by clearing RX_DATA_BK1 in the endpoint's UDP_CSRx register.
12. A fourth Data OUT packet can be accepted by the USB device and copied in the FIFO Bank 0.

Figure 43-11. Data OUT Transfer for Ping-pong Endpoint



Note: An interrupt is pending while the RX_DATA_BK0 or RX_DATA_BK1 flag is set.

Warning: When RX_DATA_BK0 and RX_DATA_BK1 are both set, there is no way to determine which one to clear first. Thus the software must keep an internal counter to be sure to clear alternatively RX_DATA_BK0 then RX_DATA_BK1. This situation may occur when the software application is busy elsewhere and the two banks are filled by the USB host. Once the application comes back to the USB driver, the two flags are set.

43.5.2.8 Stall Handshake

A stall handshake can be used in one of two distinct occasions. (For more information on the stall handshake, refer to Chapter 8 of the *Universal Serial Bus Specification, Rev 2.0*.)

- A functional stall is used when the halt feature associated with the endpoint is set. (Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev 2.0,* for more information on the halt feature.)
- To abort the current request, a protocol stall is used, but uniquely with control transfer.

The following procedure generates a stall packet:

1. The microcontroller sets the FORCESTALL flag in the UDP_CSRx endpoint's register.
2. The host receives the stall packet.

3. The microcontroller is notified that the device has sent the stall by polling the STALLSENT to be set. An endpoint interrupt is pending while STALLSENT is set. The microcontroller must clear STALLSENT to clear the interrupt.

When a setup transaction is received after a stall handshake, STALLSENT must be cleared in order to prevent interrupts due to STALLSENT being set.

Figure 43-12. Stall Handshake (Data IN Transfer)

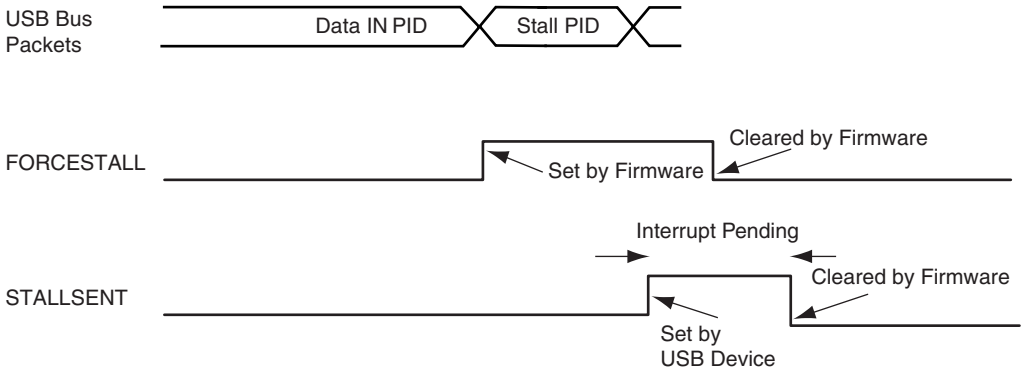
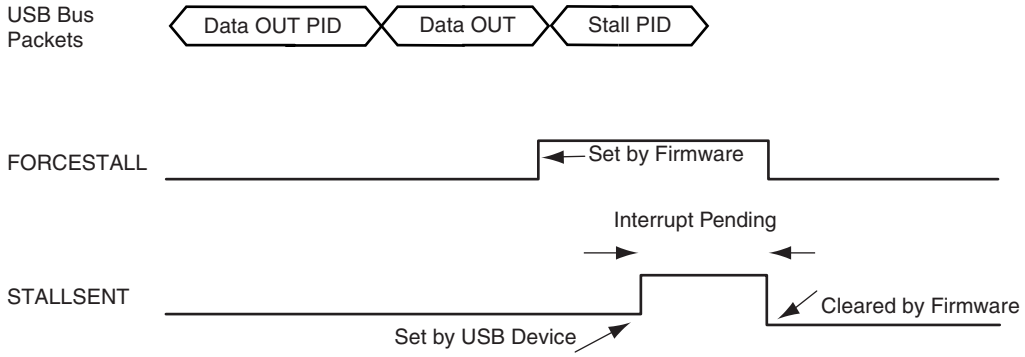


Figure 43-13. Stall Handshake (Data OUT Transfer)



43.5.2.9 Transmit Data Cancellation

Some endpoints have dual-banks whereas some endpoints have only one bank. The procedure to cancel transmission data held in these banks is described below.

To see the organization of dual-bank availability refer to Table 43-1 ”USB Endpoint Description”.

43.5.2.10 Endpoints ***Without*** Dual-Banks

There are two possibilities: In one case, TXPKTRDY field in UDP_CSR has already been set. In the other instance, TXPKTRDY is not set.

- TXPKTRDY is not set:
  - Reset the endpoint to clear the FIFO (pointers). (See, Section 43.6.9 ”UDP Reset Endpoint Register”.)
- TXPKTRDY has already been set:
  - Clear TXPKTRDY so that no packet is ready to be sent
  - Reset the endpoint to clear the FIFO (pointers). (See, Section 43.6.9 ”UDP Reset Endpoint Register”.)

43.5.2.11 Endpoints ***With*** Dual-Banks

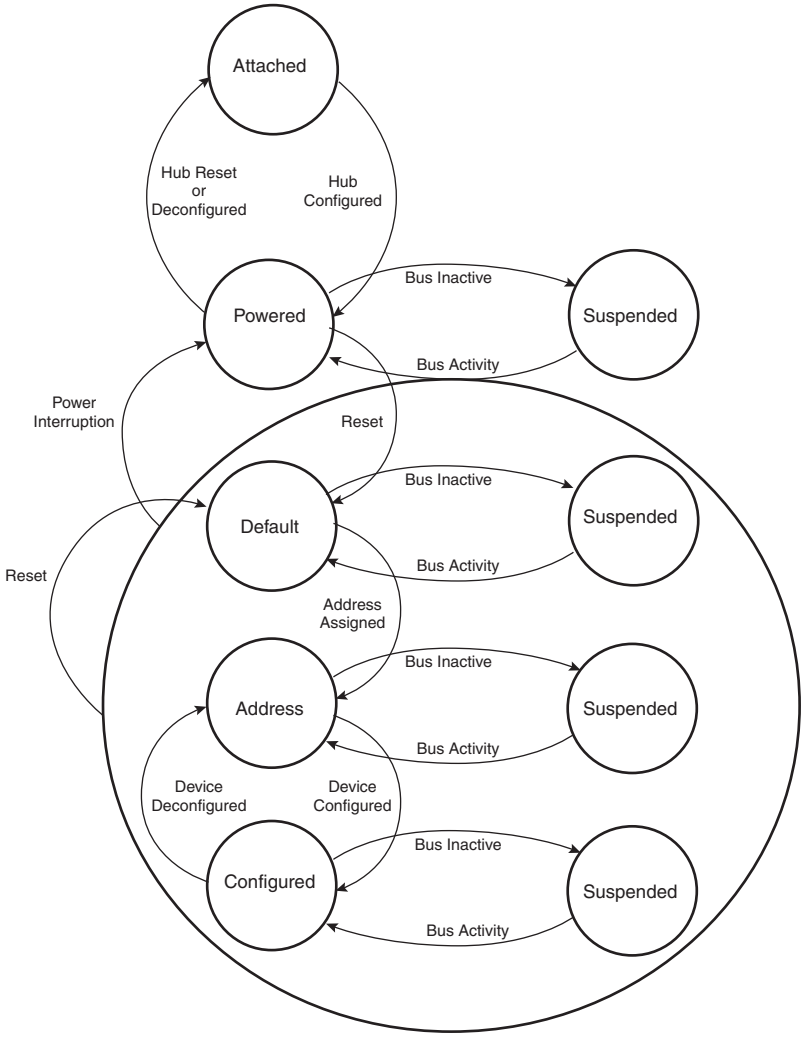
There are two possibilities: In one case, TXPKTRDY field in UDP_CSR has already been set. In the other instance, TXPKTRDY is not set.

- TXPKTRDY is not set:
  - Reset the endpoint to clear the FIFO (pointers). (See, Section 43.6.9 ”UDP Reset Endpoint Register”.)
- TXPKTRDY has already been set:
  - Clear TXPKTRDY and read it back until actually read at 0.
  - Set TXPKTRDY and read it back until actually read at 1.
  - Clear TXPKTRDY so that no packet is ready to be sent.
  - Reset the endpoint to clear the FIFO (pointers). (See, Section 43.6.9 ”UDP Reset Endpoint Register”.)

43.5.3 Controlling Device States

A USB device has several possible states. Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev 2.0*.

Figure 43-14. USB 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.

43.5.3.1 Not Powered State

Self powered devices can detect 5V VBUS using a PIO as described in the typical connection section. When the device is not connected to a host, device power consumption can be reduced by disabling MCK for the UDP, disabling UDPCK and disabling the transceiver. DDP and DDM lines are pulled down by 330 KΩ resistors.

43.5.3.2 Entering Attached State

When no device is connected, the USB DP and DM signals are tied to GND by 15 KΩ pull-down resistors integrated in the hub downstream ports. When a device is attached to a hub downstream port, the device connects a 1.5 KΩ pull-up resistor on DP. The USB bus line goes into IDLE state, DP is pulled up by the device 1.5 KΩ resistor to 3.3V and DM is pulled down by the 15 KΩ resistor of the host. To enable integrated pullup, the PUON bit in the UDP_TXVC register must be set.

Warning: To write to the UDP_TXVC register, MCK clock must be enabled on the UDP. This is done in the Power Management Controller.

After pullup connection, the device enters the powered state. In this state, the UDPCK and MCK must be enabled in the Power Management Controller. The transceiver can remain disabled.

43.5.3.3 From Powered State to Default State

After its connection to a USB host, the USB device waits for an end-of-bus reset. The unmaskable flag ENDBUSRES is set in the register UDP_ISR and an interrupt is triggered.

Once the ENDBUSRES interrupt has been triggered, the device enters Default State. In this state, the UDP software must:

- Enable the default endpoint, setting the EPEDS flag in the UDP_CSR[0] register and, optionally, enabling the interrupt for endpoint 0 by writing 1 to the UDP_IER register. The enumeration then begins by a control transfer.
- Configure the interrupt mask register which has been reset by the USB reset detection
- Enable the transceiver clearing the TXVDIS flag in the UDP_TXVC register.

In this state UDPCK and MCK must be enabled.

Warning: Each time an ENDBUSRES interrupt is triggered, the Interrupt Mask Register and UDP_CSR registers have been reset.

43.5.3.4 From Default State to Address State

After a set address standard device request, the USB host peripheral enters the address state.

Warning: Before the device enters in address state, it must achieve the Status IN transaction of the control transfer, i.e., the UDP device sets its new address once the TXCOMP flag in the UDP_CSR[0] register has been received and cleared.

To move to address state, the driver software sets the FADDEN flag in the UDP_GLB_STAT register, sets its new address, and sets the FEN bit in the UDP_FADDR register.

43.5.3.5 From Address State to Configured State

Once a valid Set Configuration standard request has been received and acknowledged, the device enables endpoints corresponding to the current configuration. This is done by setting the EPEDS and EPTYPE fields in the UDP_CSRx registers and, optionally, enabling corresponding interrupts in the UDP_IER register.

43.5.3.6 Entering in Suspend State

When a Suspend (no bus activity on the USB bus) is detected, the RXSUSP signal in the UDP_ISR register is set. This triggers an interrupt if the corresponding bit is set in the UDP_IMR register.This flag is cleared by writing to the UDP_ICR register. Then the device enters Suspend Mode.

In this state bus powered devices must drain less than 500uA from the 5V VBUS. As an example, the microcontroller switches to slow clock, disables the PLL and main oscillator, and goes into Idle Mode. It may also switch off other devices on the board.

The USB device peripheral clocks can be switched off. Resume event is asynchronously detected. MCK and UDPCK can be switched off in the Power Management controller and the USB transceiver can be disabled by setting the TXVDIS field in the UDP_TXVC register.

Warning: Read, write operations to the UDP registers are allowed only if MCK is enabled for the UDP peripheral. Switching off MCK for the UDP peripheral must be one of the last operations after writing to the UDP_TXVC and acknowledging the RXSUSP.

43.5.3.7 Receiving a Host Resume

In suspend mode, a resume event on the USB bus line is detected asynchronously, transceiver and clocks are disabled (however the pullup shall not be removed).

Once the resume is detected on the bus, the WAKEUP signal in the UDP_ISR is set. It may generate an interrupt if the corresponding bit in the UDP_IMR register is set. This interrupt may be used to wake up the core, enable PLL and main oscillators and configure clocks.

Warning: Read, write operations to the UDP registers are allowed only if MCK is enabled for the UDP peripheral. MCK for the UDP must be enabled before clearing the WAKEUP bit in the UDP_ICR register and clearing TXVDIS in the UDP_TXVC register.

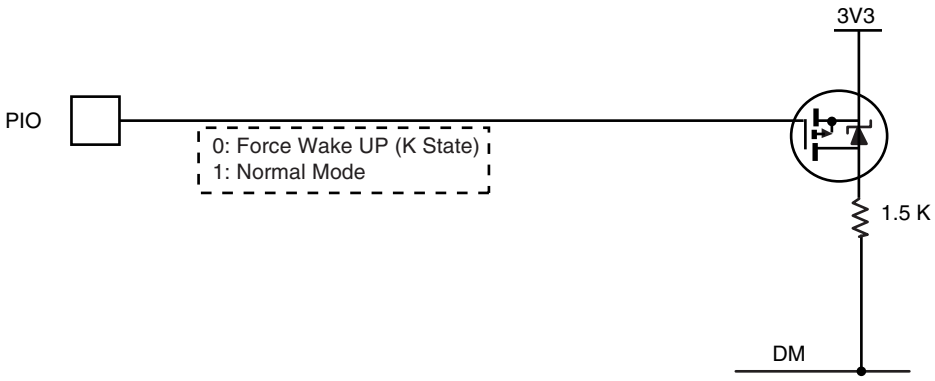
43.5.3.8 Sending a Device Remote Wakeup

In Suspend state it is possible to wake up the host sending an external resume.

- The device must wait at least 5 ms after being entered in suspend before sending an external resume.
- The device has 10 ms from the moment it starts to drain current and it forces a K state to resume the host.
- The device must force a K state from 1 to 15 ms to resume the host

To force a K state to the bus (DM at 3.3V and DP tied to GND), it is possible to use a transistor to connect a pullup on DM. The K state is obtained by disabling the pullup on DP and enabling the pullup on DM. This should be under the control of the application.

Figure 43-15. Board Schematic to Drive a K State



43.6 USB Device Port (UDP) User Interface

WARNING: The UDP peripheral clock in the Power Management Controller (PMC) must be enabled before any read/write operations to the UDP registers, including the UDP_TXVC register.

Table 43-4. Register Mapping

Offset	Register	Name	Access	Reset
0x000	Frame Number Register	UDP_FRM_NUM	Read-only	0x0000_0000
0x004	Global State Register	UDP_GLB_STAT	Read-write	0x0000_0000
0x008	Function Address Register	UDP_FADDR	Read-write	0x0000_0100
0x00C	Reserved	–	–	–
0x010	Interrupt Enable Register	UDP_IER	Write-only	
0x014	Interrupt Disable Register	UDP_IDR	Write-only	
0x018	Interrupt Mask Register	UDP_IMR	Read-only	0x0000_1200
0x01C	Interrupt Status Register	UDP_ISR	Read-only	–(1)
0x020	Interrupt Clear Register	UDP_ICR	Write-only	
0x024	Reserved	–	–	–
0x028	Reset Endpoint Register	UDP_RST_EP	Read-write	0x0000_0000
0x02C	Reserved	–	–	–
0x030 + 0x4 * (ept_num - 1)	Endpoint Control and Status Register	UDP_CSR	Read-write	0x0000_0000
0x050 + 0x4 * (ept_num - 1)	Endpoint FIFO Data Register	UDP_FDR	Read-write	0x0000_0000
0x070	Reserved	–	–	–
0x074	Transceiver Control Register(2)	UDP_TXVC	Read-write	0x0000_0000
0x078 - 0xFC	Reserved	–	–	–

Notes: 1. Reset values are not defined for UDP_ISR.
2. See Warning above the "Register Mapping" on this page.

43.6.1 UDP Frame Number Register

Register Name:UDP_FRM_NUM

Address: 0xFFF78000

Access Type: Read-only

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

- **FRM_NUM[10:0]: Frame Number as Defined in the Packet Field Formats**

This 11-bit value is incremented by the host on a per frame basis. This value is updated at each start of frame.

Value Updated at the SOF_EOP (Start of Frame End of Packet).

- **FRM_ERR: Frame Error**

This bit is set at SOF_EOP when the SOF packet is received containing an error.

This bit is reset upon receipt of SOF_PID.

- **FRM_OK: Frame OK**

This bit is set at SOF_EOP when the SOF packet is received without any error.

This bit is reset upon receipt of SOF_PID (Packet Identification).

In the Interrupt Status Register, the SOF interrupt is updated upon receiving SOF_PID. This bit is set without waiting for EOP.

Note: In the 8-bit Register Interface, FRM_OK is bit 4 of FRM_NUM_H and FRM_ERR is bit 3 of FRM_NUM_L.

43.6.2 UDP Global State Register

Register Name: UDP_GLB_STAT

Address: 0xFFF78004

Access Type: Read-write

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

This register is used to get and set the device state as specified in Chapter 9 of the *USB Serial Bus Specification, Rev.2.0*.

- **FADDEN: Function Address Enable**

Read:

0 = Device is not in address state.

1 = Device is in address state.

Write:

0 = No effect, only a reset can bring back a device to the default state.

1 = Sets device in address state. This occurs after a successful Set Address request. Beforehand, the UDP_FADDR register must have been initialized with Set Address parameters. Set Address must complete the Status Stage before setting FADDEN. Refer to chapter 9 of the *Universal Serial Bus Specification, Rev. 2.0* for more details.

- **CONFG: Configured**

Read:

0 = Device is not in configured state.

1 = Device is in configured state.

Write:

0 = Sets device in a non configured state

1 = Sets device in configured state.

The device is set in configured state when it is in address state and receives a successful Set Configuration request. Refer to Chapter 9 of the *Universal Serial Bus Specification, Rev. 2.0* for more details.

43.6.3 UDP Function Address Register

Register Name: UDP_FADDR

Address: 0xFFF78008

Access Type: Read-write

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

- **FADD[6:0]: Function Address Value**

The Function Address Value must be programmed by firmware once the device receives a set address request from the host, and has achieved the status stage of the no-data control sequence. Refer to the *Universal Serial Bus Specification, Rev. 2.0* for more information. After power up or reset, the function address value is set to 0.

- **FEN: Function Enable**

Read:

0 = Function endpoint disabled.

1 = Function endpoint enabled.

Write:

0 = Disables function endpoint.

1 = Default value.

The Function Enable bit (FEN) allows the microcontroller to enable or disable the function endpoints. The microcontroller sets this bit after receipt of a reset from the host. Once this bit is set, the USB device is able to accept and transfer data packets from and to the host.

43.6.4 UDP Interrupt Enable Register

Register Name: UDP_IER

Address: 0xFFF78010

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	WAKEUP	–	SOFINT	–	RXRSM	RXSUSP
7	6	5	4	3	2	1	0
		EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: Enable Endpoint 0 Interrupt**
- **EP1INT: Enable Endpoint 1 Interrupt**
- **EP2INT: Enable Endpoint 2Interrupt**
- **EP3INT: Enable Endpoint 3 Interrupt**
- **EP4INT: Enable Endpoint 4 Interrupt**
- **EP5INT: Enable Endpoint 5 Interrupt**

0 = No effect.

1 = Enables corresponding Endpoint Interrupt.

- **RXSUSP: Enable UDP Suspend Interrupt**

0 = No effect.

1 = Enables UDP Suspend Interrupt.

- **RXRSM: Enable UDP Resume Interrupt**

0 = No effect.

1 = Enables UDP Resume Interrupt.

- **SOFINT: Enable Start Of Frame Interrupt**

0 = No effect.

1 = Enables Start Of Frame Interrupt.

- **WAKEUP: Enable UDP bus Wakeup Interrupt**

0 = No effect.

1 = Enables USB bus Interrupt.

43.6.5 UDP Interrupt Disable Register

Register Name: UDP_IDR

Address: 0xFFF78014

Access Type: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	WAKEUP	–	SOFINT	–	RXRSM	RXSUSP
7	6	5	4	3	2	1	0
		EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: Disable Endpoint 0 Interrupt**
- **EP1INT: Disable Endpoint 1 Interrupt**
- **EP2INT: Disable Endpoint 2 Interrupt**
- **EP3INT: Disable Endpoint 3 Interrupt**
- **EP4INT: Disable Endpoint 4 Interrupt**
- **EP5INT: Disable Endpoint 5 Interrupt**

0 = No effect.

1 = Disables corresponding Endpoint Interrupt.

- **RXSUSP: Disable UDP Suspend Interrupt**

0 = No effect.

1 = Disables UDP Suspend Interrupt.

- **RXRSM: Disable UDP Resume Interrupt**

0 = No effect.

1 = Disables UDP Resume Interrupt.

- **SOFINT: Disable Start Of Frame Interrupt**

0 = No effect.

1 = Disables Start Of Frame Interrupt

- **WAKEUP: Disable USB Bus Interrupt**

0 = No effect.

1 = Disables USB Bus Wakeup Interrupt.

43.6.6 UDP Interrupt Mask Register

Register Name: UDP_IMR

Address: 0xFFF78018

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	WAKEUP	BIT12	SOFINT	–	RXRSM	RXSUSP
7	6	5	4	3	2	1	0
		EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: Mask Endpoint 0 Interrupt**
- **EP1INT: Mask Endpoint 1 Interrupt**
- **EP2INT: Mask Endpoint 2 Interrupt**
- **EP3INT: Mask Endpoint 3 Interrupt**
- **EP4INT: Mask Endpoint 4 Interrupt**
- **EP5INT: Mask Endpoint 5 Interrupt**

0 = Corresponding Endpoint Interrupt is disabled.

1 = Corresponding Endpoint Interrupt is enabled.

- **RXSUSP: Mask UDP Suspend Interrupt**

0 = UDP Suspend Interrupt is disabled.

1 = UDP Suspend Interrupt is enabled.

- **RXRSM: Mask UDP Resume Interrupt.**

0 = UDP Resume Interrupt is disabled.

1 = UDP Resume Interrupt is enabled.

- **SOFINT: Mask Start Of Frame Interrupt**

0 = Start of Frame Interrupt is disabled.

1 = Start of Frame Interrupt is enabled.

- **BIT12: UDP_IMR Bit 12**

Bit 12 of UDP_IMR cannot be masked and is always read at 1.

- **WAKEUP: USB Bus WAKEUP Interrupt**

0 = USB Bus Wakeup Interrupt is disabled.

1 = USB Bus Wakeup Interrupt is enabled.

Note: When the USB block is in suspend mode, the application may power down the USB logic. In this case, any USB HOST resume request that is made must be taken into account and, thus, the reset value of the RXRSM bit of the register UDP_IMR is enabled.

43.6.7 UDP Interrupt Status Register

Register Name: UDP_ISR

Address: 0xFFF7801C

Access Type: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	WAKEUP	ENDBUSRES	SOFINT	–	RXRSM	RXSUSP
7	6	5	4	3	2	1	0
		EP5INT	EP4INT	EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: Endpoint 0 Interrupt Status**
- **EP1INT: Endpoint 1 Interrupt Status**
- **EP2INT: Endpoint 2 Interrupt Status**
- **EP3INT: Endpoint 3 Interrupt Status**
- **EP4INT: Endpoint 4 Interrupt Status**
- **EP5INT: Endpoint 5 Interrupt Status**

0 = No Endpoint0 Interrupt pending.

1 = Endpoint0 Interrupt has been raised.

Several signals can generate this interrupt. The reason can be found by reading UDP_CSR0:

RXSETUP set to 1

RX_DATA_BK0 set to 1

RX_DATA_BK1 set to 1

TXCOMP set to 1

STALLSENT set to 1

EP0INT is a sticky bit. Interrupt remains valid until EP0INT is cleared by writing in the corresponding UDP_CSR0 bit.

- **RXSUSP: UDP Suspend Interrupt Status**

0 = No UDP Suspend Interrupt pending.

1 = UDP Suspend Interrupt has been raised.

The USB device sets this bit when it detects no activity for 3ms. The USB device enters Suspend mode.

- **RXRSM: UDP Resume Interrupt Status**

0 = No UDP Resume Interrupt pending.

1 =UDP Resume Interrupt has been raised.

The USB device sets this bit when a UDP resume signal is detected at its port.

After reset, the state of this bit is undefined, the application must clear this bit by setting the RXRSM flag in the UDP_ICR register.

- **SOFINT: Start of Frame Interrupt Status**

0 = No Start of Frame Interrupt pending.

1 = Start of Frame Interrupt has been raised.

This interrupt is raised each time a SOF token has been detected. It can be used as a synchronization signal by using isochronous endpoints.

- **ENDBUSRES: End of BUS Reset Interrupt Status**

0 = No End of Bus Reset Interrupt pending.

1 = End of Bus Reset Interrupt has been raised.

This interrupt is raised at the end of a UDP reset sequence. The USB device must prepare to receive requests on the endpoint 0. The host starts the enumeration, then performs the configuration.

- **WAKEUP: UDP Resume Interrupt Status**

0 = No Wakeup Interrupt pending.

1 = A Wakeup Interrupt (USB Host Sent a RESUME or RESET) occurred since the last clear.

After reset the state of this bit is undefined, the application must clear this bit by setting the WAKEUP flag in the UDP_ICR register.

43.6.8 UDP Interrupt Clear Register

Register Name: UDP_ICR

Address: 0xFFF78020

Access Type: Write-only

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

- **RXSUSP: Clear UDP Suspend Interrupt**

0 = No effect.

1 = Clears UDP Suspend Interrupt.

- **RXRSM: Clear UDP Resume Interrupt**

0 = No effect.

1 = Clears UDP Resume Interrupt.

- **SOFINT: Clear Start Of Frame Interrupt**

0 = No effect.

1 = Clears Start Of Frame Interrupt.

- **ENDBUSRES: Clear End of Bus Reset Interrupt**

0 = No effect.

1 = Clears End of Bus Reset Interrupt.

- **WAKEUP: Clear Wakeup Interrupt**

0 = No effect.

1 = Clears Wakeup Interrupt.

43.6.9 UDP Reset Endpoint Register

Register Name: UDP_RST_EP

Address: 0xFFF78028

Access Type: Read-write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
		EP5	EP4	EP3	EP2	EP1	EP0

- **EP0: Reset Endpoint 0**
- **EP1: Reset Endpoint 1**
- **EP2: Reset Endpoint 2**
- **EP3: Reset Endpoint 3**
- **EP4: Reset Endpoint 4**
- **EP5: Reset Endpoint 5**

This flag is used to reset the FIFO associated with the endpoint and the bit RXBYTECOUNT in the register UDP_CSRx.It also resets the data toggle to DATA0. It is useful after removing a HALT condition on a BULK endpoint. Refer to Chapter 5.8.5 in the *USB Serial Bus Specification, Rev.2.0*.

Warning: This flag must be cleared at the end of the reset. It does not clear UDP_CSRx flags.

0 = No reset.

1 = Forces the corresponding endpoint FIF0 pointers to 0, therefore RXBYTECNT field is read at 0 in UDP_CSRx register.

Resetting the endpoint is a two-step operation:

1. Set the corresponding EPx field.
2. Clear the corresponding EPx field.

43.6.10 UDP Endpoint Control and Status Register

Register Name:UDP_CSRx [x = 0..5]

Address: 0xFFF7802C

Access Type: Read-write

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

23	22	21	20	19	18	17	16
RXBYTECNT							

15	14	13	12	11	10	9	8
EPEDS	–	–	–	DTGLE	EPTYPE		

7	6	5	4	3	2	1	0
DIR	RX_DATA_BK1	FORCE STALL	TXPKTRDY	STALLSENT ISOERROR	RXSETUP	RX_DATA_BK0	TXCOMP

WARNING: Due to synchronization between MCK and UDPCK, the software application must wait for the end of the write operation before executing another write by polling the bits which must be set/cleared.

```
//! Clear flags of UDP UDP_CSR register and waits for synchronization
#define Udp_ep_clr_flag(pInterface, endpoint, flags) { \
   pInterface->UDP_CSR[endpoint] &= ~(flags); \
   while ( (pInterface->UDP_CSR[endpoint] & (flags)) == (flags) ); \
   }
//! Set flags of UDP UDP_CSR register and waits for synchronization
#define Udp_ep_set_flag(pInterface, endpoint, flags) { \
   pInterface->UDP_CSR[endpoint] |= (flags); \
   while ( (pInterface->UDP_CSR[endpoint] & (flags)) != (flags) ); \
   }
```

Note: In a preemptive environment, set or clear the flag and wait for a time of 1 UDPCK clock cycle and 1peripheral clock cycle. However, RX_DATA_BLK0, TXPKTRDY, RX_DATA_BK1 require wait times of 3 UDPCK clock cycles and 3 peripheral clock cycles before accessing DPR.

- **TXCOMP: Generates an IN Packet with Data Previously Written in the DPR**

This flag generates an interrupt while it is set to one.

Write (Cleared by the firmware):

0 = Clear the flag, clear the interrupt.

1 = No effect.

Read (Set by the USB peripheral):

0 = Data IN transaction has not been acknowledged by the Host.

1 = Data IN transaction is achieved, acknowledged by the Host.

After having issued a Data IN transaction setting TXPKTRDY, the device firmware waits for TXCOMP to be sure that the host has acknowledged the transaction.

- **RX_DATA_BK0: Receive Data Bank 0**

This flag generates an interrupt while it is set to one.

Write (Cleared by the firmware):

0 = Notify USB peripheral device that data have been read in the FIFO's Bank 0.

1 = To leave the read value unchanged.

Read (Set by the USB peripheral):

0 = No data packet has been received in the FIFO's Bank 0.

1 = A data packet has been received, it has been stored in the FIFO's Bank 0.

When the device firmware has polled this bit or has been interrupted by this signal, it must transfer data from the FIFO to the microcontroller memory. The number of bytes received is available in RXBYTCENT field. Bank 0 FIFO values are read through the UDP_FDRx register. Once a transfer is done, the device firmware must release Bank 0 to the USB peripheral device by clearing RX_DATA_BK0.

After setting or clearing this bit, a wait time of 3 UDPCK clock cycles and 3 peripheral clock cycles is required before accessing DPR.

- **RXSETUP: Received Setup**

This flag generates an interrupt while it is set to one.

Read:

0 = No setup packet available.

1 = A setup data packet has been sent by the host and is available in the FIFO.

Write:

0 = Device firmware notifies the USB peripheral device that it has read the setup data in the FIFO.

1 = No effect.

This flag is used to notify the USB device firmware that a valid Setup data packet has been sent by the host and successfully received by the USB device. The USB device firmware may transfer Setup data from the FIFO by reading the UDP_FDRx register to the microcontroller memory. Once a transfer has been done, RXSETUP must be cleared by the device firmware.

Ensuing Data OUT transaction is not accepted while RXSETUP is set.

- **STALLSENT: Stall Sent (Control, Bulk Interrupt Endpoints)/ISOERROR (Isochronous Endpoints)**

This flag generates an interrupt while it is set to one.

STALLSENT: This ends a STALL handshake.

Read:

0 = The host has not acknowledged a STALL.

1 = Host has acknowledged the stall.

Write:

0 = Resets the STALLSENT flag, clears the interrupt.

1 = No effect.

This is mandatory for the device firmware to clear this flag. Otherwise the interrupt remains.

Refer to chapters 8.4.5 and 9.4.5 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the STALL handshake.

ISOERROR: A CRC error has been detected in an isochronous transfer.

Read:

0 = No error in the previous isochronous transfer.

1 = CRC error has been detected, data available in the FIFO are corrupted.

Write:

0 = Resets the ISOERROR flag, clears the interrupt.

1 = No effect.

- **TXPKTRDY: Transmit Packet Ready**

This flag is cleared by the USB device.

This flag is set by the USB device firmware.

Read:

0 = There is no data to send.

1 = The data is waiting to be sent upon reception of token IN.

Write:

0 = Can be used in the procedure to cancel transmission data. (See, Section 43.5.2.9 “Transmit Data Cancellation” on page 873)

1 = A new data payload has been written in the FIFO by the firmware and is ready to be sent.

This flag is used to generate a Data IN transaction (device to host). Device firmware checks that it can write a data payload in the FIFO, checking that TXPKTRDY is cleared. Transfer to the FIFO is done by writing in the UDP_FDRx register. Once the data payload has been transferred to the FIFO, the firmware notifies the USB device setting TXPKTRDY to one. USB bus transactions can start. TXCOMP is set once the data payload has been received by the host.

After setting or clearing this bit, a wait time of 3 UDPCK clock cycles and 3 peripheral clock cycles is required before accessing DPR.

- **FORCESTALL: Force Stall (used by Control, Bulk and Isochronous Endpoints)**

Read:

0 = Normal state.

1 = Stall state.

Write:

0 = Return to normal state.

1 = Send STALL to the host.

Refer to chapters 8.4.5 and 9.4.5 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the STALL handshake.

Control endpoints: During the data stage and status stage, this bit indicates that the microcontroller cannot complete the request.

Bulk and interrupt endpoints: This bit notifies the host that the endpoint is halted.

The host acknowledges the STALL, device firmware is notified by the STALLSENT flag.

- **RX_DATA_BK1: Receive Data Bank 1 (only used by endpoints with ping-pong attributes)**

This flag generates an interrupt while it is set to one.

Write (Cleared by the firmware):

0 = Notifies USB device that data have been read in the FIFO's Bank 1.

1 = To leave the read value unchanged.

Read (Set by the USB peripheral):

0 = No data packet has been received in the FIFO's Bank 1.

1 = A data packet has been received, it has been stored in FIFO's Bank 1.

When the device firmware has polled this bit or has been interrupted by this signal, it must transfer data from the FIFO to microcontroller memory. The number of bytes received is available in RXBYTECNT field. Bank 1 FIFO values are read through UDP_FDRx register. Once a transfer is done, the device firmware must release Bank 1 to the USB device by clearing RX_DATA_BK1.

After setting or clearing this bit, a wait time of 3 UDPCK clock cycles and 3 peripheral clock cycles is required before accessing DPR.

- **DIR: Transfer Direction (only available for control endpoints)**

Read-write

0 = Allows Data OUT transactions in the control data stage.

1 = Enables Data IN transactions in the control data stage.

Refer to Chapter 8.5.3 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on the control data stage.

This bit must be set before UDP_CSRx/RXSETUP is cleared at the end of the setup stage. According to the request sent in the setup data packet, the data stage is either a device to host (DIR = 1) or host to device (DIR = 0) data transfer. It is not necessary to check this bit to reverse direction for the status stage.

- **EPTYPE[2:0]: Endpoint Type**

Read-write

000	Control
001	Isochronous OUT
101	Isochronous IN
010	Bulk OUT
110	Bulk IN
011	Interrupt OUT
111	Interrupt IN

- **DTGLE: Data Toggle**

Read-only

0 = Identifies DATA0 packet.

1 = Identifies DATA1 packet.

Refer to Chapter 8 of the *Universal Serial Bus Specification, Rev. 2.0* for more information on DATA0, DATA1 packet definitions.

- **EPEDS: Endpoint Enable Disable**

Read:

0 = Endpoint disabled.

1 = Endpoint enabled.

Write:

0 = Disables endpoint.

1 = Enables endpoint.

Control endpoints are always enabled. Reading or writing this field has no effect on control endpoints.

Note: After reset, all endpoints are configured as control endpoints (zero).

- **RXBYTECNT[10:0]: Number of Bytes Available in the FIFO**

Read-only

When the host sends a data packet to the device, the USB device stores the data in the FIFO and notifies the microcontroller. The microcontroller can load the data from the FIFO by reading RXBYTECENT bytes in the UDP_FDRx register.

43.6.11 UDP FIFO Data Register

Register Name: UDP_FDRx [x = 0..5]

Address: 0xFFF7804C

Access Type: Read-write

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
FIFO_DATA							

- **FIFO_DATA[7:0]: FIFO Data Value**

The microcontroller can push or pop values in the FIFO through this register.

RXBYTECNT in the corresponding UDP_CSRx register is the number of bytes to be read from the FIFO (sent by the host).

The maximum number of bytes to write is fixed by the Max Packet Size in the Standard Endpoint Descriptor. It can not be more than the physical memory size associated to the endpoint. Refer to the *Universal Serial Bus Specification, Rev. 2.0* for more information.

43.6.12 UDP Transceiver Control Register

Register Name: UDP_TXVC

Address: 0xFFF78074

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
–	–	–	–	–	–	PUON	TXVDIS
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

WARNING: The UDP peripheral clock in the Power Management Controller (PMC) must be enabled before any read/write operations to the UDP registers including the UDP_TXVC register.

- **TXVDIS: Transceiver Disable**

When UDP is disabled, power consumption can be reduced significantly by disabling the embedded transceiver. This can be done by setting TXVDIS field.

To enable the transceiver, TXVDIS must be cleared.

- **PUON: Pullup On**

0: The 1.5KΩ integrated pullup on DP is disconnected.

1: The 1.5 KΩ integrated pullup on DP is connected.

NOTE: If the USB pullup is not connected on DP, the user should not write in any UDP register other than the UDP_TXVC register. This is because if DP and DM are floating at 0, or pulled down, then SE0 is received by the device with the consequence of a USB Reset.

44. LCD Controller (LCDC)

44.1 Overview

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.

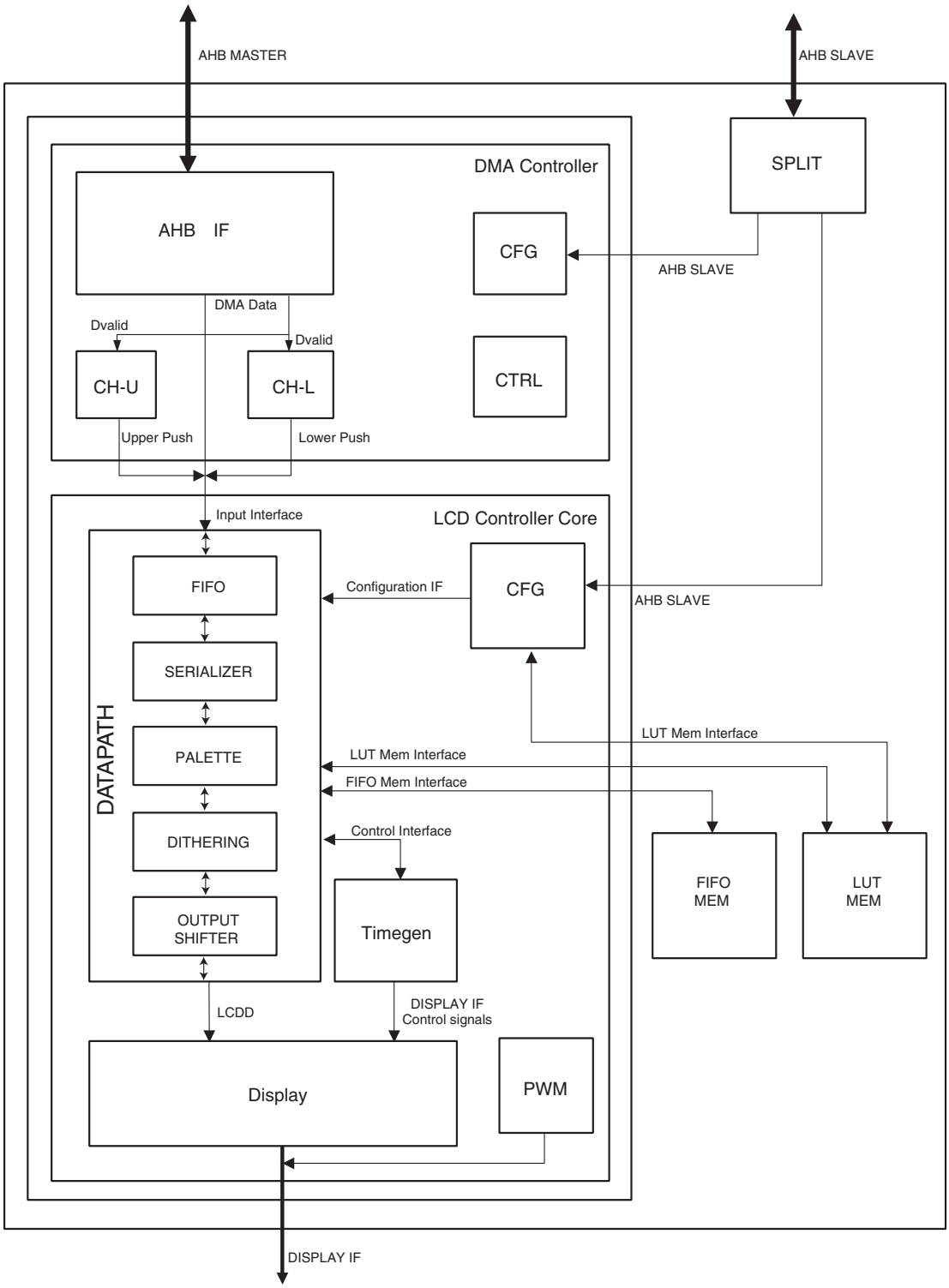
The LCD Controller has a display input buffer (FIFO) to allow a flexible connection of the external AHB master interface, and a lookup table to allow palletized display configurations.

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.

44.2 Block Diagram

Figure 44-1. LCD Macrocell Block Diagram



44.3 I/O Lines Description

Table 44-1. 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

44.4 Product Dependencies

44.4.1 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.

44.4.2 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).

44.4.3 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.

44.5 Functional Description

The LCD Controller consists of two main blocks (Figure 44-1 on page 898), 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.

44.5.1 DMA Controller

44.5.1.1 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 925.

44.5.1.2 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 933.

44.5.1.3 Channel-U

This block stores the base address and the number of words transferred for this channel (frame in single scan mode and Upper Panel in dual scan mode) since the beginning of the frame. It also generates the end of frame signal.

It has two pointers, the base address and the number of words to transfer. When the module receives a new_frame signal, it reloads the number of words to transfer pointer with the size of the frame/panel. When the module receives the new_frame signal, it also reloads the base address with the base address programmed by the host.

The size of the frame/panel can be programmed in the FRMSIZE field of the DMAFRMCFG Register. This size is calculated as follows:

$$\text{Frame_size} = \left\lceil \frac{\text{X_size}*\text{Y_size}}{32} \right\rceil$$

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 922), 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.

44.5.1.4 Channel-L

This block has the same functionality as Channel-U, but for the Lower Panel in dual scan mode only.

44.5.1.5 Control

This block receives the request signals from the LCDC core and generates the requests for the channels.

44.5.2 LCD Controller Core

44.5.2.1 Configuration Block

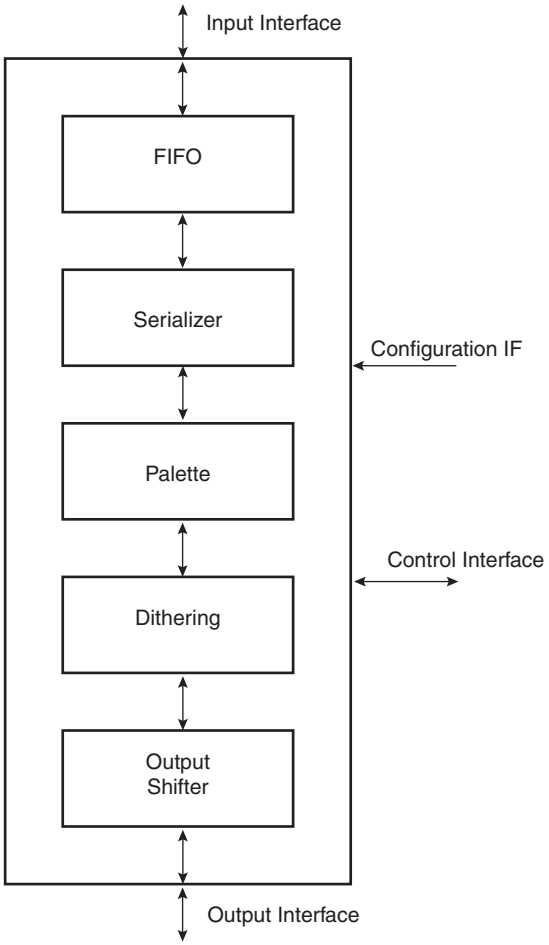
The configuration block is a set of programmable registers that are used to configure the LCDC core operation. These registers are written via the AHB slave interface. Only word access is allowed.

The description of the configuration registers can be found in "LCD Controller (LCDC) User Interface" on page 925.

44.5.2.2 Datapath

The datapath block contains five submodules: FIFO, Serializer, Palette, Dithering and Shifter. The structure of the datapath is shown in Figure 44-2.

Figure 44-2. 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 44-2.

Table 44-2. Datapath Parameters

Configuration				
DISTYPE	**SCAN**	**IFWIDTH**	**initial_latency**	**cycles_per_data**
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

44.5.2.3 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

44.5.2.4 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 44-4 and Table 44-5.

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 922.).

Table 44-4. 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 44-5. 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															

Table 44-5. Big Endian Memory Organization

Mem Addr	0x3	0x2	0x1	0x0
Pixel 24bpp packed	0			1
Pixel 24bpp packed	1		2	
Pixel 24bpp packed	2	3		
Pixel 24bpp packed	4			5
Pixel 24bpp unpacked	not used	0		

Table 44-6. 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																							

44.5.2.5 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 44-7. 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 44-7. Palette Configurations

Configuration		
DISTYPE	**PIXELSIZE**	**Palette**
TFT	1, 2, 4, 8	Palletized
TFT	16, 24	Non-palletized
STN Mono	1, 2	Palletized
STN Mono	4	Non-palletized
STN Color	1, 2, 4, 8	Palletized
STN Color	16	Non-palletized

The lookup table can be accessed by the host in R/W mode to allow the host to program and check the values stored in the palette. It is mapped in the LCD controller configuration memory map. The LUT is mapped as 16-bit half-words aligned at word boundaries, only word write access is allowed (the 16 MSB of the bus are not used). For the detailed memory map, see Table 44-14 on page 925.

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 44-8.

Table 44-8. 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]).

44.5.2.6 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 44-9 shows the correspondences between the gray levels and the duty cycle.

Table 44-9. 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 44-10.

Table 44-10. Dithering Algorithm for Monochrome Mode

Frame Number	Pattern	Pixel a	Pixel b	Pixel c	Pixel d
N	1010	ON	OFF	ON	OFF
N+1	0101	OFF	ON	OFF	ON
N+2	1010	ON	OFF	ON	OFF
N+3	0101	OFF	ON	OFF	ON
N+4	1111	ON	ON	ON	ON
N+5	1010	ON	OFF	ON	OFF
N+6	0101	OFF	ON	OFF	ON
N+7	1010	ON	OFF	ON	OFF
...	...	...	...	...	...

Consider now color display mode and two pixels p0 and p1 with the horizontal coordinates 4*n+0, and 4*n+1. A color pixel is composed of three components: {R, G, B}. Pixel p0 will be displayed sending the color components {R0, G0, B0} to the display. Pixel p1 will be displayed sending the color components {R1, G1, B1}. Suppose that the data read from memory and mapped to the lookup tables corresponds to shade level 10 for the three color components of both pixels, with the dithering pattern to apply to all of them being DP2_3 = “1101 1011 0110”. Table 44-11 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 44-11. 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
…	…	…	…	…	…	…	…

Table 44-11. Dithering Algorithm for Color Mode (Continued)

Frame	Signal	Shadow Level	Bit used	Dithering Pattern	4-bit LCDD	8-bit LCDD	Output
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.

44.5.2.7 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 925.

For a more detailed description of the LCD Interface, see "LCD Interface" on page 913.

44.5.2.8 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 44-12.

$$f_{LCDDOTCK} = \frac{f_{LCDC_clock}}{2 \times CLKVAL}$$

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 44-12. 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
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 44-2 on page 902. This limitation is given by the following formula:

44.5.2.9 Equation 1

$$(\mathrm{VHDLY} + \mathrm{HPW} + \mathrm{HBP} + 3) \times \mathrm{PCLK_PERIOD} \geq \mathrm{DPATH_LATENCY}$$

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 44-2 on page 902

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 44-3, Figure 44-4 and Figure 44-5 show the timing of LCDDOTCK, LCDDEN, LCDHSYNC and LCDVSYNC signals:

Figure 44-3. STN Panel Timing, CLKMOD 0

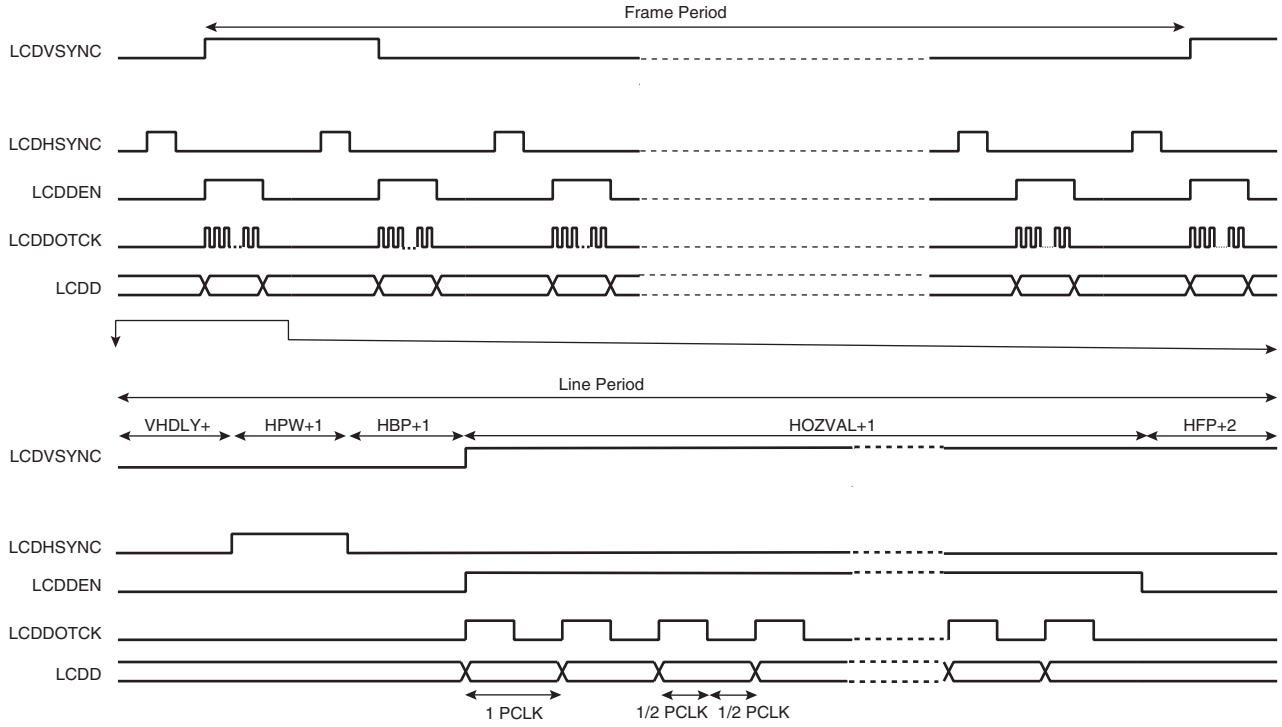


Figure 44-4. TFT Panel Timing, CLKMOD = 0, VPW = 2, VBP = 2, VFP = 1

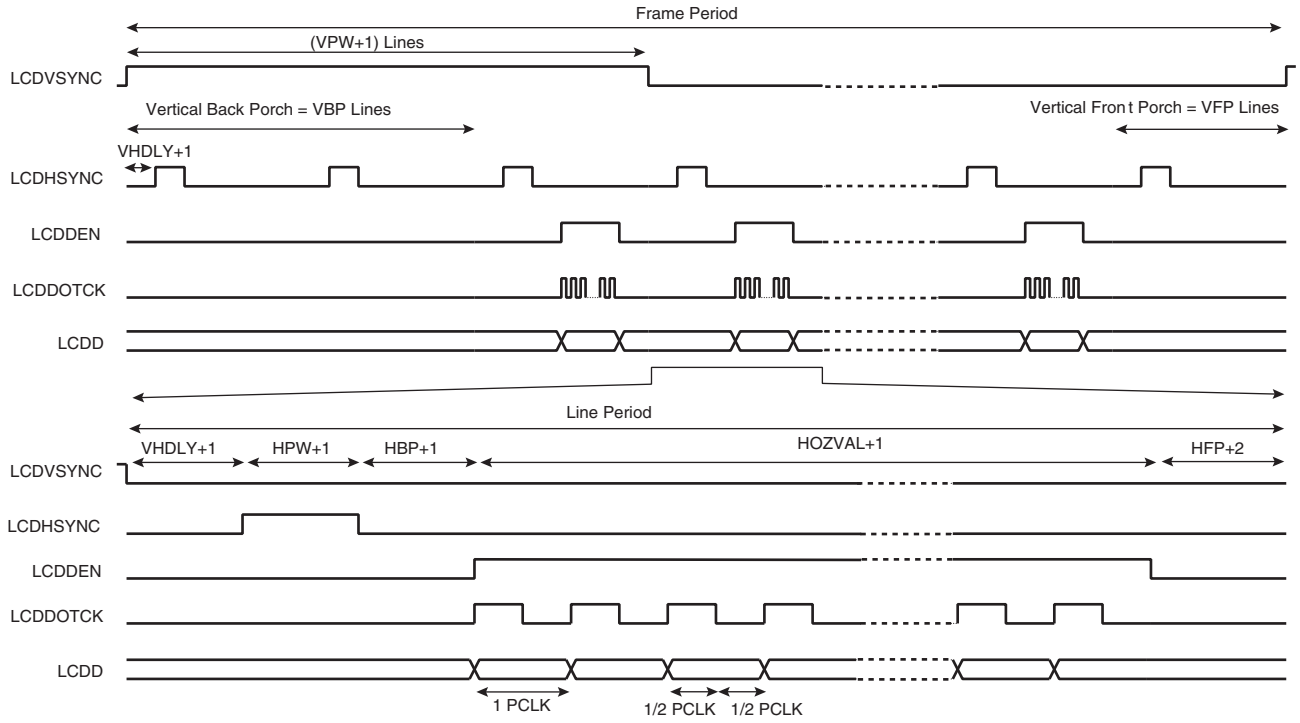
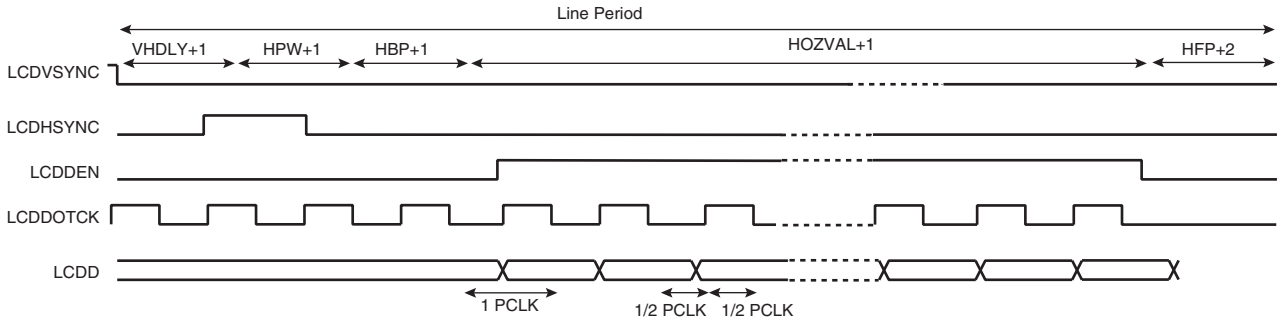


Figure 44-5. TFT Panel Timing (Line Expanded View), CLKMOD=1



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 909.

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.

44.5.2.10 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.

44.5.2.11 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.

44.5.3 LCD Interface

The LCD Controller interfaces with the LCD Module through the LCD Interface (Table 44-13 on page 919). 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 44-6 to Figure 44-10. Figure 44-6 on page 914 shows the 24-bit single scan TFT display timing; Figure 44-7 on page 915 shows the 4-bit single scan STN display timing for monochrome and color modes; Figure 44-8 on page 916 shows the 8-bit single scan STN display timing for monochrome and color modes; Figure 44-9 on page 917 shows the 8-bit Dual Scan STN display timing for monochrome and color modes; Figure 44-10 on page 918 shows the 16-bit Dual Scan STN display timing for monochrome and color modes.

Figure 44-6. TFT Timing (First Line Expanded View)

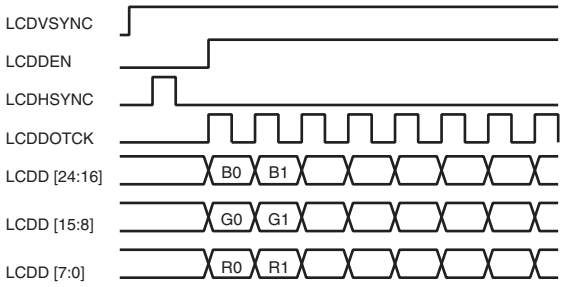


Figure 44-7. Single Scan Monochrome and Color 4-bit Panel Timing (First Line Expanded View)

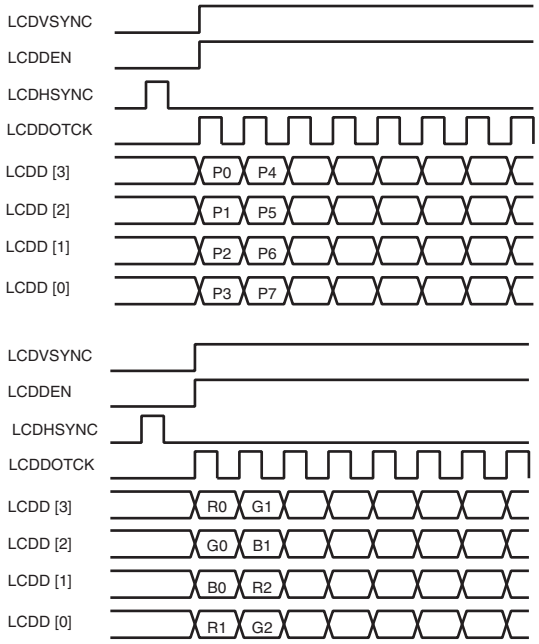


Figure 44-8. Single Scan Monochrome and Color 8-bit Panel Timing (First Line Expanded View)

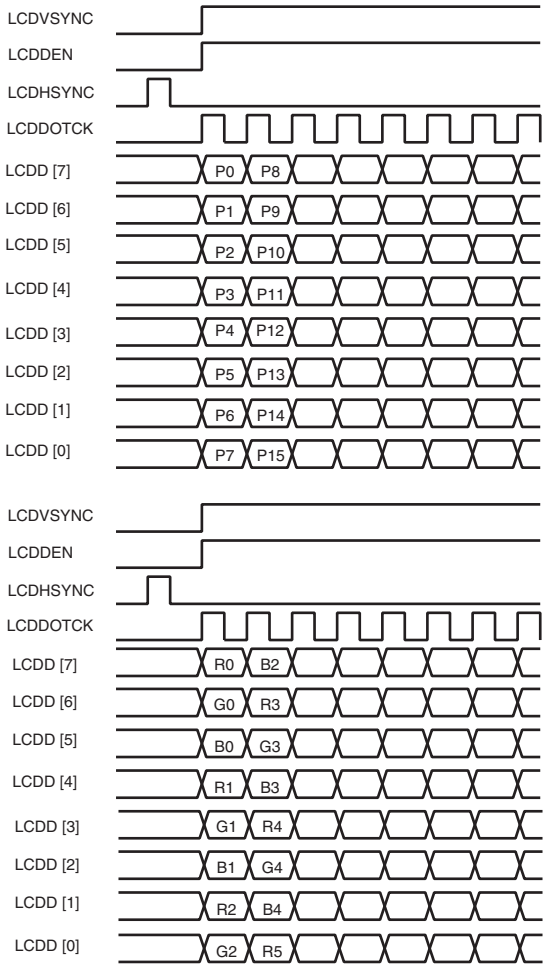


Figure 44-9. Dual Scan Monochrome and Color 8-bit Panel Timing (First Line Expanded View)

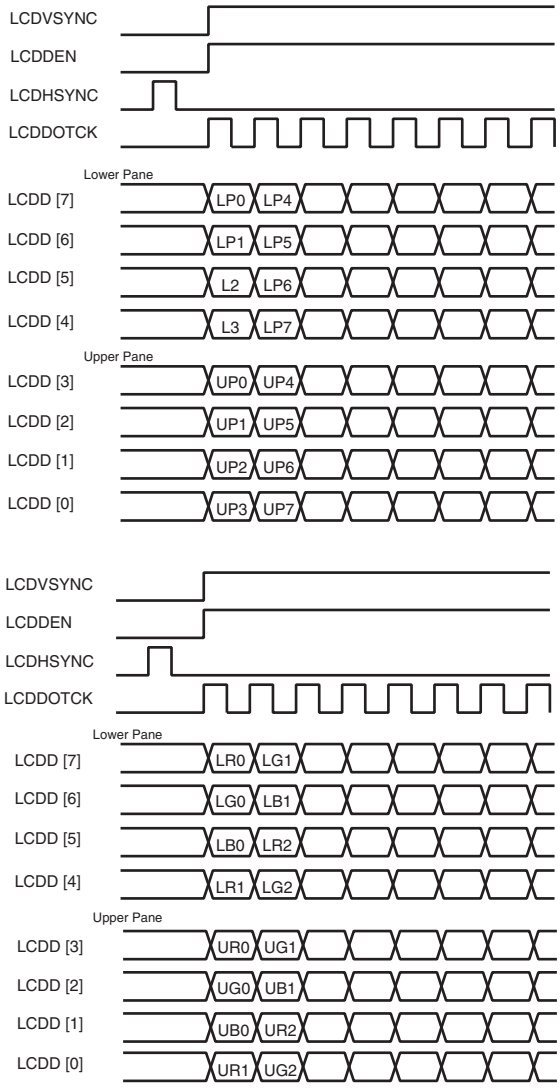


Figure 44-10. Dual Scan Monochrome and Color 16-bit Panel Timing (First Line Expanded View)

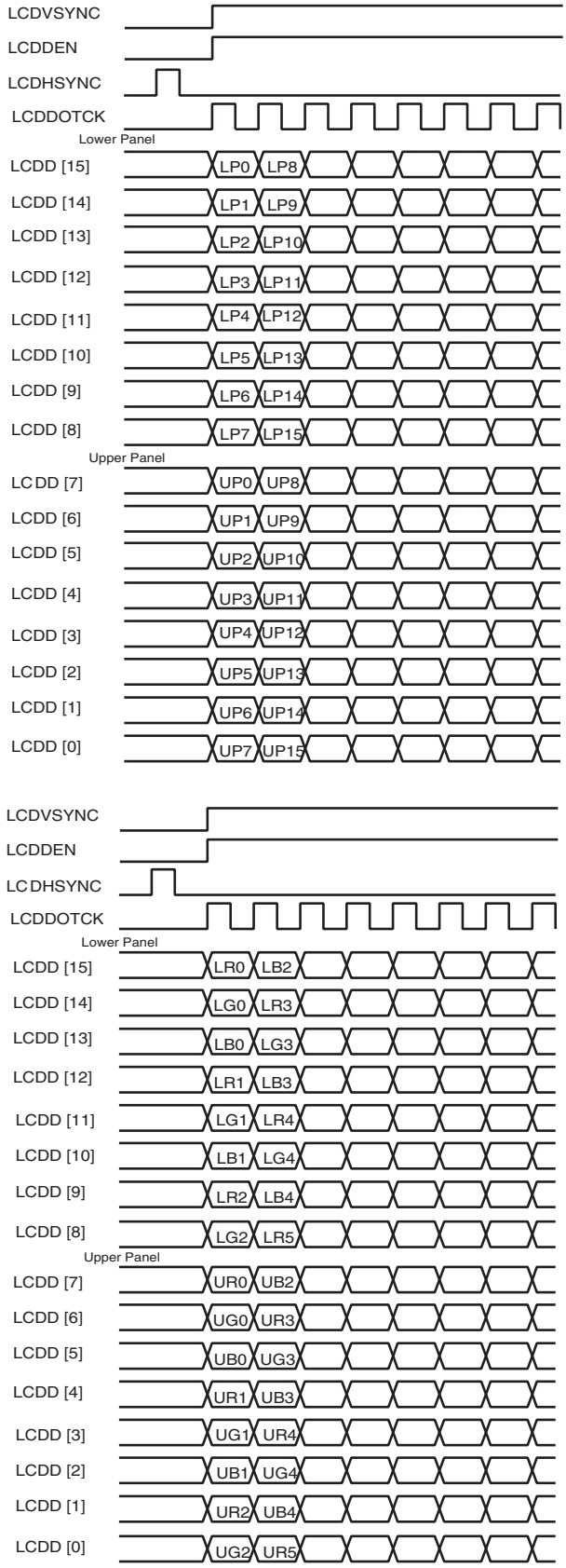


Table 44-13. LCD Signal Multiplexing

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	

44.6 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 921.
- 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 925.

44.7 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 904.
- 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 44-12 on page 909. 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 908

- 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 902
- 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 922).

• 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.

44.8 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.

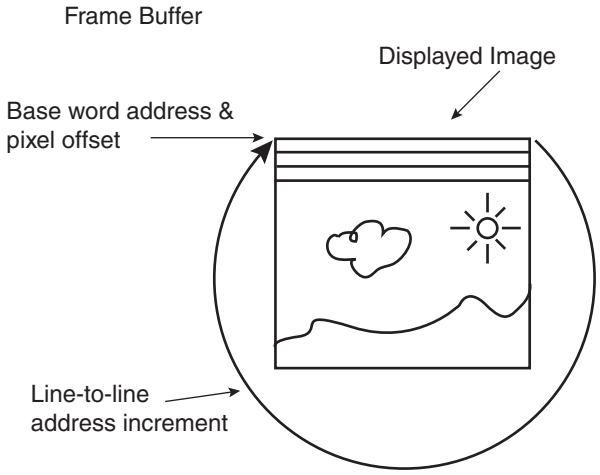
Once the host has updated the Frame Base Address Registers and the next DMA end of frame IRQ arrives, the backbuffer and the primary buffer are swapped and the host can work with the new backbuffer.

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.

44.9 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 44-11. 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.).

44.10 Register Configuration Guide

Program the PIO Controller to enable LCD signals.

Enable the LCD controller clock in the Power Management Controller.

44.10.1 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;

44.10.2 TFT Mode Example

This example is based on the NEC TFT color LCD module NL6448BC20-08.

TFT 640*480, 16-bit single scan, 60 frames/sec, pixel clock frequency = [21MHz..29MHz] with a typical value = 25.175 MHz.

The Master clock must be (2*(n + 1))*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;

44.11 LCD Controller (LCDC) User Interface

Table 44-14. 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 44-14. 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	

44.11.1 DMA Base Address Register 1

Name: DMABADDR1

Address:0x00700000

Access: Read-write

Reset value: 0x00000000

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

44.11.2 DMA Base Address Register 2

Name: DMABADDR2

Address:0x00700004

Access: Read-write

Reset value: 0x00000000

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

- **BADDR-L**

Base Address for the lower panel in dual scan mode only.

If a dual scan configuration is selected in LCDCON2 register or bit DMA2DEN in register DMACON is set, the bit DMAUPDT in that same register must be written after writing any new value to this field in order to make the DMA controller use this new value.

44.11.3 DMA Frame Pointer Register 1

Name: DMAFRMPT1

Address:0x00700008

Access: Read-only

Reset value: 0x00000000

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

- **FRMPT-U**

Current value of frame pointer for the upper panel in dual scan mode. Current value of frame pointer for the complete frame in single scan mode. Down count from FRMSIZE to 0.

Note: This register is read-only and contains the current value of the frame pointer (number of words to the end of the frame). It can be used as an estimation of the number of words transferred from memory for the current frame.

44.11.4 DMA Frame Pointer Register 2

Name: DMAFRMPT2

Address:0x0070000C

Access: Read-only

Reset value: 0x00000000

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

- **FRMPT-L**

Current value of frame pointer for the Lower panel in dual scan mode only. Down count from FRMSIZE to 0.

Note: This register is read-only and contains the current value of the frame pointer (number of words to the end of the frame). It can be used as an estimation of the number of words transferred from memory for the current frame.

44.11.5 DMA Frame Address Register 1

Name: DMAFRMADD1

Address:0x00700010

Access: Read-only

Reset value: 0x00000000

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

- **FRMADD-U**

Current value of frame address for the upper panel in dual scan mode. Current value of frame address for the complete frame in single scan.

Note: This register is read-only and contains the current value of the last DMA transaction in the bus for the panel/frame.

44.11.6 DMA Frame Address Register 2

Name: DMAFRMADD2

Address:0x00700014

Access: Read-only

Reset value: 0x00000000

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

- **FRMADD-L**

Current value of frame address for the lower panel in single scan mode only.

Note: This register is read-only and contains the current value of the last DMA transaction in the bus for the panel.

44.11.7 DMA Frame Configuration Register

Name: DMAFRMCFG

Address:0x00700018

Access: Read-write

Reset value: 0x00000000

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

- **FRMSIZE: Frame Size**

In single scan mode, this is the frame size in words. In dual scan mode, this is the size of each panel.

If a dual scan configuration is selected in LCDCON2 register or bit DMA2DEN in register DMACON is set, the bit DMAUPDT in that same register must be written after writing any new value to this field in order to make the DMA controller use this new value.

- **BRSTLN: Burst Length**

Program with the desired burst length - 1

44.11.8 DMA Control Register

Name: DMACON

Address:0x0070001C

Access: Read-write

Reset value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	DMA2DEN	DMAUPDT	DMABUSY	DMARST	DMAEN

- **DMAEN: DMA Enable**

0: DMA is disabled.

1: DMA is enabled.

- **DMARST: DMA Reset (Write-only)**

0: No effect.

1: Reset DMA module. DMA Module should be reset only when disabled and in idle state.

- **DMABUSY: DMA Busy**

0: DMA module is idle.

1: DMA module is busy (doing a transaction on the AHB bus).

- **DMAUPDT: DMA Configuration Update**

0: No effect

1: Update DMA Configuration. Used for simultaneous updating of DMA parameters in dual scan mode or when using 2D addressing. The values written in the registers DMABADDR1, DMABADDR2 and DMA2DCFG, and in the field FRMSIZE of register DMAFRMCFG, are accepted by the DMA controller and are applied at the next frame. This bit is used only if a dual scan configuration is selected (bit SCANMOD of LCDCON2 register) or 2D addressing is enabled (bit DMA2DEN in this register). Otherwise, the LCD controller accepts immediately the values written in the registers referred to above.

- **DMA2DEN: DMA 2D Addressing Enable**

0: 2D addressing is disabled (values in register DMA2DCFG are “don’t care”).

1: 2D addressing is enabled.

44.11.9 LCD DMA 2D Addressing Register

Name: DMA2DCFG

Address:0x00700020

Access: Read-write

Reset value: 0x00000000

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

- **ADDRINC: DMA 2D Addressing Address increment**

When 2-D DMA addressing is enabled (bit DMA2DEN is set in register DMACON), this field specifies the number of bytes that the DMA controller must jump between screen lines. Itb must be programmed as: [({address of first 32-bit word in a screen line} - {address of last 32-bit word in previous line})]. In other words, it is equal to 4*[number of 32-bit words occupied by each line in the complete frame buffer minus the number of 32-bit words occupied by each displayed line]. Bit DMAUPDT in register DMACON must be written after writing any new value to this field in order to make the DMA controller use this new value.

- **PIXELOFF: DAM2D Addressing Pixel offset**

When 2D DMA addressing is enabled (bit DMA2DEN is set in register DMACON), this field specifies the offset of the first pixel in each line within the memory word that contains this pixel. The offset is specified in number of bits in the range 0-31, so for example a value of 4 indicates that the first pixel in the screen starts at bit 4 of the 32-bit word pointed by register DMABADDR1. Bits 0 to 3 of that word are not used. This example is valid for little endian memory organization. When using big endian memory organization, this offset is considered from bit 31 downwards, or equivalently, a given value of this field always selects the pixel in the same relative position within the word, independently of the memory ordering configuration. Bit DMAUPDT in register DMACON must be written after writing any new value to this field in order to make the DMA controller use this new value.

44.11.10 LCD Control Register 1

Name: LCDCON1

Address:0x00700800

Access: Read-write, except LINECNT: Read-only

Reset value: 0x00002000

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

- **BYPASS: Bypass LCDDOTCK divider**

0: The divider is not bypassed. LCDDOTCK frequency defined by the CLKVAL field.

1: The LCDDOTCK divider is bypassed. LCDDOTCK frequency is equal to the LCDC Clock frequency.

- **CLKVAL: Clock divider**

9-bit divider for pixel clock (LCDDOTCK) frequency.

$$Pixel_clock = system_clock / (CLKVAL + 1) \times 2$$

- **LINECNT: Line Counter (Read-only)**

Current Value of 11-bit line counter. Down count from LINEVAL to 0.

44.11.11 LCD Control Register 2

Name: LCDCON2

Address:0x00700804

Access: Read-write

Reset value: 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

44.11.12 LCD Timing Configuration Register 1

Name: LCDTIM1

Address:0x00700808

Access: Read-write

Reset value: 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.

44.11.13 LCD Timing Configuration Register 2

Name: LCDTIM2

Address:0x0070080C

Access: Read-write

Reset value: 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.

44.11.14 LCD Frame Configuration Register

Name: LCDFRMCFG

Address:0x00700810

Access: Read-write

Reset value: 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**

44.11.15 LCD FIFO Register

Name: LCDFIFO

Address:0x00700814

Access: Read-write

Reset value: 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 933.

44.11.16 Dithering Pattern DP1_2 Register

Name: DP1_2

Address:0x0070081C

Access: Read-write

Reset value: 0xA5

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
DP1_2							

- **DP1_2: Pattern value for ½ duty cycle**

44.11.17 Dithering Pattern DP4_7 Register

Name: DP4_7

Address:0x00700820

Access: Read-write

Reset value: 0x5AF0FA5

31	30	29	28	27	26	25	24
–	–	–	–	DP4_7			
23	22	21	20	19	18	17	16
DP4_7							
15	14	13	12	11	10	9	8
DP4_7							
7	6	5	4	3	2	1	0
DP4_7							

- **DP4_7: Pattern value for 4/7 duty cycle**

44.11.18 Dithering Pattern DP3_5 Register

Name: DP3_5

Address:0x00700824

Access: Read-write

Reset value: 0xA5A5F

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	DP3_5			
15	14	13	12	11	10	9	8
DP3_5							
7	6	5	4	3	2	1	0
DP3_5							

- **DP3_5: Pattern value for 3/5 duty cycle**

44.11.19 Dithering Pattern DP2_3 Register

Name: DP2_3: Dithering Pattern DP2_3 Register

Address:0x00700828

Access: Read-write

Reset value: 0xA5F

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	DP2_3			
7	6	5	4	3	2	1	0
DP2_3							

- **DP2_3: Pattern value for 2/3 duty cycle**

44.11.20 Dithering Pattern DP5_7 Register

Name: DP5_7:

Address:0x0070082C

Access: Read-write

Reset value: 0xFAF5FA5

31	30	29	28	27	26	25	24
–	–	–	–	DP5_7			
23	22	21	20	19	18	17	16
DP5_7							
15	14	13	12	11	10	9	8
DP5_7							
7	6	5	4	3	2	1	0
DP5_7							

- **DP5_7: Pattern value for 5/7 duty cycle**

44.11.21 Dithering Pattern DP3_4 Register

Name: DP3_4

Address:0x00700830

Access: Read-write

Reset value: 0xFAF5

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
DP3_4							
7	6	5	4	3	2	1	0
DP3_4							

- **DP3_4: Pattern value for 3/4 duty cycle**

44.11.22 Dithering Pattern DP4_5 Register

Name: DP4_5

Address:0x00700834

Access: Read-write

Reset value: 0xFAF5F

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	DP4_5			
15	14	13	12	11	10	9	8
DP4_5							
7	6	5	4	3	2	1	0
DP4_5							

- **DP4_5: Pattern value for 4/5 duty cycle**

44.11.23 Dithering Pattern DP6_7 Register

Name: DP6_7

Address:0x00700838

Access: Read-write

Reset value: 0xF5FFAFF

31	30	29	28	27	26	25	24
–	–	–	–	DP6_7			
23	22	21	20	19	18	17	16
DP6_7							
15	14	13	12	11	10	9	8
DP6_7							
7	6	5	4	3	2	1	0
DP6_7							

- **DP6_7: Pattern value for 6/7 duty cycle**

44.11.24 Power Control Register

Name: PWRCON

Address:0x0070083C

Access: Read-write

Reset value: 0x0000000e

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

- **LCD_PWR: LCD module power control**

0 = lcd_pwr signal is low, other lcd_* signals are low.

0->1 = lcd_* signals activated, lcd_pwr is set high with the delay of GUARD_TIME frame periods.

1 = lcd_pwr signal is high, other lcd_* signals are active.

1->0 = lcd_pwr signal is low, other lcd_* signals are active, but are set low after GUARD_TIME frame periods.

- **GUARD_TIME**

Delay in frame periods between applying control signals to the LCD module and setting LCD_PWR high, and between setting LCD_PWR low and removing control signals from LCD module

- **LCD_BUSY**

Read-only field. If 1, it indicates that the LCD is busy (active and displaying data, in power on sequence or in power off sequence).

44.11.25 Contrast Control Register

Name: CONTRAST_CTR

Address:0x00700840

Access: Read-write

Reset value: 0x00000000

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

- **PS**

This 2-bit value selects the configuration of a counter prescaler. The meaning of each combination is as follows:

PS		
0	0	The counter advances at a rate of 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.

44.11.26 Contrast Value Register

Name: CONTRAST_VAL

Address:0x00700844

Access: Read-write

Reset value: 0x00000000

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

- **CVAL**

PWM compare value. Used to adjust the analog value obtained after an external filter to control the contrast of the display.

44.11.27 LCD Interrupt Enable Register

Name: LCD_IER

Address:0x00700848

Access: Write-only

Reset value: 0x0

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

- **LNIE: Line interrupt enable**

0: No effect

1: Enable each line interrupt

- **LSTLNIE: Last line interrupt enable**

0: No effect

1: Enable last line interrupt

- **EOFIE: DMA End of frame interrupt enable**

0: No effect

1: Enable End Of Frame interrupt

- **UFLWIE: FIFO underflow interrupt enable**

0: No effect

1: Enable FIFO underflow interrupt

- **OWRIE: FIFO overwrite interrupt enable**

0: No effect

1: Enable FIFO overwrite interrupt

- **MERIE: DMA memory error interrupt enable**

0: No effect

1: Enable DMA memory error interrupt

44.11.28 LCD Interrupt Disable Register

Name: LCD_IDR

Address:0x0070084C

Access: Write-only

Reset value: 0x0

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

- **LNID: Line interrupt disable**

0: No effect

1: Disable each line interrupt

- **LSTLNID: Last line interrupt disable**

0: No effect

1: Disable last line interrupt

- **EOFID: DMA End of frame interrupt disable**

0: No effect

1: Disable End Of Frame interrupt

- **UFLWID: FIFO underflow interrupt disable**

0: No effect

1: Disable FIFO underflow interrupt

- **OWRID: FIFO overwrite interrupt disable**

0: No effect

1: Disable FIFO overwrite interrupt

- **MERID: DMA Memory error interrupt disable**

0: No effect

1: Disable DMA Memory error interrupt

44.11.29 LCD Interrupt Mask Register

Name: LCD_IMR

Address:0x00700850

Access: Read-only

Reset value: 0x0

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

- **LNIM: Line interrupt mask**

0: Line Interrupt disabled

1: Line interrupt enabled

- **LSTLNIM: Last line interrupt mask**

0: Last Line Interrupt disabled

1: Last Line Interrupt enabled

- **EOFIM: DMA End of frame interrupt mask**

0: End Of Frame interrupt disabled

1: End Of Frame interrupt enabled

- **UFLWIM: FIFO underflow interrupt mask**

0: FIFO underflow interrupt disabled

1: FIFO underflow interrupt enabled

- **OWRIM: FIFO overwrite interrupt mask**

0: FIFO overwrite interrupt disabled

1: FIFO overwrite interrupt enabled

- **MERIM: DMA Memory error interrupt mask**

0: DMA Memory error interrupt disabled

1: DMA Memory error interrupt enabled

44.11.30 LCD Interrupt Status Register

Name: LCD_ISR

Address:0x00700854

Access: Read-only

Reset value: 0x0

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

- **LNIS: Line interrupt status**

0: Line Interrupt not active

1: Line Interrupt active

- **LSTLNIS: Last line interrupt status**

0: Last Line Interrupt not active

1: Last Line Interrupt active

- **EOFIS: DMA End of frame interrupt status**

0: End Of Frame interrupt not active

1: End Of Frame interrupt active

- **UFLWIS: FIFO underflow interrupt status**

0: FIFO underflow interrupt not active

1: FIFO underflow interrupt active

- **OWRIS: FIFO overwrite interrupt status**

0: FIFO overwrite interrupt not active

1: FIFO overwrite interrupt active

- **MERIS: DMA Memory error interrupt status**

0: DMA Memory error interrupt not active

1: DMA Memory error interrupt active

44.11.31 LCD Interrupt Clear Register

Name: LCD_ICR

Address:0x00700858

Access: Write-only

Reset value: 0x0

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

- **LNIC: Line interrupt clear**

0: No effect

1: Clear each line interrupt

- **LSTLNIC: Last line interrupt clear**

0: No effect

1: Clear Last line Interrupt

- **EOFIC: DMA End of frame interrupt clear**

0: No effect

1: Clear End Of Frame interrupt

- **UFLWIC: FIFO underflow interrupt clear**

0: No effect

1: Clear FIFO underflow interrupt

- **OWRIC: FIFO overwrite interrupt clear**

0: No effect

1: Clear FIFO overwrite interrupt

- **MERIC: DMA Memory error interrupt clear**

0: No effect

1: Clear DMA Memory error interrupt

44.11.32 LCD Interrupt Test Register

Name: LCD_ITR

Address:0x00700860

Access: Write-only

Reset value: 0x0

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

- **LNIT: Line interrupt test**

0: No effect

1: Set each line interrupt

- **LSTLNIT: Last line interrupt test**

0: No effect

1: Set Last line interrupt

- **EOFIT: DMA End of frame interrupt test**

0: No effect

1: Set End Of Frame interrupt

- **UFLWIT: FIFO underflow interrupt test**

0: No effect

1: Set FIFO underflow interrupt

- **OWRIT: FIFO overwrite interrupt test**

0: No effect

1: Set FIFO overwrite interrupt

- **MERIT: DMA Memory error interrupt test**

0: No effect

1: Set DMA Memory error interrupt

44.11.33 LCD Interrupt Raw Status Register

Name: LCD_IRR

Address:0x00700864

Access: Write-only

Reset value: 0x0

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

- **LNIR: Line interrupt raw status**

0: No effect

1: Line interrupt condition present

- **LSTLNIR: Last line interrupt raw status**

0: No effect

1: Last line Interrupt condition present

- **EOFIR: DMA End of frame interrupt raw status**

0: No effect

1: End Of Frame interrupt condition present

- **UFLWIR: FIFO underflow interrupt raw status**

0: No effect

1: FIFO underflow interrupt condition present

- **OWRIR: FIFO overwrite interrupt raw status**

0: No effect

1: FIFO overwrite interrupt condition present

- **MERIR: DMA Memory error interrupt raw status**

0: No effect

1: DMA Memory error interrupt condition present

45. Two D Graphics Controller (TDGC)

45.1 Description

The Two D Graphics Controller (TDGC) features a hardware accelerator which highly simplifies drawing tasks and graphic management operations. The hardware accelerator makes it easy to draw lines and complex polygons and to perform block transfers within the frame buffer.

The TDGC also features a draw command queue that automatically executes a more complex drawing function that is composed of several register accesses.

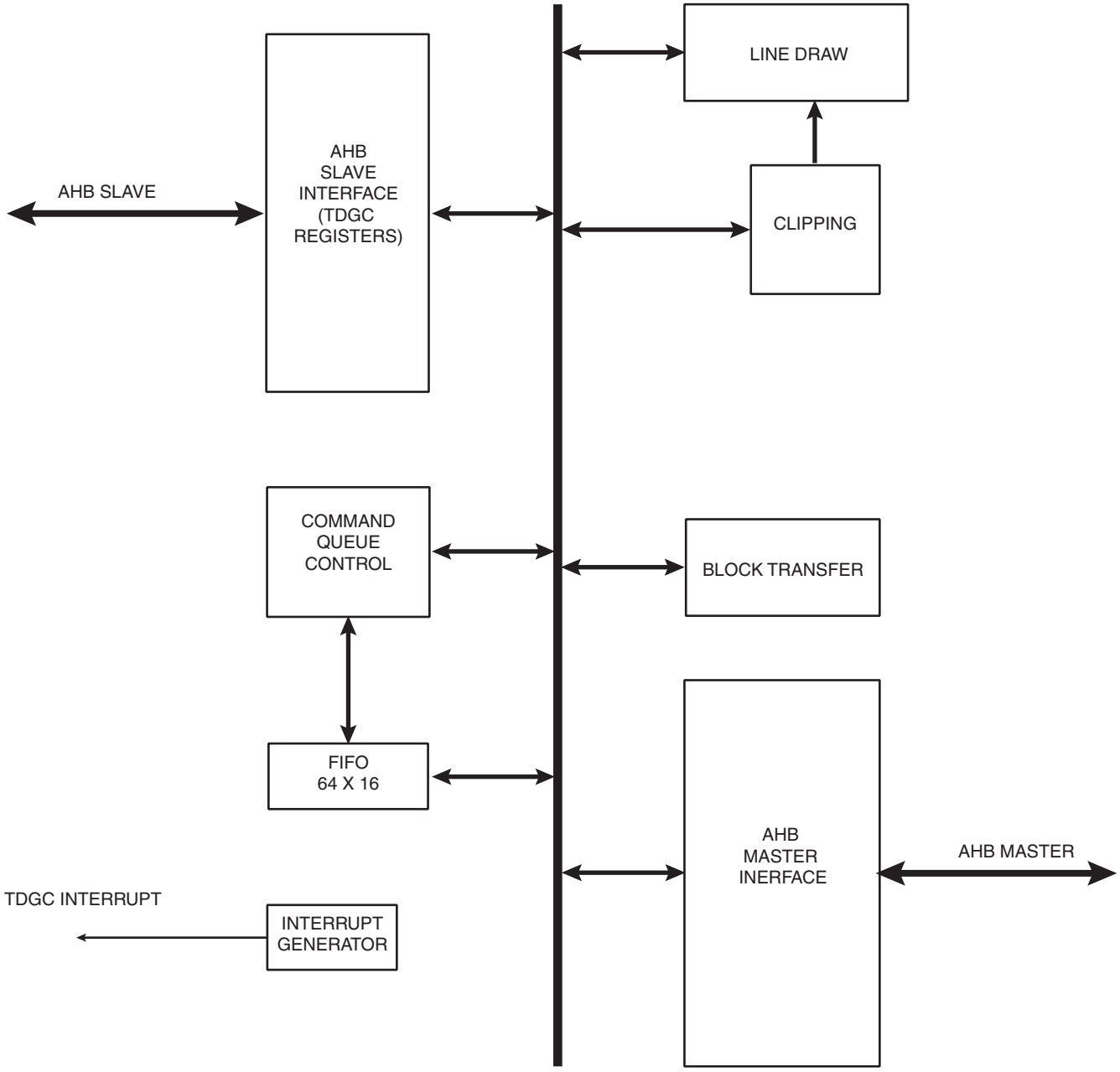
The TDGC supports access to both external video RAM mapped to any EBI chip selects and internal RAM (frame buffer). The external video RAM can have an 8-bit, 16-bit or 32-bit data bus.

The TDGC supports 1 bit, 2 bits, 4 bits, 8 bits, 16 bits and 24 bits per pixel. The maximum virtual memory page can be 2048 x 2048 pixels. The data written into the video RAM by draw functions can be in little endian, big endian and WinCE format.

The TDGC is connected to the AHB (Advanced High Performance Bus) in two ways: first, as a master for reading and writing pixel data, and second, as a slave for register configuration.

45.2 Block Diagram

Figure 45-1. Two D Graphic Controller (TGDC) Block Diagram



45.3 Functional Description

45.3.1 Hardware Acceleration

The hardware acceleration performs multiple block transfers, line draw or fill commands by issuing draw commands to the controller. This technique makes it possible to get rid of complex software layers.

45.3.1.1 Line Draw

Lines can be drawn up to a specific width scalable from 1 to 16 pixels. They can also be drawn as a broken line with a pattern set by a 16-bit pattern register.

The following registers need to be programmed by the software to start a line draw:

- **TDGC_SBXR:** Starting position in pixel units on X-axis.
- **TDGC_TEXR:** Ending position in pixel units on X-axis.
- **TDGC_SBYR:** Starting position in pixel units on Y-axis.
- **TDGC_TEYR:** Ending position in pixel units on Y-axis.
- **TDGC_LOR:** Set LOC bitfield to select logic operation MOV.
- **TDGC_CSR:** Set CLR[3:0] bitfield to select a color.
- **TDGC_GOR:** Set GOC bitfield and OP bitfields to the to select the following values.
  - Set GOC[7:4] to select line draw.
  - Set OP[0] to select a 1D or 2D line draw.
  - Set OP[1] to select update X & Y or no update X & Y.
  - Set OP[2] to select relative or absolute.
  - Set OP[3] to select no transformation.
- **TDGC_LWR:** Set LWD bitfield to select line width in pixels.
- **TDGC_LPR:** Set LPT bitfield to select a pattern for the line. 0xFFFF is for solid.

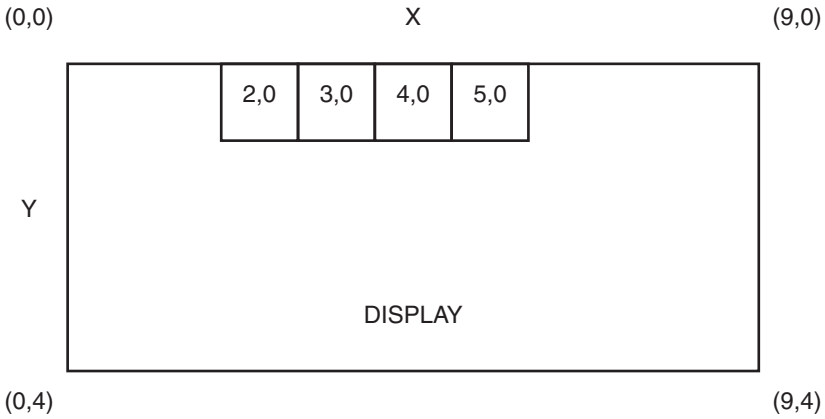
LTB bit in TDGC_GSR will signal the completion of the drawing operation.

45.3.1.2 Line Draw Modes

- Absolute Line Draw

First pixel position on the line is the pixel position loaded into source/begin x and source/begin y registers. Last pixel position on the line is the pixel position loaded into target/end x and target/end y registers.

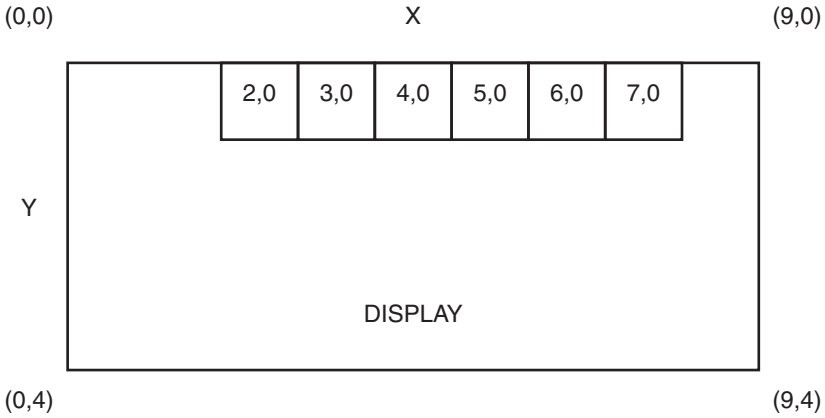
Figure 45-2. Absolute Line Draw From (2,0) to (5,0)



- Relative Line Draw

First pixel position on the line is the pixel position loaded into source/begin x and source/begin y registers. Last pixel position on the line is the pixel position loaded into target/end x and target/end y registers plus the start pixel position loaded into source/begin x and source/begin y registers. The target/end x and target/end y registers stand for the length of the line to be drawn.

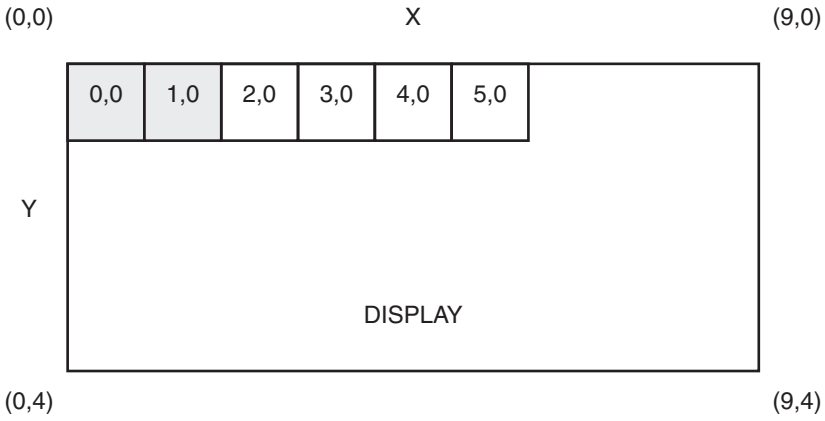
Figure 45-3. Relative Line Draw From (2,0) to (5,0)



45.3.1.3 Relative Line Draw with Update XY Option

First pixel position is loaded into source/begin x and source/begin y registers and last pixel position is loaded into target/end x target/end y registers. After the necessary color and pattern are loaded into appropriate registers, respectively TDGC_CSR and TDGC_LPR, line draw is initiated with a write to TDGC_LOR register. If the line needs to be extended, then a single write to the TDGC_GOR register with update XY option set initiates line draw (extension).

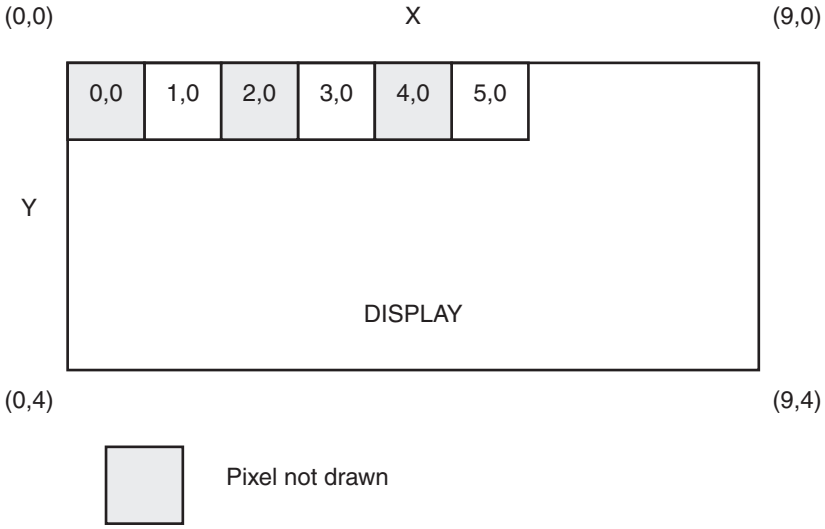
Figure 45-4. Line Draw From (0,0) to (1,0) and Update XY Four Times



45.3.1.4 1D Line Draw with Broken Pattern 0xAAAA

The first pixel drawn is based on the least significant bit of the pattern register. In this example the first pixel is not drawn. After rendering each pixel, the pattern register is rotated to the right by one bit.

Figure 45-5. 1D Line Draw From (0,0) to (5,0) with Broken Pattern 0xAAAA



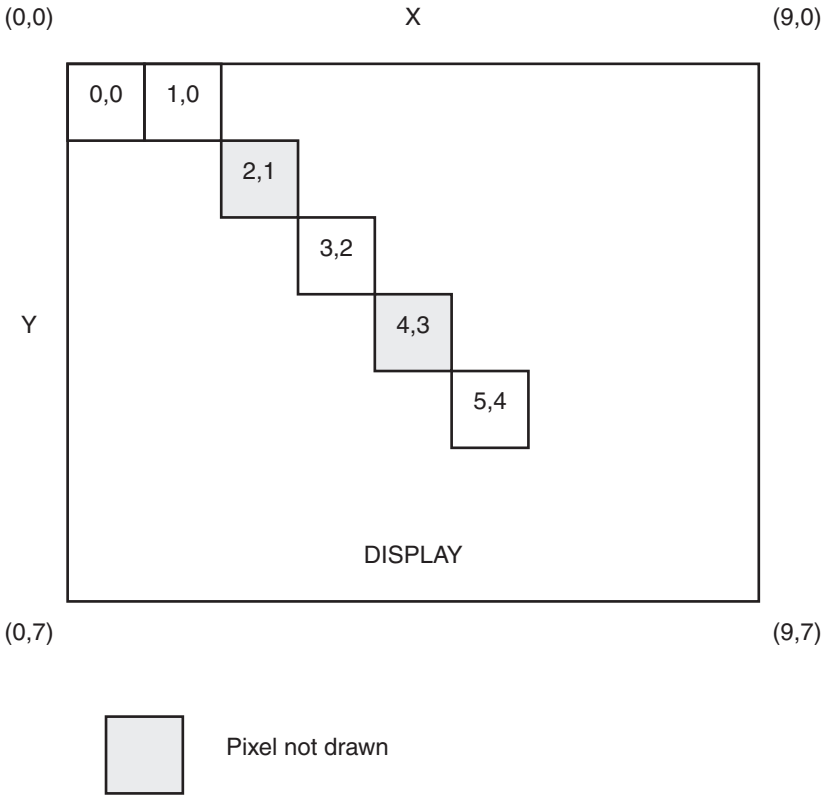
45.3.1.5 2D Line Draw with Broken Pattern 0xEBBB

In 2D line drawing, the bit pattern does not rotate. Each pixel rendered is based on the last two bits of its X and Y addresses as shown below.

Table 45-1. 2D Line Draw Pattern

	X[1:0] = 0	**X[1:0] = 1**	**X[1:0] = 2**	**X[1:0] = 3**
Y[1:0] = 0	LPT[12]	LPT[13]	LPT[14]	LPT[15]
Y[1:0] = 1	LPT[8]	LPT[9]	LPT[10]	LPT[11]
Y[1:0] = 2	LPT[4]	LPT[5]	LPT[6]	LPT[7]
Y[1:0] = 3	LPT[0]	LPT[1]	LPT[2]	LPT[3]

Figure 45-6. 2D Line Draw From (0,0) to (5,4) with Broken Pattern 0xEBBB



45.3.1.6 Block Transfer

A rectangle (or square) shape in pixel units can be transferred between areas in the virtual memory area of the VRAM. Logical operations such as AND, OR, XOR, NOT, NOR, NAND and XNOR can be made between the pixels in the source area and the destination area and stored in the destination area.

The following registers need to be programmed to perform a block transfer:

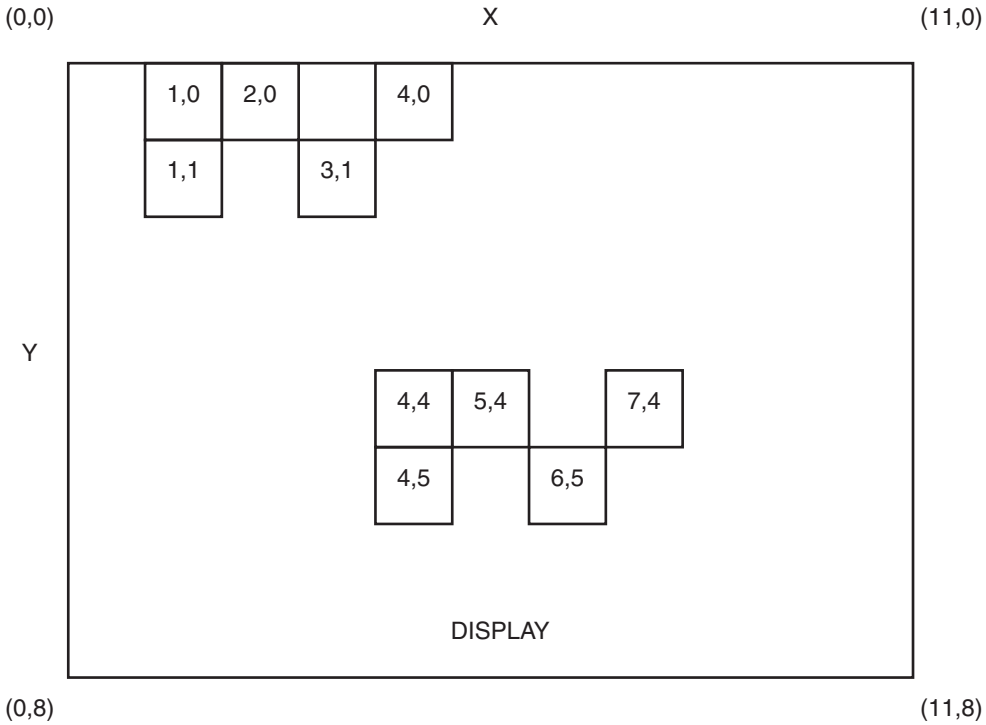
- **TDGC_BTSXR:** size in pixel units on X-axis.
- **TDGC_BTSYR:** size in pixel units on Y-axis.
- **TDGC_SBXR:** starting position in pixel units on X-axis.
- **TDGC_TEXR:** ending position in pixel units on X-axis.
- **TDGC_SBYR:** starting position in pixel units on Y-axis.
- **TDGC_TEYR:** ending position in pixel units on Y-axis.
- **TDGC_LOR:** set LOC bitfield to select a logic operation MOV/AND/OR/XOR/NOT/NOR/NAND/XNOR.
- **TDGC_CSR:** set CLR[3:0] bitfield to select a color.
- **TDGC_GOR:** set GOC bitfield and OP bitfields to select the following values.
  - Set GOC to select block transfer.
  - Set OP[1:0] to select no update X & Y.
  - Set OP[2] to select absolute.
  - Set OP[3] to select no transformation.

BTB bit in TDGC_GSR signals the completion of the block transfer operation.

45.3.1.7 *Absolute Block Transfer*

A block of data is copied (or xor/or/and/xnor/nand/not/nor/xnor) from the start position loaded into start x and start y registers to the target position loaded into target x and target y registers. The size of data is based on size x and size y registers.

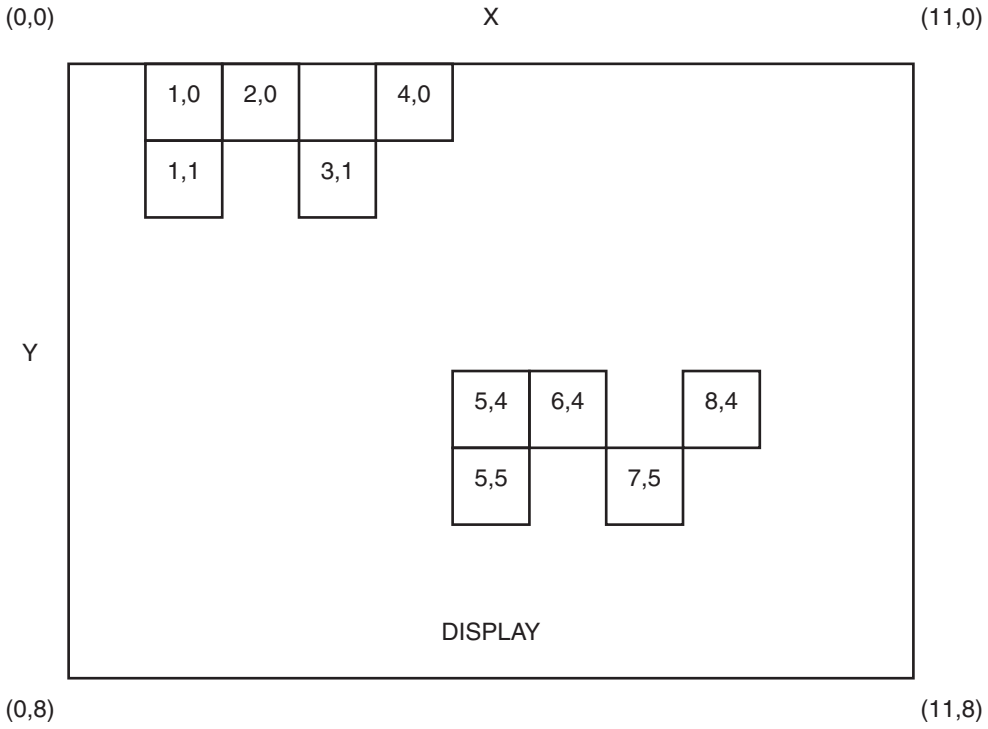
Figure 45-7. Absolute Block Transfer from (1,0) to (4,4) Size (4,2)



45.3.1.8 *Relative Block Transfer*

A block of data is copied (or xor/or/and/xnor/nand/not/nor/xnor) from the start position loaded into start x and start y registers to a position offset based on the target position loaded into target x and target y registers with respect to start position. The size of data is based on size x and size y registers.

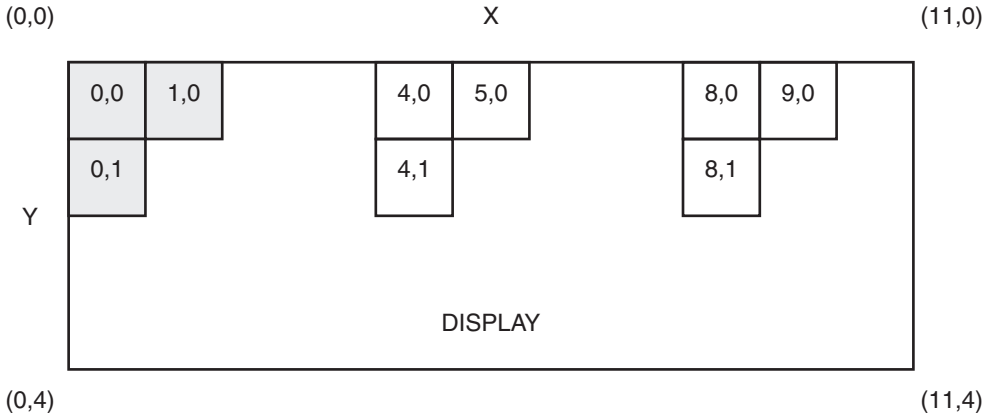
Figure 45-8. Relative Block Transfer From (1,0) to (4,4) Size (4,2)



45.3.1.9 Block Transfer with Update X/Y

If a drawing is rendered and needs to be duplicated along the X-axis or Y-axis, then a single write to the TDGC_GOC will duplicate the drawing in the desired direction. This saves writing actions to startx, starty, endx, endy, color and logic operation registers and results in a faster rendering.

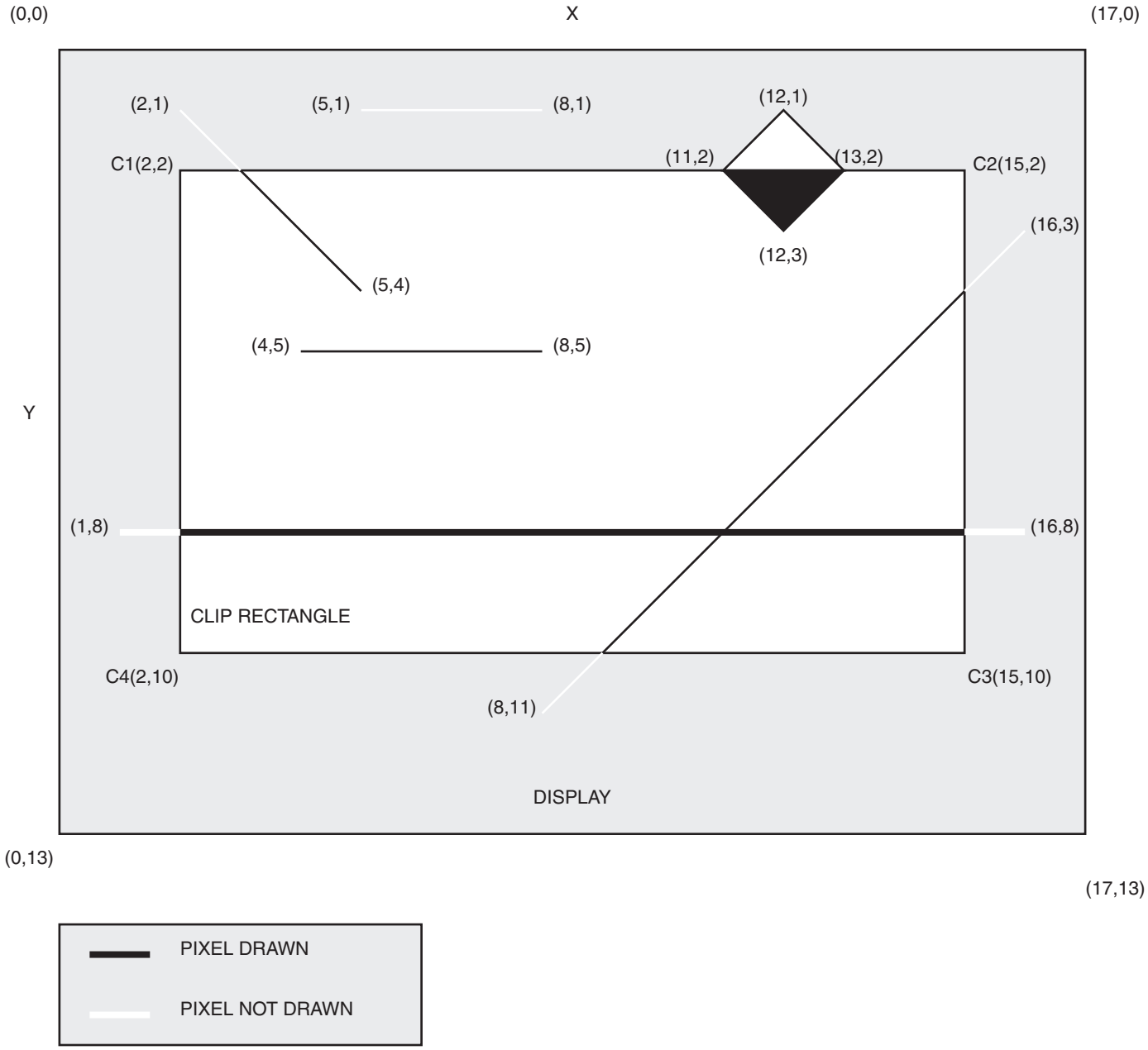
Figure 45-9. Block Transfer with Update X Two Times



45.3.1.10 Clipping

This function disables drawing outside the selected rectangular field. The clipping x and y coordinates are defined by the values, in pixel units, programmed in four clipping vertex registers TDGC_CXMINR, TDGC_CXMAXR, TDGC_CYMINR, TDGC_CYMAXR. Clipping is enabled by setting CEN bitfield in TDGC_CCR.

Figure 45-10. Line and Polygon Clipping



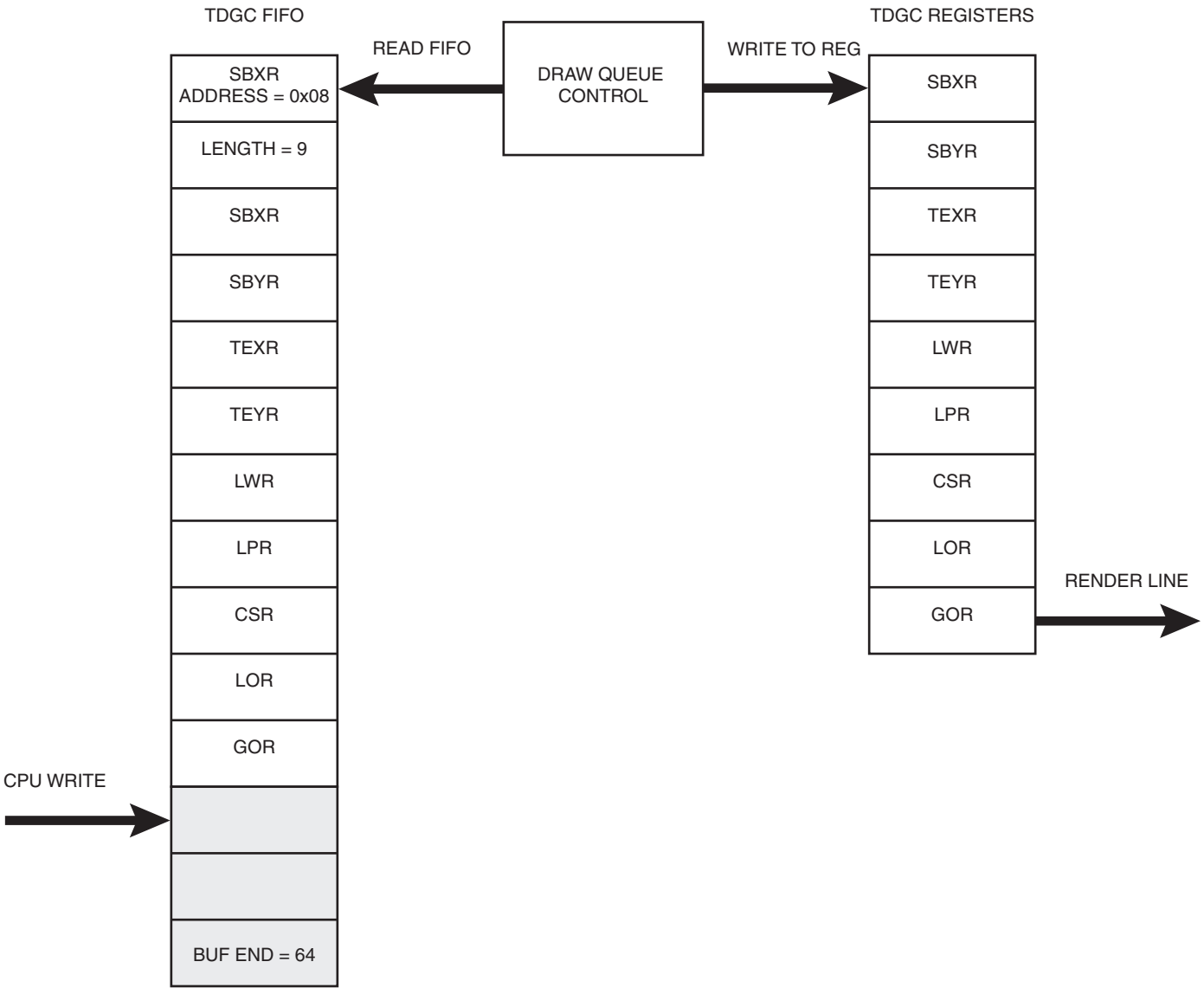
45.3.1.11 Draw Command Queuing

Multiple block transfer, line draw or fill commands can be issued to the TDGC by writing the commands to a 64 x 16-bit wide FIFO. All drawing specific registers can be written via writes to the FIFO by writing the address, length and the register value to the command queue.

Length is based on the number of consecutive register writes. All accesses to the command queue FIFO are by reading/writing to the 16-bit wide TDGC_CQR. The pointer to the command queue is automatically incremented by the controller. FIFO command size is limited to 64 half words.

In order to optimize the small command FIFO (64 half words) use, and due to the restriction of the APB access that has to be full word only, all 32-bit registers are split into two registers. This only affects the clipping command where the start and end coordinates of the line to be clipped could be outside the range of 2048 pixels and may need 32 bits to define line coordinates. Thus extra accesses to the FIFO must be made if the clip line coordinate has to be represented in 32 bits versus 16 bits. See the specific example of clipping command that shows the change in code as described above.

Figure 45-11. Command Queuing



45.3.1.12 Recommended Procedure for Using the Command Queue

- Load the entire queue (equal to 64) with commands.
- Enable command queue buffer empty interrupt (BUFE) in TDGC_GMR if using interrupts instead of polling.

- Wait for Command queue buffer empty status (BUFE) in TDGC_GIR with a five second timeout or wait for an interrupt event if interrupt is enabled (recommended).
- Post an event to the graphics task when interrupt is triggered or exit the loop checking for the status when command queue buffer is empty (BUFE in TDGC_GIR). Load the next set of commands (refer to code examples).

45.3.1.13 Procedure to Switch from Command Queue Drawing to Direct Drawing

- Wait for command queue buffer empty status in TDGC_GIR.
- Wait for line drawing engine bit (LTB bitfield in TDGC_GSR) to clear if a line is being drawn.
- Wait for block transfer engine bit (BTB bitfield in TDGC_GSR) to clear if a block is being transferred.

45.3.2 Video RAM

The Video RAM type and the address generated to access it are mainly based on data bus type (8, 16 or 32 bits), on number of bits per pixel and the virtual memory size required by the system.

The TDGC supports SRAM, PSRAM and SDRAM memory chips of 8-bit, 16-bit or 32-bit data buses. Memory chips can either be external memory (connected to EBI) or internal memory. The most significant 12 bits of the 32-bit video memory address can be programmed with the required offset.

The TDGC sees the video memory as a maximum virtual page of 2048(column-x) x 2048(rows-y) pixels with a pixel resolution up to 24 bpp. Hence, the maximum video memory that TDGC can see is 12MBytes = 2048 * 2048 * 24/8.

The row size of the virtual memory can be programmed to be 256, 512, 1024 or 2048 pixels. Since the minimum row size selection is 256 pixels and the next size up is 512 pixels, some chips that are tailored for 240(column size) x 320(row size) at 8 bpp LCDs have only 80 Kbytes of internal RAM and thus do not fit in the resolution scheme defined above. In order to make the TDGC compliant with this kind of use, a special option was added to make 320 pixel wide row at 8 bpp selection possible. However, this slows down the drawing process. For instance, when a line draw command is issued, the TDGC calculates the row offset based on the start/end pixel coordinates of the line draw versus a predefined shift in bits for row size selections of 256, 512, 1024 and 2048 (they are all powers of 2 and hence the shift is pre-defined in logic).

There are two suggestions to solve this problem:

- If a significant amount of drawing using the TDGC is required and 240 x 320 at 8 bpp is not mandatory, a bigger frame can be used thus taking advantage of the TDGC drawing speed.
- If there is a firm requirement for 240 x 320 at 8 bpp, then the special option can be enabled in the TDGC that makes 240 x 340 at 8 bpp support possible but slow, or the TDGC can be disabled and software that can be faster is used instead.

However, a pixel resolution of 320(column) x 240(row) at 8 bpp can use the internal memory of 80 Kbytes if necessary, as a row size selection of 256 pixels can be used. There is however no problem with any ¼ VGA at anything less than 8 bpp.

45.4 Examples of Drawing Functions

45.4.1 Line Draw

This function draws a thick (2 pixels wide) solid black line from start point (startx, starty) to end point (endx, endy). startx, starty, endx, endy should be in pixel units.

```
Void line_draw(unsigned short startx, unsigned short starty,
    unsigned short endx, unsigned short endy)
{
while(graphics_control.TDGC_GSR & 3);
graphics_control.TDGC_SBXR = startx;
graphics_control.TDGC_SBYR = starty;
graphics_control.TDGC_TEXR = endx;
graphics_control.TDGC_TEYR = endy;
graphics_control.TDGC_LOR = 0x00; // Select logic operation MOV
graphics_control.TDGC_CSR = 0x00; // Colour black
graphics_control.TDGC_LWR = 0x02; // 2 pixels wide
graphics_control.TDGC_LPR = 0xFFFF; // Solid line
graphics_control.TDGC_GOR = 0xD5; // Line draw, absolute, no update, 1D
pattern
while(graphics_control.TDGC_GSR & 1);
}
```

45.4.2 Block Transfer

This function OR's source data (startx, starty) of size (sizex, sizey) with destination data (endx, endy) and writes it to the destination memory area. sizex, sizey, startx, starty, endx, endy should be in pixel units.

```
Void block_transfer(unsigned short startx, unsigned short starty, unsigned
short endx,
    unsigned short endy, unsigned short sizex, unsigned short sizey)
{
while(graphics_control.TDGC_GSR & 3);
graphics_control.TDGC_TDGC_BTSXR = sizex;
graphics_control.TDGC_TDGC_BTSYR = sizey;
graphics_control.TDGC_SBXR = startx;
graphics_control.TDGC_SBYR = starty;
graphics_control.TDGC_TEXR = endx;
graphics_control.TDGC_TEYR = endy;
graphics_control.TDGC_LOR = 0x01; // Select logic operation OR
graphics_control.TDGC_GOR = 0xB4; // Selects block transfer, absolute, no
update
while(graphics_control.TDGC_GSR & 2);
}
```

45.4.3 Clipped Line Draw

This function draws a thick (2 pixels wide) patterned (pixel ON,OFF, ON, OFF…) black line from start point (startx, starty) to end point (endx, endy). Only the pixels that fall on or within the clip rectangle boundary are drawn. startx, starty, endx, endy should be in pixel units.

```
Void clipped_line_draw(signed long int startx, signed long int starty,
      signed long int endx, signed long int endy)
{
while(graphics_control.TDGC_GSR & 3);
```

```
graphics_control.TDGC_SBXR = startx;
graphics_control.TDGC_SBYR = starty;
graphics_control.TDGC_TEXR = endx;
graphics_control.TDGC_TEYR = endy;
graphics_control.TDGC_LOR = 0x00; // Select logic operation MOV
graphics_control.TDGC_CSR = 0x00; // Colour black
graphics_control.TDGC_LWR = 0x02; // 2 pixels wide
graphics_control.TDGC_LPR = 0x5555; // Patterned line ON, OFF, ON, OFF …

// set clip rectangle boundary (4,2), (8,2), (4,4), (8,4)
graphics_control.TDGC_CXMINR = 4;
graphics_control.TDGC_CXMAXR = 8;
graphics_control.TDGC_CYMINR = 2;
graphics_control.TDGC_CYMAXR = 4;
graphics_control.TDGC_CCR = 1; // Enable clipping
graphics_control.TDGC_GOR = 0xD5; // Line draw, absolute, no update, 1D
pattern
while(graphics_control.TDGC_GSR & 1);
}
```

45.4.4 Drawing Using Command Queue

This function draws a series of lines. All commands are written into the FIFO which is automatically converted to drawing commands. startx, starty, endx and endy should be in pixel units.

```
void command_queue_draw(unsigned short startx, unsigned short starty,
unsigned short endx)
{
int add_count=6;
int i;

add_count = 6; // number of writes to TDGC_CQR
chk_for_buffer_empty(add_count);
graphics_control.TDGC_CQR  =  0x08;        // start address, start x
register address
graphics_control.TDGC_CQR  =  0x04;       // length, number of registers to
update
graphics_control.TDGC_CQR  =  120;          // start x
graphics_control.TDGC_CQR  =  160;          // start y
graphics_control.TDGC_CQR  =  0;              // end x
graphics_control.TDGC_CQR  =  0;              // end y

for(i = 0;i < 320; i ++)
{
add_count = 7;
chk_for_buffer_empty(add_cnt);
graphics_control.TDGC_CQR  =  0x14;       // start address, end y register
address
```

```
graphics_control.TDGC_CQR  =  0x05;        // length,# of registers to
update
graphics_control.TDGC_CQR  =  i;               // end y
graphics_control.TDGC_CQR  =  0xFFFF;   // line pattern
graphics_control.TDGC_CQR  =  0x04;        // colour select
graphics_control.TDGC_CQR  =  0x00;        // logic operation, MOV
graphics_control.TDGC_CQR  =  0xD4;       // operation, line draw
}
}

void chk_for_buffer_empty(int add_cnt)
{
// Get BUFW_CNTR from TDGC_CQCR and see if there is enough room in FIFO
if((((graphics_control.TDGC_CQCR & 0xFC0) >> 6) + add_cnt) >= 64)
{
if(interrupt_enabled)
// Wait for buffer empty interrupt to rise which should be enabled at first
place in TDGC_GIMR.
sleep_until_event_from_isr;
else
while(!graphics_control.TDGC_GIR);
}
```

45.5 Two D Graphic Controller (TDGC) User Interface

Table 45-2. TDGC Register Mapping

Offset	Register	Name	Access	Reset
0x00	Block Transfer Size X Register	TDGC_BTSXR	Read-write	0x00000000
0x04	Block Transfer Size Y Register	TDGC_BTSYR	Read-only	0x00000000
0x08	Source/Begin X Register	TDGC_SBXR	Write-only	0x00000000
0x0C	Source/Begin Y Register	TDGC_SBYR	Read-write	0x00000000
0x10	Target/End X Register	TDGC_TEXR	Read-only	0x00000000
0x14	Target/End Y Register	TDGC_TEYR	Write-only	0x00000000
0x18	Line Width Register	TDGC_LWR	Read-write	0x00000000
0x1C	Line Pattern Register	TDGC_LPR	Read-write	0x00000000
0x20	Color Select Register	TDGC_CSR	Read-write	0x00000000
0x24	Logic Operation Register	TDGC_LOR	Read-write	0x00000000
0x28	Graphics Operation Register	TDGC_GOR	Read-write	0x00000000
0x2C	Extended Begin X Register	TDGC_EBXR	Read-write	0x00000000
0x30	Extended Begin Y Register	TDGC_EBYR	Read-write	0x00000000
0x34	Extended End X Register	TDGC_EEXR	Read-write	0x00000000
0x38	Extended End Y Register	TDGC_EEYR	Read-write	0x00000000
0x3C	Extended Color Select Register	TDGC_ECSR	Read-write	0x00000000
0x40	Clip Control Register	TDGC_CCR	Read-write	0x00000000
0x44	Clip Rectangle X Minimum Register	TDGC_CXMINR	Read-write	0x00000000
0x48	Clip Rectangle X Maximum Register	TDGC_CXMAXR	Read-write	0x00000000
0x4C	Clip Rectangle Y Minimum Register	TDGC_CYMINR	Read-write	0x00000000
0x50	Clip Rectangle Y Maximum Register	TDGC_CYMAXR	Read-write	0x00000000
0x54	Graphics Status Register	TDGC_GSR	Read-write	0x00000000
0x58	VRAM Size Register	TDGC_VSR	Read-write	0x00000000
0x60	Reserved			0x00000000
0x64	Graphics Interrupt Register	TDGC_GIR	Read-write	0x00000000
0x68	Graphics Interrupt Mask Register	TDGC_GIMR	Read-write	0x00000000
0x70	Bits Per Pixel Register	TDGC_BPPR	Read-write	0x00000000
0x78	Command Queue Count Register	TDGC_CQCR	Read-write	0x00000000
0x7C	Command Queue Status Register	TDGC_CQSR	Read-write	0x00000000
0x80	Command Queue Register	TDGC_CQR	Read-write	0x00000000
0x90	Reserved			0x00000000
...	...	...	...	...
0xCC	Reserved			0x00000000
0x120	Reserved			0x00000000
...	...	...	...	...

Table 45-2. TDGC Register Mapping

Offset	Register	Name	Access	Reset
0x15C	Reserved		Read-write	0x00000000
0x200	VRAM Offset Register	TDGC_VOR	Read-write	0x00000000
0x204	Data Format Register	TDGC_DFR	Read-write	0x00000000
0xXX - 0xFC	Reserved	TDGC_RES	–	–

45.5.1 Block Transfer Size X Register

Name: TDGC_BTSXR

Address:0xFFFC8000

Access: Read-write

Reset Value: 0x00000000

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

- **XSIZE**

Sets size X of a bit block transfer in pixel units.

45.5.2 Block Transfer Size Y Register

Name: TDGC_BTSYR

Address:0xFFFC8004

Access: Read-write

Reset Value: 0x00000000

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

- **YSIZE**

Sets size Y of a bit block transfer in pixel units.

45.5.3 Source/Begin X Register

Name: TDGC_SBXR

Address:0xFFFC8008

Access: Read-write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
XSRC							

23	22	21	20	19	18	17	16
XSRC							

15	14	13	12	11	10	9	8
XSRC							

7	6	5	4	3	2	1	0
XSRC							

- **XSRC**

XSRC[10 - 0]: Sets source X of a bit block transfer/begin point X of line draw in pixel units.

XSRC[31 - 0]: Sets begin point X of clipped line draw in pixel units.

45.5.4 Source/Begin Y Register

Name: TDGC_SBYR

Address:0xFFFC800C

Access: Read-write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
YSRC							

23	22	21	20	19	18	17	16
YSRC							

15	14	13	12	11	10	9	8
YSRC							

7	6	5	4	3	2	1	0
YSRC							

- **YSRC**

YSRC[10 - 0]: Sets source Y of a bit block transfer/begin point Y of line draw in pixel units.

YSRC[31 - 0]: Sets begin point Y of clipped line draw in pixel units.

45.5.5 Target/End X Register

Name: TDGC_TEXR

Address:0xFFFC8010

Access: Read-write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
XEND							

23	22	21	20	19	18	17	16
XEND							

15	14	13	12	11	10	9	8
XEND							

7	6	5	4	3	2	1	0
XEND							

- **XEND**

XEND[10 - 0]: Sets source Y of a bit block transfer/begin point Y of line draw in pixel units.

XEND[31 - 0]: Sets begin point Y of clipped line draw in pixel units.

45.5.6 Target/End Y Register

Name: TDGC_TEYR

Address:0xFFFC8014

Access: Read-write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
YEND							

23	22	21	20	19	18	17	16
YEND							

15	14	13	12	11	10	9	8
YEND							

7	6	5	4	3	2	1	0
YEND							

- **YEND**

YEND[10 - 0]: Sets source Y of a bit block transfer/begin point Y of line draw in pixel units.

YEND[31 - 0]: Sets begin point Y of clipped line draw in pixel units.

45.5.7 Line Width Register

Name: TDGC_LWR

Address:0xFFFC8018

Access: Read-write

Reset Value: 0x00000000

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

- **LWD**

Line width in pixel units.

45.5.8 Line Pattern Register

Name: TDGC_LPR

Address:0xFFFC801C

Access: Read-write

Reset Value: 0x00000000

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

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

15	14	13	12	11	10	9	8
LPT							

7	6	5	4	3	2	1	0
LPT							

- **LPT: Line Pattern**

Sets 16-bit 1D pattern or 4 x 4-bit 2D pattern for line drawing.

In 1D pattern drawing, LPT[0] is the starting point of the pattern. After each operation, LPT will rotate one bit to the right. In 2D pattern drawing, the bit pattern does not rotate. The operation is determined on the last 2 bits of X and Y addresses as shown in the table below.

	X[1:0]=0	X[1:0]=1	X[1:0]=2	X[1:0]=3
Y[1:0]=0	LPT[12]	LPT[13]	LPT[14]	LPT[15]
Y[1:0]=0	LPT[8]	LPT[9]	LPT[10]	LPT[11]
Y[1:0]=0	LPT[4]	LPT[5]	LPT[6]	LPT[7]
Y[1:0]=0	LPT[0]	LPT[1]	LPT[2]	LPT[3]

45.5.9 Color Select Register

Name: TDGC_CSR

Address:0xFFFC8020

Access: Read-write

Reset Value: 0x00000000

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

- **CLR: Color**

At 1 bpp only CLR[3] is active.

0: white is selected.

1: black is selected.

At 2 bpp only CLR[3:2] is active.

Values	Color
00	white
01	light grey
10	dark grey
11	black

At 4 bpp, only CLR[3:0] is active. It selects one of the 16 grey shades for monochrome displays or one of the 16 simultaneous colors available.

At 8 bpp, only CLR[7:0] is active. It selects one the 256 simultaneous colors available.

At 16 bpp, only CLR[15:0] is active. It selects one of the 32768 colors available.

At 24 bpp CLR[23:0] is active. It selects one of the 16M colors available.

45.5.10 Logic Operation Register

Name: TDGC_LOR

Address:0xFFFC8024

Access: Read-write

Reset Value: 0x00000000

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

- **LOC: Logic Operation Code**

Valid logic function code

LOC	Function
0000	Write (MOV)
0001	OR
0010	AND
0011	XOR
0100	NOT
0101	NOR
0110	NAND
0111	XNOR

45.5.11 Graphics Operation Register

Name: TDGC_GOR

Address:0xFFFC8028

Access: Read-write

Reset Value: 0x00000000

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
GOC				OP3	OP2	OP1	OP0

- **GOC: Graphic Operation Code**

GOC	Function
0000	No Operation
1101	Line Drawing
1011	Block Transfer

- **OPx: Option**

There are four options, each one allows each operation code to behave as shown below.

	0	1
OP3	No Transformation	Transformation
OP2	Relative	Absolute
OP1	No Update X,Y	Update X,Y
OP0	2D Pattern	1D Pattern

	0	1
OP3	No Transformation	Transformation
OP2	Relative	Absolute
OP1	No Update X	Update X
OP0	No Update Y	Update

Note: Transformation bit is reserved for future use.

The table below gives the possible operation depending on command code = (TDGC_LOR) | (TDGC_GOR << 8)

Command Code[15:0]	Function
0x00	No Operation
0xB00	Block Transfer, Relative, No Update, MOV
0xB10	Block Transfer, Relative, Update Y, MOV
0xB20	Block Transfer, Relative, Update X, MOV
0xB40	Block Transfer, Absolute, No Update, MOV
0xB50	Block Transfer, Absolute, Update Y, MOV
0xB60	Block Transfer, Absolute, Update X, MOV
0xB01	Block Transfer, Relative, No Update, OR
0xB11	Block Transfer, Relative, Update Y, OR
0xB21	Block Transfer, Relative, Update X, OR
0xB41	Block Transfer, Absolute, No Update, OR
0xB51	Block Transfer, Absolute, Update Y, OR
0xB61	Block Transfer, Absolute, Update X, OR
0xB02	Block Transfer, Relative, No Update, AND
0xB12	Block Transfer, Relative, Update Y, AND
0xB22	Block Transfer, Relative, Update X, AND
0xB42	Block Transfer, Absolute, No Update, AND
0xB52	Block Transfer, Absolute, Update Y, AND
0xB62	Block Transfer, Absolute, Update X, AND
0xB03	Block Transfer, Relative, No Update, XOR
0xB13	Block Transfer, Relative, Update Y, XOR
0xB23	Block Transfer, Relative, Update X, XOR
0xB43	Block Transfer, Absolute, No Update, XOR
0xB53	Block Transfer, Absolute, Update Y, XOR
0xB63	Block Transfer, Absolute, Update X, XOR
0xD00	Line Drawing, Relative, No Update, 2D Pattern
0xD10	Line Drawing, Relative, No Update, 1D Pattern
0xD20	Line Drawing, Relative, Update X and Y, 2D Pattern
0xD30	Line Drawing, Relative, Update X and Y, 1D Pattern
0xD40	Line Drawing, Absolute, No Update, 2D Pattern
0xD50	Line Drawing, Absolute, No Update, 1D Pattern
0xD60	Line Drawing, Absolute, Update X and Y, 2D Pattern
0xD70	Line Drawing, Absolute, Update X and Y, 1D Pattern

45.5.12 Extended Begin X Register

Name: TDGC_EBXR

Address:0xFFFC802C

Access: Read-write

Reset Value: 0x00000000

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

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

15	14	13	12	11	10	9	8
EXT_BX							

7	6	5	4	3	2	1	0
EXT_BX							

- **EXT_BX**

Sets begin point x (MSB[31:16]) of line draw in pixel units. This register is only for clipped line draw when command queue is used for drawing. Since the interface to the command queue is only 16, the MSB[31:16] is written to a separate register.

45.5.13 Extended Begin Y Register

Name: TDGC_EBYR

Address:0xFFFC8030

Access: Read-write

Reset Value: 0x00000000

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

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

15	14	13	12	11	10	9	8
EXT_BY							

7	6	5	4	3	2	1	0
EXT_BY							

- **EXT_BY**

Sets begin point y (MSB[31:16]) of line draw in pixel units. This register is only for clipped line draw when command queue is used for drawing. Since the interface to the command queue is only 16, the MSB[31:16] is written to a separate register.

45.5.14 Extended End X Register

Name: TDGC_EEXR

Address:0xFFFC8034

Access: Read-write

Reset Value: 0x00000000

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

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

15	14	13	12	11	10	9	8
EXT_EX							

7	6	5	4	3	2	1	0
EXT_EX							

- **EXT_EX**

Sets end point x (MSB[31:16]) of line draw in pixel units. This register is only for clipped line draw when command queue is used for drawing. Since the interface to the command queue is only 16, the MSB[31:16] is written to a separate register.

45.5.15 Extended End Y Register

Name: TDGC_EEYR

Address:0xFFFC8038

Access: Read-write

Reset Value: 0x00000000

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

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

15	14	13	12	11	10	9	8
EXT_EY							

7	6	5	4	3	2	1	0
EXT_EY							

- **EXT_EY**

Sets end point y (MSB[31:16]) of line draw in pixel units. This register is only for clipped line draw when command queue is used for drawing. Since the interface to the command queue is only 16, the MSB[31:16] is written to a separate register.

45.5.16 Extended Color Select Register

Name: TDGC_ECSR

Address:0xFFFC803C

Access: Read-write

Reset Value: 0x00000000

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

- **EXT_CSR**

Sets MSB[23:16] of color selection for 24 bpp. This register is only used when in 24 bpp mode and when command queue is used for drawing. Since the interface to the command queue is only 16, the MSB[31:24] is written to a separate register.

45.5.17 Clip Control Register

Name: TDGC_CCR

Address:0xFFFC8040

Access: Read-write

Reset Value: 0x00000000

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

- **CEN**

1: Enable clipping.

0: Disable clipping.

45.5.18 Clip Rectangle Minimum X Register

Name: TDGC_CXMINR

Address:0xFFFC8044

Access: Read-write

Reset Value: 0x000000F0

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

- **CXMIN**

Minimum x-coordinate boundary of the clip rectangle.

45.5.19 Clip Rectangle Maximum X Register

Name: TDGC_CXMAXR

Address:0xFFFC8048

Access: Read-write

Reset Value: 0x000000F0

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

- **CXMAX**

Maximum x-coordinate boundary of the clip rectangle.

45.5.20 Clip Rectangle Minimum Y Register

Name: TDGC_CYMINR

Address:0xFFFC804C

Access: Read-write

Reset Value: 0x000000F0

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

- **CYMIN**

Minimum y-coordinate boundary of the clip rectangle.

45.5.21 Clip Rectangle Maximum Y Register

Name: TDGC_CYMAXR

Address:0xFFFC8050

Access: Read-write

Reset Value: 0x000000F0

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

- **CYMAX**

Maximum y-coordinate boundary of the clip rectangle.

45.5.22 Graphics Status Register

Name: TDGC_GSR

Address:0xFFFC8054

Access: Read-write

Reset Value: 0x00000000

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

- **LTB**

1: Line drawing engine is busy.

0: Line drawing engine is available.

- **BTB**

1: Block transfer engine busy.

0: Block transfer engine available.

45.5.23 VRAM Size Register

Name: TDGC_VSR

Address:0xFFFC8058

Access: Read-write

Reset Value: 0x000000001VSIZE

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	VSIZE			

- **VSIZE**

Value	Description
0x0	Row offset (Y) = 512; VRAM type = 8-bit wide data bus
0x1	Row offset (Y) = 1024
0x2	Row offset (Y) = 2048
0x3	Row offset (Y) = 256
0x4	Row offset (Y) = 512; VRAM type = 16-bit wide data bus
0x5	Row offset (Y) = 1024
0x6	Row offset (Y) = 2048
0x7	Row offset (Y) = 256
0x8	Row offset (Y) = 512; VRAM type = 32-bit wide data bus
0x9	Row offset (Y) = 1024
0xA	Row offset (Y) = 2048
0xB	Row offset (Y) = 256

45.5.24 Graphics Interrupt Register

Name: TDGC_GIR

Address:0xFFFC8064

Access: Read-write

Reset Value: 0x00000000

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

- **BUFE: Command queue buffer empty interrupt**

1: Signals buffer empty. Writing a 1 to this bit clears the interrupt.

45.5.25 Graphics Interrupt Mask Register

Name: TDGC_GIMR

Address:0xFFFC8068

Access: Read-write

Reset Value: 0x00000000

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

- **BUFE: Command queue buffer empty interrupt enable**

1: Enable command queue buffer empty interrupt.

0: Disable command queue buffer empty interrupt.

45.5.26 Bits Per Pixel Register

Name: TDGC_BPPR

Address:0xFFFC8070

Access: Read-write

Reset Value: 0x00000000

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	BPP		

- **BPP: Bits Per Pixel**

BPP	Bits per pixel
0	1 BPP
1	2 BPP
2	4 BPP
3	8 BPP
4	16 BPP
5	24 BPP

45.5.27 Command Queue Count Register

Name: TDGC_CQCR

Address:0xFFFC8078

Access: Read-only

Reset Value: 0x00000000

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

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

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

7	6	5	4	3	2	1	0
BUFW_CNTR		BUFR_CNTR					

- **BUFR_CNTR**

Number of half words read from the FIFO by internal logic.

- **BUFW_CNTR**

Number of half words written to the FIFO by CPU.

45.5.28 Command Queue Status Register

Name: TDGC_CQSR

Address:0xFFFC807C

Access: Read-only

Reset Value: 0x00000000

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

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

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	BE

- **BE: Buffer Empty**

1: Buffer is empty.

0: Buffer is not empty.

45.5.29 Command Queue Register

Name: TDGC_CQR

Address:0xFFFC8080

Access: Write-only

Reset Value: 0x00000000

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

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

15	14	13	12	11	10	9	8
Q_DATA							

7	6	5	4	3	2	1	0
Q_DATA							

- **Q_DATA**

All data to be stored in the queue is written to this register.

45.5.30 VRAM OFFSET Register

Name: TDGC_VOR

Address:0xFFFC8200

Access: Read-write

Reset Value: 0x00000000

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

23	22	21	20	19	18	17	16
–	–	–	–	–	–	S8	PK

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

7	6	5	4	3	2	1	0
OFFSET							

- **OFFSET**

.Offset into the VRAM which gives the first pixel position in the memory.

- **PK: Packed Mode**

1: Selects the packed 24 bpp mode.

- **S8**

1: Allows the use of 320 pixel offset for rows (y). This special mode allows the use of 80K internal RAM for frame size of 240 x 320 at 8 bpp.

45.5.31 DATA Format Register

Name: TDGC_DFR

Address:0xFFFC8204

Access: Read-write

Reset Value: 0x00000000

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

- **ENDIAN: ENDIANESS**

1: DATA format is little endian.

0: DATA format is big endian.

- **WINCE**

1: DATA format is WINCE compliant.

0: DATA format is not WINCE compliant.

46. Image Sensor Interface (ISI)

46.1 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 46-3 on page 998).

Several input formats such as preprocessed RGB or YCbCr are supported through the data bus interface.

It supports two modes of synchronization:

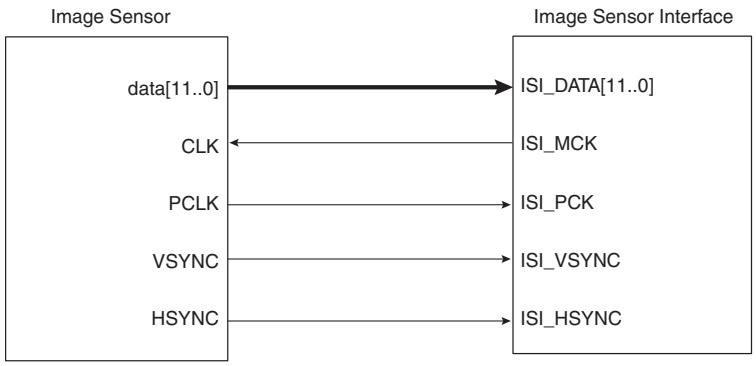
1. The hardware with ISI_VSYNC and ISI_HSYNC signals
2. The International Telecommunication Union Recommendation *ITU-R BT.656-4* Start-of-Active-Video (SAV) and End-of-Active-Video (EAV) synchronization sequence.

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 46-1. I/O Description

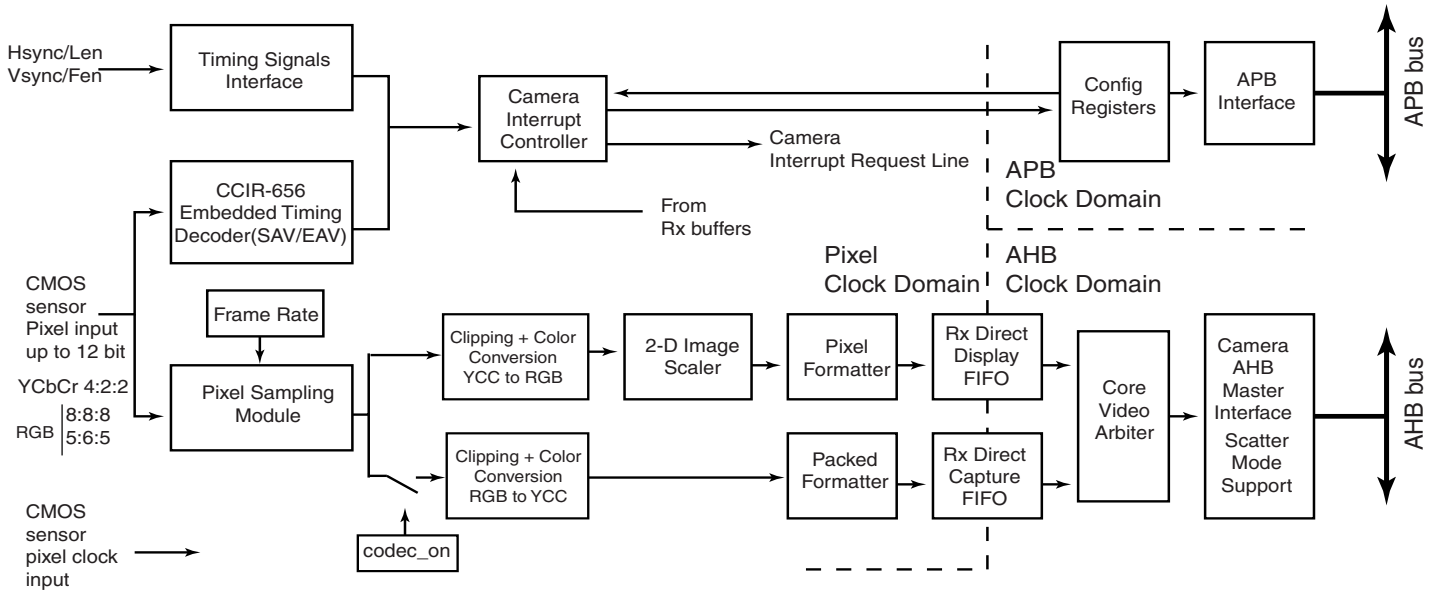
Signal	Dir	Description
ISI_VSYNC	IN	Vertical Synchronization
ISI_HSYNC	IN	Horizontal Synchronization
ISI_DATA[11..0]	IN	Sensor Pixel Data
ISI_MCK	OUT	Master Clock Provided to the Image Sensor
ISI_PCK	IN	Pixel Clock Provided by the Image Sensor

Figure 46-1. ISI Connection Example



46.2 Block Diagram

Figure 46-2. Image Sensor Interface Block Diagram



46.3 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.

46.3.1 Data Timing

The two data timings using horizontal and vertical synchronization and EAV/SAV sequence synchronization are shown in Figure 46-3 and Figure 46-4.

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 46-3. HSYNC and VSYNC Synchronization

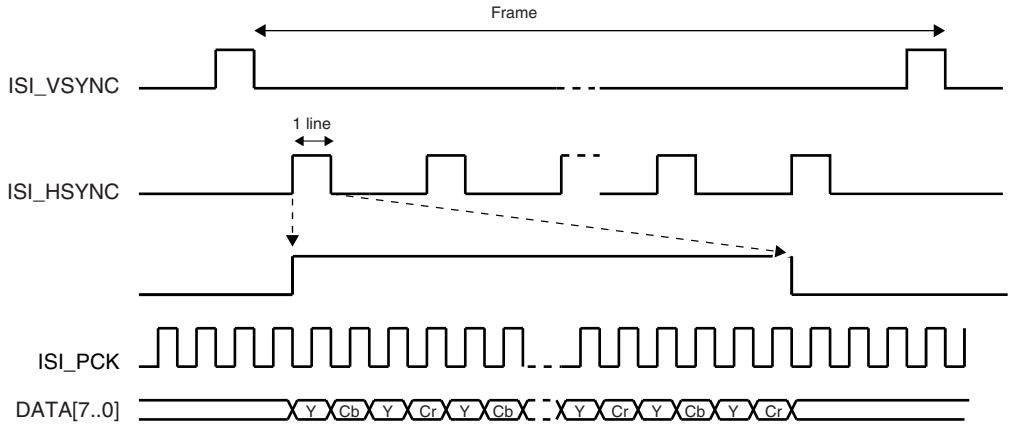
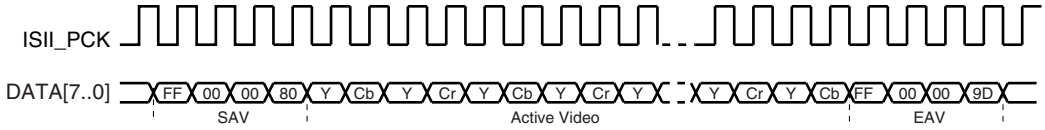


Figure 46-4. SAV and EAV Sequence Synchronization



46.3.2 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 46-2. 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 46-3. 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 46-4. 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 46-5. 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.

46.3.3 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.

46.3.4 Preview Path

46.3.4.1 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 46-6. 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 46-7. Decimation and Scaler Offset Values

INPUT OUTPUT		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:

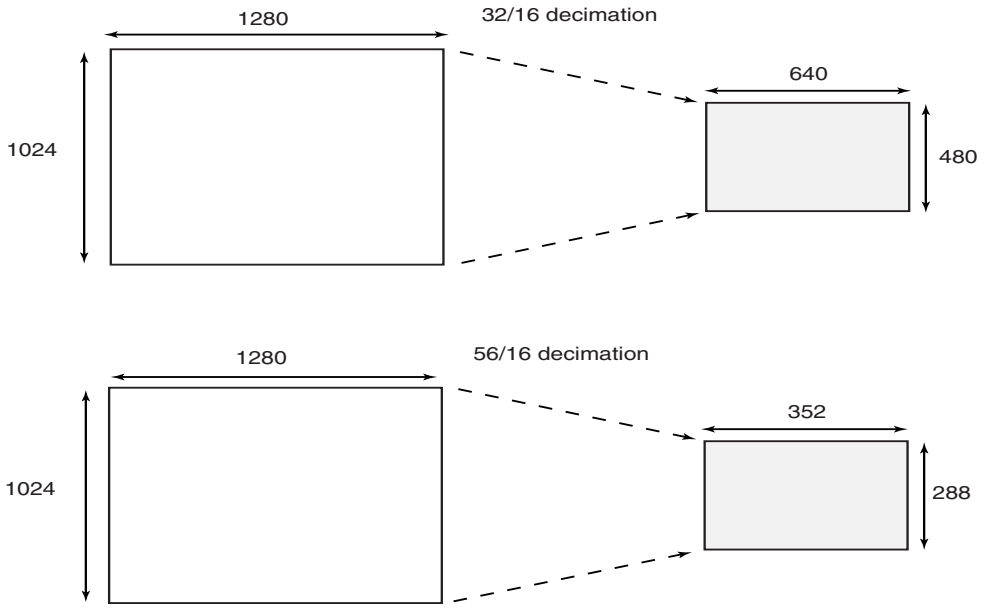
Input 1280*1024 Output=640*480

Hratio = 1280/640 =2

Vratio = 1024/480 =2.1333

The decimation factor is 2 so 32/16.

Figure 46-5. Resize Examples



46.3.4.2 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:

$$\begin{cases} R = Y + 1.596 \cdot V \\ G = Y - 0.394 \cdot U - 0.436 \cdot V \\ B = Y + 2.032 \cdot U \end{cases}$$

46.3.4.3 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 46-8. 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

46.3.4.4 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.

46.3.4.5 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.

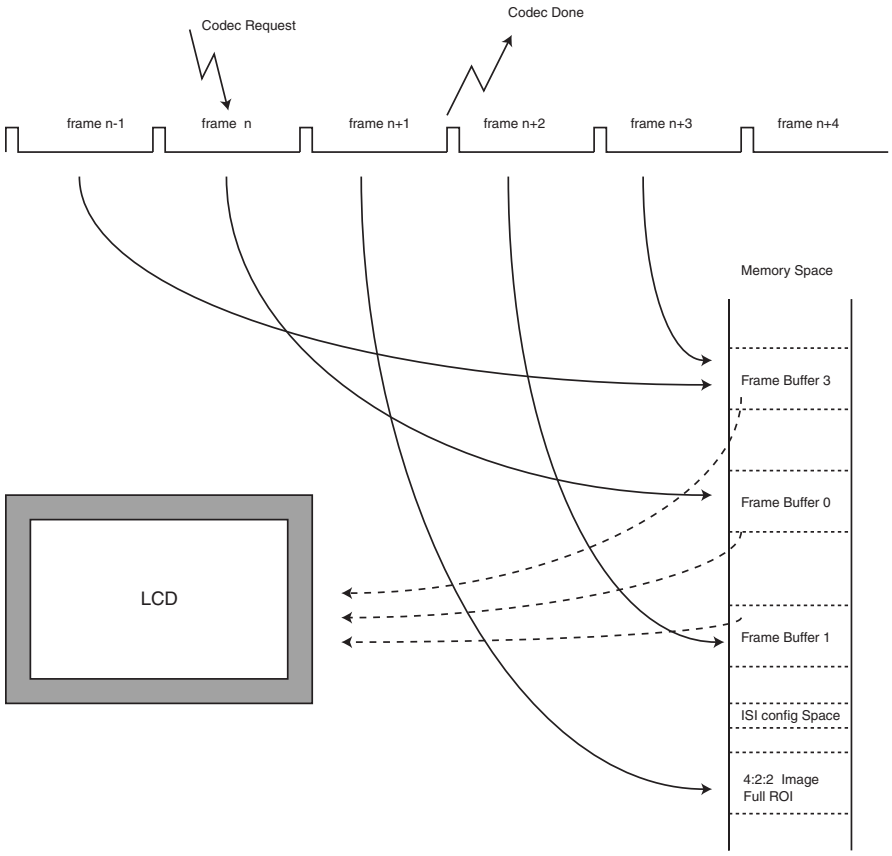
Destination Address: frame buffer ID1 0x3A000

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 46-6 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 46-6. Three Frame Buffers Application and Memory Mapping



46.3.5 Codec Path

46.3.5.1 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:

$$\begin{cases} Y = 0.257 \cdot R + 0.504 \cdot G + 0.098 \cdot B + 16 \\ C_r = 0.439 \cdot R - 0.368 \cdot G - 0.071 \cdot B + 128 \\ C_b = -0.148 \cdot R - 0.291 \cdot G + 0.439 \cdot B + 128 \end{cases}$$

46.3.5.2 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.

46.3.5.3 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.

46.4 Image Sensor Interface (ISI) User Interface

Table 46-9. 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.

46.4.1 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 46.4.3 “ISI Status Register” on page 1010 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 1010.

- **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.

46.4.2 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.

46.4.3 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.

46.4.4 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.

46.4.5 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.

46.4.6 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.

46.4.7 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).

46.4.8 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.

46.4.9 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.

46.4.10 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.

46.4.11 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

46.4.12 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

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

46.4.13 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

46.4.14 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

46.4.15 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

47. AT91SAM9263 Electrical Characteristics

47.1 Absolute Maximum Ratings

Table 47-1. Absolute Maximum Ratings*

Operating Temperature (Industrial)-40· C to +125· C
Storage Temperature-60°C to +150°C
Voltage on Input Pins with Respect to Ground-0.3V to +4.0V
Maximum Operating Voltage (VDDCORE and VDDBU)1.5V
Maximum Operating Voltage (VDDOSC, VDDPLL, VDDIOMx and VDDIOPx)4.0V
Total DC Output Current on all I/O lines500 mA

*NOTICE: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

47.2 DC Characteristics

The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C, unless otherwise specified.

Table 47-2. DC Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{VDDCORE}$ (1)	DC Supply Core		1.08		1.32	V
V_{VDDBU}	DC Supply Backup		1.08		1.32	
V_{VDDOSC}	DC Supply Oscillator		3.0		3.6	V
V_{VDDPLL}	DC Supply PLL		3.0		3.6	V
$V_{VDDIOM0}$	DC Supply Memory 0 I/Os		1.65		1.95	V
			3.0		3.6	V
$V_{VDDIOM1}$	DC Supply Memory 1 I/Os		1.65		1.95	V
			3.0		3.6	V
$V_{VDDIOP0}$ (1)	DC Supply Peripheral 0 I/Os		3.0		3.6	V
$V_{VDDIOP1}$ (1)	DC Supply Peripheral 1 I/Os		1.65		3.6	V
V_{IL}	Input Low-level Voltage	V_{VDDIO} from 3.0V to 3.6V	-0.3		0.8	V
		V_{VDDIO} from 1.65V to 1.95V	-0.3		0.3 x V_{VDDIO}	V
V_{IH}	Input High-level Voltage	V_{VDDIO} from 3.0V to 3.6V	2.0		V_{VDDIO} +0.3V	V
		V_{VDDIO} from 1.65V to 1.95V	0.7 x V_{VDDIO}		V_{VDDIO} +0.3V	V
V_{OL}	Output Low-level Voltage	I_O max, V_{VDDIO} from 3.0V to 3.6V			0.4	V
		CMOS (I_O <0.3 mA) V_{VDDIO} from 1.65V to 1.95V			0.1	V
		TTL (I_O Max) V_{VDDIO} from 1.65V to 1.95V			0.4	V

Table 47-2. DC Characteristics (Continued)

Symbol	Parameter	Conditions		Min	Typ	Max	Units
V_{OH}	Output High-level Voltage	I_O Max, V_{VDDIO} from 3.0V to 3.6V		V_{VDDIO} - 0.4			V
		CMOS (I_O <0.3 mA) V_{VDDIO} from 1.65V to 1.95V		V_{VDDIO} - 0.1			V
		TTL (I_O Max) V_{VDDIO} from 1.65V to 1.95V		V_{VDDIO} - 0.4			V
R_{PULLUP}	Pull-up Resistance	PA0-PA31, PB0-PB31, PC0-PC31, PD0-PD31, PE0-PE31		70	100	175	kOhm
I_O	Output Current	PA0-PA31, PB0-PB31, PC0-PC31, PD0-PD31, PE0-PE31				8	mA
I_{SC}	Static Current	On $V_{VDDCORE}$ = 1.2V, MCK = 0 Hz	T_A =25°C		240		µA
		All inputs driven TMS, TDI, TCK, NRST = 1	T_A =85°C			3500	
		On V_{VDDBU} = 1.2V, Logic cells consumption	T_A =25°C		3		µA
		All inputs driven WKUP = 0	T_A =85°C			17	

Note: 1. Even during startup V_{VDDIOP} must always be superior or equal to $V_{VDDCORE}$.

47.3 Power Consumption

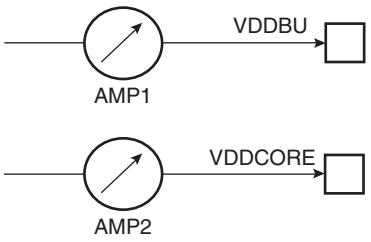
- Power consumption of power supply in four different modes: Full Speed (PCK and MCK present), Idle (only MCK present), Quasi Static (the system is running at 500 Hz), Backup (in Shutdown Mode)
- Power consumption by peripheral: calculated as the difference in current measurement after having enabled then disabled the corresponding clock.

47.3.1 Power Consumption versus Modes

The values in Table 47-3 and Table 47-4 on page 1029 are estimated values of the power consumption with operating conditions as follows:

- $V_{DDIOM0} = V_{DDIOM1} = V_{DDIOP0} = V_{DDIOP1} = 3.3$ V
- $V_{DDPLL} = V_{DDOSC} = 3.3$V
- There is no consumption on the I/Os of the device
- All peripheral clocks are deactivated

Figure 47-1. Measures Schematics



These figures represent the power consumption measured on the power supplies.

Table 47-3. Power Consumption for Different Modes

Mode	Conditions	Consumption	Unit
Full speed (PCK and MCK present)	ARM Core clock is 198 MHz. MCK is 96 MHz. Dhrystone running in Icache. V_{DDCORE} = 1.08V T_A = 85° C onto AMP2	55.9	mA
	ARM Core clock is 240 MHz. MCK is 120 MHz. Dhrystone running in Icache. V_{DDCORE} = 1.2V T_A = 85° C onto AMP2	73.3	
	ARM Core clock is 240 MHz. MCK is 120 MHz. Dhrystone running in Icache. V_{DDCORE} = 1.2V T_A = 25° C onto AMP2	70.9	
Idle[1] (only MCK present)	MCK is 96 MHz. ARM core in idle state, waiting an interrupt. Processor clock disabled V_{DDCORE} = 1.08V T_A = 85° C onto AMP2	20.2	mA
	MCK is 96 MHz. ARM core in idle state, waiting an interrupt. Processor clock disabled V_{DDCORE} = 1.2V T_A = 85° C onto AMP2	22.7	
	MCK is 96 MHz. ARM core in idle state, waiting an interrupt. Processor clock disabled V_{DDCORE} = 1.2V T_A = 25° C onto AMP2	19.5	

Table 47-3. Power Consumption for Different Modes (Continued)

Mode	Conditions	Consumption	Unit
Quasi Static (system running at 500 Hz)	ARM Core clock is 500 Hz. MCK is 500 Hz V_{DDCORE} = 1.08V T_A = 85° C onto AMP2	2720	µA
	ARM Core clock is 500 Hz. MCK is 500 Hz V_{DDCORE} = 1.2V T_A = 85° C onto AMP2	3080	
	ARM Core clock is 500 Hz. MCK is 500 Hz V_{DDCORE} = 1.2V T_A = 25° C onto AMP2	248	
Backup (in Shutdown Mode)	In Shutdown Mode V_{DDBU} = 1.08V T_A = 85° C onto AMP1	14.8	µA
	In Shutdown Mode V_{DDBU} = 1.2V T_A = 85° C onto AMP1	16.9	
	In Shutdown Mode V_{DDBU} = 1.2V T_A = 25° C onto AMP1	3.4	

Note: 1. No SRAM access in Idle Mode.

Table 47-4. Power Consumption by Peripheral onto AMP2 (T_A = 25° C, V_{DDCORE} = 1.2V)

Peripheral	Consumption	Unit
PIO Controller A or B	5	µA/MHz
PIO Controller C to E	14	
USART	13	
UHP	12	
UDP	9	
TWI	2	
SPI	9	
MCI	13	
SSC	16	
Timer Counter Channels	8	
CAN	50	
PWMC	7	
EMAC	40	
LCDC	45	
Image Sensor Interface	8	
AC97	13	

47.4 Core Power Supply POR Characteristics

The AT91SAM9263 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 prevent the device from booting.

47.4.1 Power-up Sequence

Table 47-5. Power-On-Reset (POR) Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{th+}	Threshold Voltage Rising	Minimum Slope of +2.0V/30ms	0.70	0.85	1.00	V
V_{th-}	Threshold Voltage Falling		0.60	0.75	0.9	V
T_{RES}	Reset Time		80	140	200	µs

VDDBU powers the Backup power switch; it must be always powered to ensure correct behavior.

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.

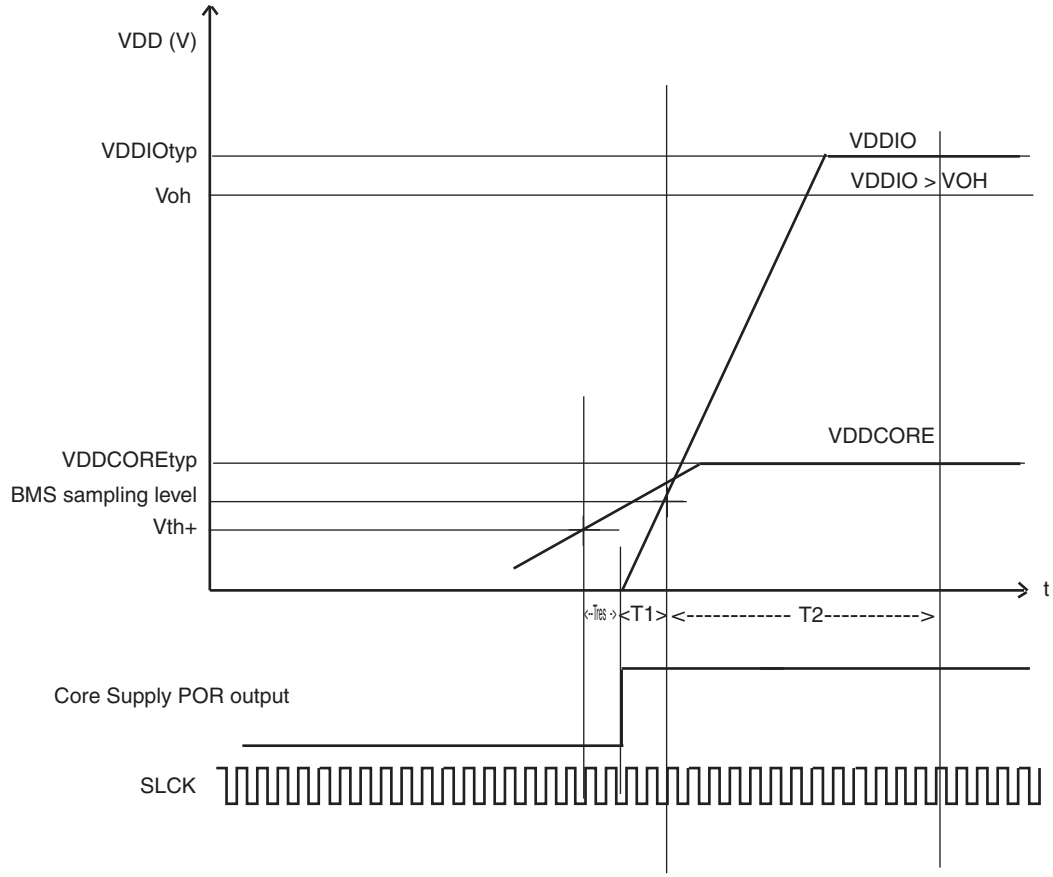
- VDDIOM0, VDDIOM1, VDDIOP0, VDDIOP1 and VDDIOP2 must NOT be powered until VDDCORE has reached a level superior to V_{th+}.

- VDDIOP0 must be ≥ to V_{IH} (refer to Table 47-2, “DC Characteristics” for more details) within (Tres + T1) after VDDCORE reached V_{th+}.
- VDDIOM0 and VDDIOPM1 must reach V_{OH} (refer to DC characteristics table for more details) within (Tres +T1 +T2) after VDDCORE has reached V_{th+}
  - T_{RES} is a POR characteristic
  - T1 = 3 x T_{SLCK}
  - T2 = 1 6 x T_{SLCK}

As T_{SLCK} is the period of the external 32.768 kHz oscillator.

  - T_{RES} = 80 µs
  - T1 = 91.5 µs
  - T2 = 488 µs

Figure 47-2. VVDDCORE and VVDDIO Constraints at Startup



47.4.2 Power Down Sequence

Switch-off the VDDIOMx and VDDIOPx power supply prior to or at the same time as VDDCORE.

No power-up or power-down restrictions apply to other power supplies.

47.5 Clock Characteristics

These parameters are given in the specified conditions.

47.5.1 Processor Clock Characteristics

Table 47-6. Processor Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Unit
$1/(t_{CPPCK})$	Processor Clock Frequency	VDDCORE = 1.1V T = 85°C		200	MHz
$1/(t_{CPPCK})$	Processor Clock Frequency	VDDCORE = 1.2V T = 85°C		240	MHz

47.5.2 Master Clock Characteristics

Table 47-7. Master Clock Waveform Parameters

Symbol	Parameter	Conditions	Min	Max	Unit
$1/(t_{CPMCK})$	Master Clock Frequency	VDDCORE = 1.1V T = 85°C		100	MHz
$1/(t_{CPMCK})$	Master Clock Frequency	VDDCORE = 1.2V T = 85°C		120	MHz

47.5.3 XIN Clock Characteristics

Table 47-8. XIN Clock Electrical Characteristics

Symbol	Parameter	Conditions	Min	Max	Unit
$1/(t_{CPXIN})$	XIN Clock Frequency			50.0	MHz
t_{CPXIN}	XIN Clock Period		20.0		ns
t_{CHXIN}	XIN Clock High Half-period		0.4 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)		25	pF
R_{IN}	XIN Pulldown Resistor	(1)		500	kΩ
V_{IN}	XIN voltage	(1)		3.3	V

Notes: 1. These characteristics apply only when the Main Oscillator is in bypass mode (i.e., when MOSCEN = 0 and OSCBYPASS = 1 in the CKGR_MOR register).

47.5.4 I/Os

Criteria used to define the maximum frequency of the I/Os:

- output duty cycle (40%-60%)
- minimum output swing: 100 mV to VDDIO - 100 mV

- Addition of rising and falling time inferior to 75% of the period

Table 47-9. I/O Characteristics

Symbol	Parameter	Conditions	Min	Max	Unit
FreqMax	VDDIOP0 powered Pins frequency	3.3V domain (1)		100	MHz
FreqMax	VDDIOP1 powered Pins frequency	3.3V domain (1)		100	MHz
		2.5V domain (2)		91	MHz
		1.8V domain (3)		66	MHz

Notes:
1. 3.3V domain: V_{VDDIOP} from 3.0V to 3.6V, maximum external capacitor = 40 pF
2. 2.5V domain: V_{VDDIOP} from 2.3V to 2.7V, maximum external capacitor = 30 pF
3. 1.8V domain: V_{VDDIOP} from 1.65V to 1.95V, maximum external capacitor = 20 pF

47.6 Crystal Oscillator Characteristics

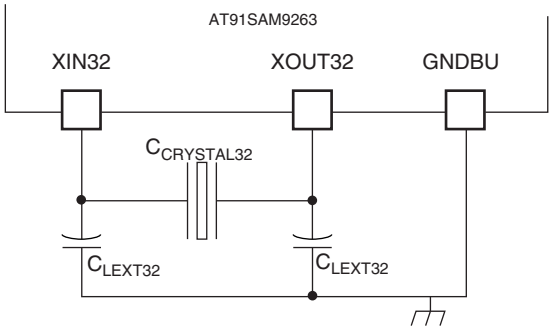
The following characteristics are applicable to the operating temperature range: T_A = -40°C to 85°C and worst case of power supply, unless otherwise specified.

47.6.1 32 kHz Oscillator Characteristics

Table 47-10. 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	$C_{CRYSTAL32}$ = 6pF(3)		4		pF
		$C_{CRYSTAL32}$ = 12.5pF(3)		17		pF
	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 user load capacitance should be subtracted from C_{LEXT32}.



47.6.2 32 kHz Crystal Characteristics

Table 47-11. 32 kHz Crystal Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
ESR	Equivalent Series Resistor Rs	Crystal @ 32.768 kHz		50	100	kΩ
C_M	Motional Capacitance	Crystal @ 32.768 kHz	0.6		3	fF
C_S	Shunt Capacitance	Crystal @ 32.768 kHz	0.6		2	pF

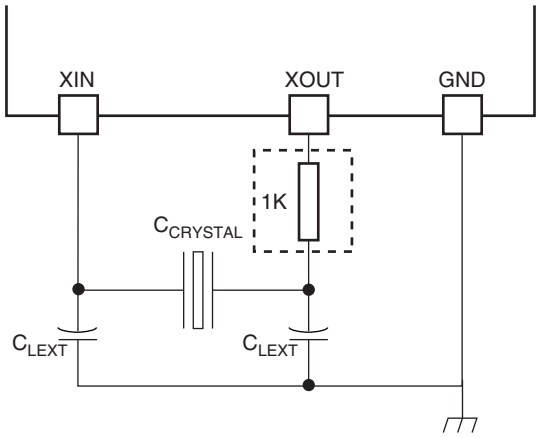
47.6.3 Main Oscillator Characteristics

Table 47-12. Main Oscillator Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$1/(t_{CPMAIN})$	Crystal Oscillator Frequency		3	16	20	MHz
$C_{CRYSTAL}$	Crystal Load Capacitance		12.5	15	17.5	pF
C_{LEXT}[7]	External Load Capacitance	$C_{CRYSTAL}$ = 12.5 pF[6]		15		pF
		$C_{CRYSTAL}$ = 15 pF[6]		18		
		$C_{CRYSTAL}$ = 17.5 pF[6]		22		
	Duty Cycle		40	50	60	%
t_{ST}	Startup Time	V_{DDPLL} = 3 to 3.6V C_S = 3 pF[1] $1/(t_{CPMAIN})$ = 3 MHz C_S = 7 pF[1] $1/(t_{CPMAIN})$ = 8 MHz C_S = 7 pF[1] $1/(t_{CPMAIN})$ = 16 MHz C_S = 7 pF[1] $1/(t_{CPMAIN})$ = 20 MHz			20 4 2 2	ms
I_{DDST}	Standby Current Consumption	Standby mode			1	µA
P_{ON}	Drive Level	@ 3 MHz @ 8 MHz @ 16 MHz @ 20 MHz			15 30 50 50	µW
$I_{DD\ ON}$	Current Dissipation	@ 3 MHz[2] @ 8 MHz[3] @ 16 MHz[4] @ 20 MHz[5]		150 300 300 450	250 530 530 650	µA

Notes:
1. C_S is the shunt capacitance.
2. R_S = 100 to 200 Ω; C_S = 2.0 to 2.5 pF; C_M = 2 to 1.5 fF (typ, worst case) using 1 kΩ serial resistor on XOUT.
3. R_S = 50 to 100 Ω; C_S = 2.0 to 2.5 pF; C_M = 4 to 3 fF (typ, worst case).
4. R_S = 25 to 50 Ω; C_S = 2.5 to 3.0 pF; C_M = 7 to 5 fF (typ, worst case).
5. R_S = 20 to 50 Ω; C_S = 3.2 to 4.0 pF; C_M = 10 to 8 fF (typ, worst case).
6. Additional user load capacitance should be subtracted from C_{LEXT}.
7. C_{LEXT} is determined by taking into account internal parasitic and package load capacitance.

AT91SAM9263



47.6.4 Crystal Characteristics

Table 47-13. Crystal Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
ESR	Equivalent Series Resistor Rs	Fundamental @ 3 MHz Fundamental @ 8 MHz Fundamental @ 16 MHz Fundamental @ 20 MHz			200 100 80 50	Ω
C_M	Motional Capacitance				8	fF
C_S	Shunt Capacitance				7	pF

47.6.5 PLL Characteristics

Table 47-14. Phase Lock Loop Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
F_{OUT}	Output Frequency	Field OUT of CKGR_PLL is 00	80		200	MHz
		Field OUT of CKGR_PLL is 10	190		240	MHz
F_{IN}	Input Frequency		1		32	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.

47.7 USB Transceiver Characteristics

47.7.1 Electrical Characteristics

Table 47-15. Electrical Parameters

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Input Levels						
V_{IL}	Low Level				0.8	V
V_{IH}	High Level		2.0			V
V_{DI}	Differential Input Sensitivity	|(D+) - (D-)|	0.2			V
V_{CM}	Differential Input Common Mode Range		0.8		2.5	V
C_{IN}	Transceiver capacitance	Capacitance to ground on each line			9.18	pF
I	Hi-Z State Data Line Leakage	$0V < V_{IN} < 3.3V$	- 10		+ 10	µA
R_{EXT}	Recommended External USB Series Resistor	In series with each USB pin with ±5%		27		Ω
Output Levels						
V_{OL}	Low Level Output	Measured with R_L of 1.425 kΩ tied to 3.6V	0.0		0.3	V
V_{OH}	High Level Output	Measured with R_L of 14.25 kΩ tied to GND	2.8		3.6	V
V_{CRS}	Output Signal Crossover Voltage	Measure conditions described in Figure 47-14	1.3		2.0	V
Pull-up and Pull-down Resistor						
R_{PUI}	Bus Pull-up Resistor on Upstream Port (idle bus)		0.900		1.575	kOhm
R_{PUA}	Bus Pull-up Resistor on Upstream Port (upstream port receiving)		1.425		3.090	kOhm

47.8 EBI Timings

47.8.1 SMC Timing Conditions

SMC Timings are given in MAX and STH corners.

Timings are given assuming a capacitance load on data, control and address pads:

Table 47-16. Capacitance Load

	Corner		
Supply	MAX	STH	MIN
3.3V	50pF	50pF	0 pf
1.8V	30 pF	30 pF	0 pF

In the tables that follow, MCK period is represented by t_{CPMCK}.

47.8.2 EBI 0 timings

47.8.2.1 Read Timings

Table 47-17. SMC Read Signals - NRD Controlled (READ_MODE = 1)

	Parameter	Min				Units
	VDDIOM supply	1.8V		3.3V		Units
Symbol	VDDCORE supply	1.08V	1.2V	1.08V	1.2V	
NO HOLD SETTINGS (nrd hold = 0)						
SMC_1	Data Setup before NRD High	16	12.6	15.2	11.9	ns
SMC_2	Data Hold after NRD High	-4.1	-4.1	-3.8	-3.8	ns
HOLD SETTINGS (nrd hold …0)						
SMC_3	Data Setup before NRD High	11.9	9.6	11.3	9.1	ns
SMC_4	Data Hold after NRD High	-3.6	-3.6	-3.3	-3.3	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} - 1.6	(nrd setup + nrd pulse)* t_{CPMCK} - 1.3	(nrd setup + nrd pulse)* t_{CPMCK} - 1.9	(nrd setup + nrd pulse)* t_{CPMCK} - 1.7	ns
SMC_6	NCS Low before NRD High	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} - 1.1	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} - 0.8	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} - 1.5	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} - 1.2	ns
SMC_7	NRD Pulse Width	nrd pulse * t_{CPMCK} - 0.3	nrd pulse * t_{CPMCK} - 0.3	nrd pulse * t_{CPMCK} - 0.7	nrd pulse * t_{CPMCK} - 0.6	ns

Table 47-18. SMC Read Signals - NCS Controlled (READ_MODE = 0)

Symbol	Parameter	Min				Units
	VDDIOM supply	1.8V		3.3V		Units
	VDDCORE supply	1.08V	1.2V	1.08V	1.2V	
NO HOLD SETTINGS (ncs rd hold = 0)						
SMC_8	Data Setup before NCS High	16.6	13.0	15.8	12.4	ns
SMC_9	Data Hold after NCS High	-3.9	-3.9	-3.7	-3.7	ns
HOLD SETTINGS (ncs rd hold …0)						
SMC_{10}	Data Setup before NCS High	12.5	10.0	11.9	9.5	ns
SMC_{11}	Data Hold after NCS High	-3.4	-3.4	-3.1	-3.1	ns
HOLD or NO HOLD SETTINGS (ncs rd hold …0, ncs rd hold = 0)						
SMC_{12}	NBS0/A0, NBS1, NBS2/A1, NBS3, A2 - A25 valid before NCS High	(ncs rd setup + ncs rd pulse)* t_{CPMCK} - 1.5	(nrd setup + nrd pulse)* t_{CPMCK} - 1.3	(nrd setup + nrd pulse)* t_{CPMCK} - 1.9	(nrd setup + nrd pulse)* t_{CPMCK} - 1.7	ns
SMC_{13}	NRD low before NCS High	(ncs rd setup + ncs rd pulse - nrd setup)* t_{CPMCK} - 0.5	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} - 0.5	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} - 0.7	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} - 0.6	ns
SMC_{14}	NCS Pulse Width	ncs rd pulse length * t_{CPMCK} - 0.5	nrd pulse * t_{CPMCK} - 0.4	nrd pulse * t_{CPMCK} - 0.9	nrd pulse * t_{CPMCK} - 0.8	ns

47.8.2.2 Write Timings

Table 47-19. SMC Write Signals - NWE Controlled (WRITE_MODE = 1)

Symbol	Parameter	Min				Units
	VDDIOM supply	1.8V		3.3V		Units
	VDDCORE supply	1.08V	1.2V	1.08V	1.2V	
HOLD or NO HOLD SETTINGS (nwe hold …0, nwe hold = 0)						
SMC_{15}	Data Out Valid before NWE High	nwe pulse * t_{CPMCK} - 0.9	nwe pulse * t_{CPMCK} - 0.5	nwe pulse * t_{CPMCK} - 1.2	nwe pulse * t_{CPMCK} - 0.8	ns
SMC_{16}	NWE Pulse Width	nwe pulse * t_{CPMCK} - 0.5	nwe pulse * t_{CPMCK} - 0.4	nwe pulse * t_{CPMCK} - 0.8	nwe pulse * t_{CPMCK} - 0.7	ns
SMC_{17}	NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25 valid before NWE low	nwe setup * t_{CPMCK} - 1.1	nwe setup * t_{CPMCK} - 0.9	nwe setup * t_{CPMCK} - 1.4	nwe setup * t_{CPMCK} - 1.2	ns
SMC_{18}	NCS low before NWE high	(nwe setup - ncs rd setup + nwe pulse) * t_{CPMCK} - 1.3	(nwe setup - ncs rd setup + nwe pulse) * t_{CPMCK} - 1.1	(nwe setup - ncs rd setup + nwe pulse) * t_{CPMCK} - 1.7	(nwe setup - ncs rd setup + nwe pulse) * t_{CPMCK} - 1.5	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} - 3.0	nwe hold * t_{CPMCK} - 2.6	nwe hold * t_{CPMCK} - 3.2	nwe hold * t_{CPMCK} - 2.8	ns

Table 47-19. SMC Write Signals - NWE Controlled (WRITE_MODE = 1) (Continued)

Symbol	Parameter	Min				Units
	VDDIOM supply	1.8V		3.3V		Units
	VDDCORE supply	1.08V	1.2V	1.08V	1.2V	
SMC_{20}	NWE High to NCS Inactive (1)	(nwe hold - ncs wr hold)* t_{CPMCK} - 0.6	(nwe hold - ncs wr hold)* t_{CPMCK} - 0.5	(nwe hold - ncs wr hold)* t_{CPMCK} - 0.6	(nwe hold - ncs wr hold)* t_{CPMCK} - 0.5	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)	3.5	3.5	3.3	3.3	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 47-20. SMC Write NCS Controlled (WRITE_MODE = 0)

Symbol	Parameter	Min				Units
	VDDIOM supply	1.8V		3.3V		Units
	VDDCORE supply	1.08V	1.2V	1.08V	1.2V	
SMC_{22}	Data Out Valid before NCS High	ncs wr pulse * t_{CPMCK} - 0.6	ncs wr pulse * t_{CPMCK} - 0.3	ncs wr pulse * t_{CPMCK} - 0.9	ncs wr pulse * t_{CPMCK} - 0.5	ns
SMC_{23}	NCS Pulse Width	ncs wr pulse * t_{CPMCK} - 0.5	ncs wr pulse * t_{CPMCK} - 0.4	ncs wr pulse * t_{CPMCK} - 0.9	ncs wr pulse * t_{CPMCK} - 0.8	ns
SMC_{24}	NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25 valid before NCS low	ncs wr setup * t_{CPMCK} - 0.8	ncs wr setup * t_{CPMCK} - 0.6	ncs wr setup * t_{CPMCK} - 1.1	ncs wr setup * t_{CPMCK} - 0.9	ns
SMC_{25}	NWE low before NCS high	(ncs wr setup - nwe setup + ncs pulse)* t_{CPMCK} - 0.7	(ncs wr setup - nwe setup + ncs pulse)* t_{CPMCK} - 0.6	(ncs wr setup - nwe setup + ncs pulse)* t_{CPMCK} - 1.1	(ncs wr setup - nwe setup + ncs pulse)* t_{CPMCK} - 0.9	ns
SMC_{26}	NCS High to Data Out, NBS0/A0, NBS1, NBS2/A1, NBS3, A2 - A25, change	ncs wr hold * t_{CPMCK} - 3.2	ncs wr hold * t_{CPMCK} - 2.8	ncs wr hold * t_{CPMCK} - 3.4	ncs wr hold * t_{CPMCK} - 2.9	ns
SMC_{27}	NCS High to NWE Inactive	(ncs wr hold - nwe hold)* t_{CPMCK} - 1.1	(ncs wr hold - nwe hold)* t_{CPMCK} - 0.9	(ncs wr hold - nwe hold)* t_{CPMCK} - 1.1	(ncs wr hold - nwe hold)* t_{CPMCK} - 0.9	ns

47.8.3 EBI 1 timings

47.8.3.1 Read Timings

Table 47-21. SMC Read Signals - NRD Controlled (READ_MODE = 1)

Symbol	Parameter	Min				Units
	VDDIOM supply	1.8V		3.3V		Units
	VDDCORE supply	1.08V	1.2V	1.08V	1.2V	
NO HOLD SETTINGS (nrd hold = 0)						
SMC_1	Data Setup before NRD High	15.2	12.0	14.6	11.5	ns
SMC_2	Data Hold after NRD High	-3.2	-3.2	-3.0	-3.0	ns
HOLD SETTINGS (nrd hold …0)						
SMC_3	Data Setup before NRD High	11.6	9.4	11.0	8.9	ns
SMC_4	Data Hold after NRD High	-2.9	-2.9	-2.6	-2.6	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} - 2.7	(nrd setup + nrd pulse)* t_{CPMCK} - 2.0	(nrd setup + nrd pulse)* t_{CPMCK} - 3.1	(nrd setup + nrd pulse)* t_{CPMCK} - 2.4	ns
SMC_6	NCS low before NRD High	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} - 1.6	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} - 1.2	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} - 2.0	(nrd setup + nrd pulse - ncs rd setup) * t_{CPMCK} - 1.5	ns
SMC_7	NRD Pulse Width	nrd pulse * t_{CPMCK}	nrd pulse * t_{CPMCK}	nrd pulse * t_{CPMCK} - 0.1	nrd pulse * t_{CPMCK} - 0.1	ns

Table 47-22. SMC Read Signals - NCS Controlled (READ_MODE = 0)

Symbol	Parameter	Min				Units
	VDDIOM supply	1.8V		3.3V		Units
	VDDCORE supply	1.08V	1.2V	1.08V	1.2V	
NO HOLD SETTINGS (ncs rd hold = 0)						
SMC_8	Data Setup before NCS High	16.7	13.1	16.1	12.5	ns
SMC_9	Data Hold after NCS High	-3.2	-3.2	-2.9	-2.9	ns
HOLD SETTINGS (ncs rd hold …0)						
SMC_{10}	Data Setup before NCS High	13.1	10.5	12.5	9.9	ns
SMC_{11}	Data Hold after NCS High	-2.8	-2.8	-2.5	-2.5	ns

Table 47-22. SMC Read Signals - NCS Controlled (READ_MODE = 0) (Continued)

HOLD or NO HOLD SETTINGS (ncs rd hold …0, ncs rd hold = 0)						
SMC_{12}	NBS0/A0, NBS1, NBS2/A1, NBS3, A2 - A25 valid before NCS High	(ncs rd setup + ncs rd pulse)* t_{CPMCK} - 3.0	(ncs rd setup + ncs rd pulse)* t_{CPMCK} - 2.2	(ncs rd setup + ncs rd pulse)* t_{CPMCK} - 3.4	(ncs rd setup + ncs rd pulse)* t_{CPMCK} - 2.6	ns
SMC_{13}	NRD low before NCS High	(ncs rd setup + ncs rd pulse - nrd setup)* t_{CPMCK} - 0.1	(ncs rd setup + ncs rd pulse - nrd setup)* t_{CPMCK} - 0.1	(ncs rd setup + ncs rd pulse - nrd setup)* t_{CPMCK} - 0.2	(ncs rd setup + ncs rd pulse - nrd setup)* t_{CPMCK} - 0.2	ns
SMC_{14}	NCS Pulse Width	ncs rd pulse length * t_{CPMCK} - 0.2	ncs rd pulse length * t_{CPMCK} - 0.2	ncs rd pulse length * t_{CPMCK} - 0.5	ncs rd pulse length * t_{CPMCK} - 0.4	ns

47.8.3.2 Write Timings

Table 47-23. SM Write Signals - NWE controlled (WRITE_MODE = 1)

	Parameter	**Min**				**Units**
	VDDIOM supply	**1.8V**		**3.3V**		**Units**
Symbol	**VDDCORE supply**	**1.08V**	**1.2V**	**1.08V**	**1.2V**	
HOLD or NO HOLD SETTINGS (nwe hold …0, nwe hold = 0)						
SMC_{15}	Data Out Valid before NWE High	nwe pulse * t_{CPMCK} - 2.7	nwe pulse * t_{CPMCK} - 1.8	nwe pulse * t_{CPMCK} - 2.9	nwe pulse * t_{CPMCK} - 2.0	ns
SMC_{16}	NWE Pulse Width	nwe pulse * t_{CPMCK} - 0.3	nwe pulse * t_{CPMCK} - 0.3	nwe pulse * t_{CPMCK} - 0.7	nwe pulse * t_{CPMCK} - 0.6	ns
SMC_{17}	NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25 valid before NWE low	nwe setup * t_{CPMCK} - 1.2	nwe setup * t_{CPMCK} - 0.9	nwe setup * t_{CPMCK} - 1.6	nwe setup * t_{CPMCK} - 1.2	ns
SMC_{18}	NCS low before NWE high	(nwe setup - ncs rd setup + nwe pulse) * t_{CPMCK} - 2.0	(nwe setup - ncs rd setup + nwe pulse) * t_{CPMCK} - 1.4	(nwe setup - ncs rd setup + nwe pulse) * t_{CPMCK} - 2.4	(nwe setup - ncs rd setup + nwe pulse) * t_{CPMCK} - 1.7	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} - 2.2	nwe hold * t_{CPMCK} - 2.0	nwe hold * t_{CPMCK} - 2.4	nwe hold * t_{CPMCK} - 2.1	ns
SMC_{20}	NWE High to NCS Inactive (1)	(nwe hold - ncs wr hold)* t_{CPMCK} - 1.0	(nwe hold - ncs wr hold)* t_{CPMCK} - 0.7	(nwe hold - ncs wr hold)* t_{CPMCK} - 1.0	(nwe hold - ncs wr hold)* t_{CPMCK} - 0.7	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)	2.9	2.9	2.7	2.7	ns

Notes: 1. Hold length = total cycle duration - setup duration - pulse duration. “hold length” is used for “ncs wr hold length” or “NWE hold length”.

Table 47-24. SMC Write NCS Controlled (WRITE_MODE = 0)

	Parameter	Min				Units
	VDDIOM supply	1.8V		3.3V		Units
Symbol	VDDCORE supply	1.08V	1.2V	1.08V	1.2V	
SMC_{22}	Data Out Valid before NCS High	ncs wr pulse * t_{CPMCK} - 2.6	ncs wr pulse * t_{CPMCK} - 1.7	ncs wr pulse * t_{CPMCK} - 2.8	ncs wr pulse * t_{CPMCK} - 2.0	ns
SMC_{23}	NCS Pulse Width	ncs wr pulse * t_{CPMCK} - 0.2	ncs wr pulse * t_{CPMCK} - 0.2	ncs wr pulse * t_{CPMCK} - 0.5	ncs wr pulse * t_{CPMCK} - 0.4	ns
SMC_{24}	NBS0/A0 NBS1, NBS2/A1, NBS3, A2 - A25 valid before NCS low	ncs wr setup * t_{CPMCK} - 1.1	ncs wr setup * t_{CPMCK} - 0.8	ncs wr setup * t_{CPMCK} - 1.5	ncs wr setup * t_{CPMCK} - 1.2	ns
SMC_{25}	NWE low before NCS high	(ncs wr setup - nwe setup + ncs pulse)* t_{CPMCK} - 1.1	(ncs wr setup - nwe setup + ncs pulse)* t_{CPMCK} - 0.8	(ncs wr setup - nwe setup + ncs pulse)* t_{CPMCK} - 1.5	(ncs wr setup - nwe setup + ncs pulse)* t_{CPMCK} - 1.2	ns
SMC_{26}	NCS High to Data Out, NBS0/A0, NBS1, NBS2/A1, NBS3, A2 - A25, change	ncs wr hold * t_{CPMCK} - 3.2	ncs wr hold * t_{CPMCK} - 2.7	ncs wr hold * t_{CPMCK} - 3.4	ncs wr hold * t_{CPMCK} - 2.8	ns
SMC_{27}	NCS High to NWE Inactive	(ncs wr hold - nwe hold)* t_{CPMCK} - 3.0	(ncs wr hold - nwe hold)* t_{CPMCK} - 2.3	(ncs wr hold - nwe hold)* t_{CPMCK} - 3.2	(ncs wr hold - nwe hold)* t_{CPMCK} - 2.5	ns

Figure 47-3. SMC timings - NCS controlled Read and Write

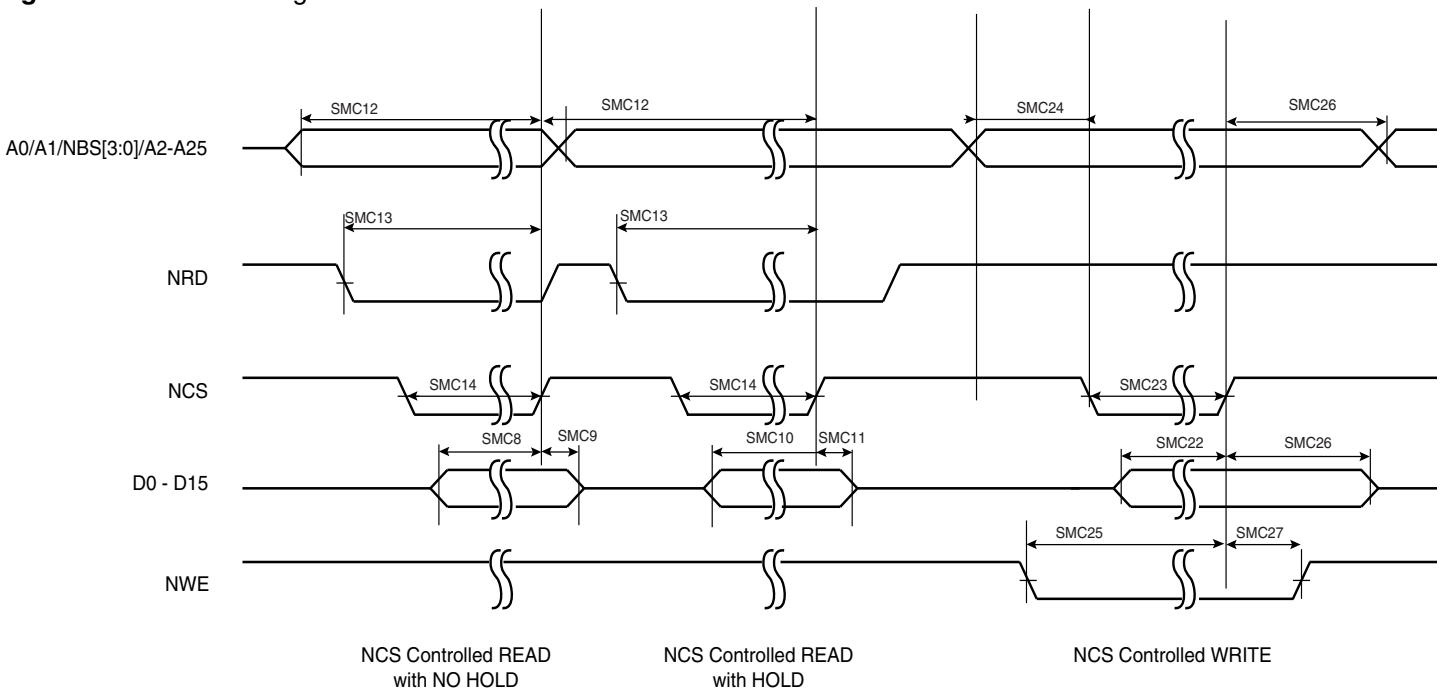
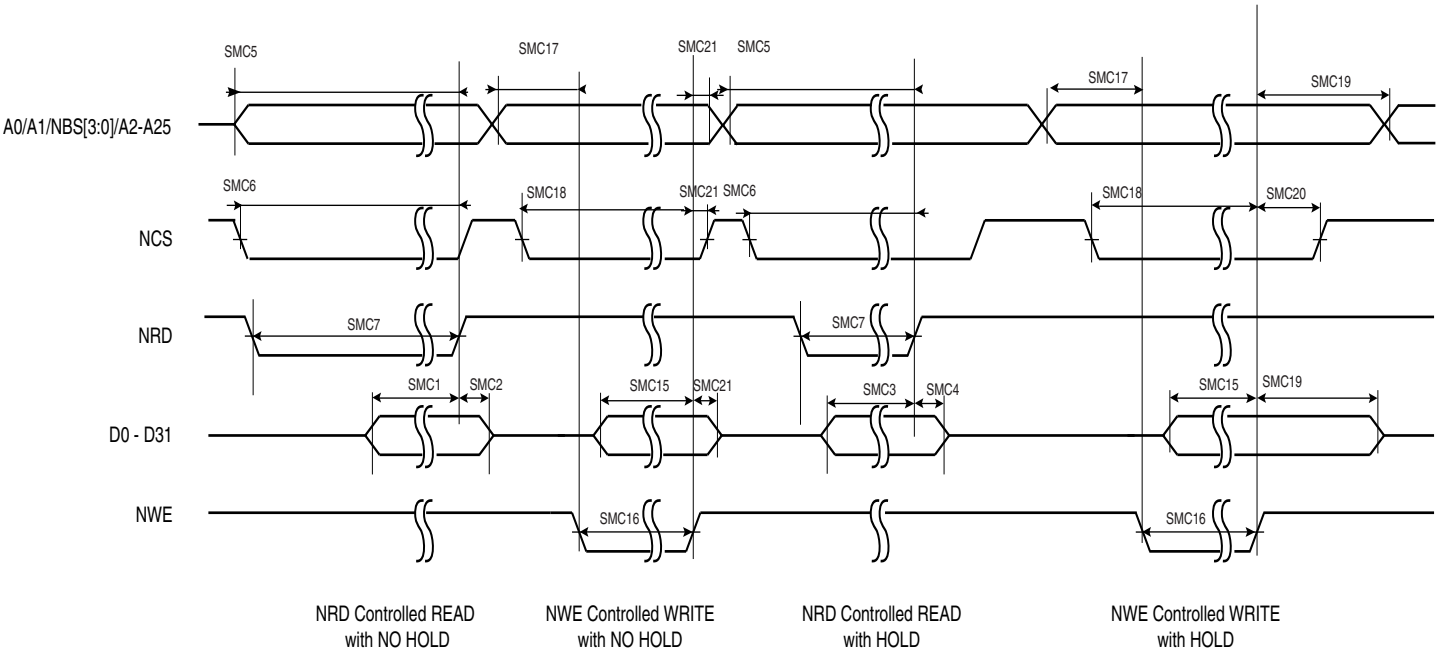


Figure 47-4. SMC timings - NRD controlled Read and NWE controlled Write



47.8.4 SDRAMC Signals

Timings are given assuming a capacitance load on data, control and address pads :

Table 47-25. Capacitance Load on data, control and address pads

	Corner
IO Supply	**MAX**
3.3V	50pF
1.8V	30 pF

Table 47-26. Capacitance Load on SDCK pad

	Corner
IO Supply	**MAX**
3.3V	10pF
1.8V	10pF

In each column, two timings are given:

- First ones are given for $V_{VDDCORE}$ = 1.08V
- Second ones are given for $V_{VDDCORE}$ = 1.2V

47.8.4.1 External Bus Interface 0 Timings

Table 47-27. SDRAMC Clock Signal

	Parameter	**Max**				
	VDDIOM supply	**1.8V**		**3.3V**		
Symbol	**VDDCORE supply**	**1.08V**	**1.2V**	**1.08V**	**1.2V**	**Units**
$1/(t_{CPSDCK})$	SDRAM Controller Clock Frequency	100	120	100	120	MHz

Table 47-28. SDRAMC Signals

	Parameter	**Min**				
	VDDIOM supply	**1.8V**		**3.3V**		
Symbol	**VDDCORE supply**	**1.08V**	**1.2V**	**1.08V**	**1.2V**	**Units**
$SDRAMC_1$	SDCKE High before SDCK Rising Edge	4.4	4.6	3.1	3.5	ns
$SDRAMC_2$	SDCKE Low after SDCK Rising Edge	4.6	4.6	5.2	5.2	ns
$SDRAMC_3$	SDCKE Low before SDCK Rising Edge	4.4	4.6	2.7	3.1	ns
$SDRAMC_4$	SDCKE High after SDCK Rising Edge	4.6	4.6	5.0	5.0	ns
$SDRAMC_5$	SDCS Low before SDCK Rising Edge	4.4	4.5	2.7	3.1	ns
$SDRAMC_6$	SDCS High after SDCK Rising Edge	4.7	4.7	5.1	5.1	ns
$SDRAMC_7$	RAS Low before SDCK Rising Edge	4.0	4.3	2.3	2.9	ns
$SDRAMC_8$	RAS High after SDCK Rising Edge	4.7	4.7	5.1	5.1	ns
$SDRAMC_9$	SDA10 Change before SDCK Rising Edge	3.8	4.2	2.5	3.1	ns

Table 47-28. SDRAMC Signals

Symbol	Parameter	Min				Units
	VDDIOM supply	1.8V		3.3V		
	VDDCORE supply	1.08V	1.2V	1.08V	1.2V	
$SDRAMC_{10}$	SDA10 Change after SDCK Rising Edge	4.6	4.6	5.1	5.1	ns
$SDRAMC_{11}$	Address Change before SDCK Rising Edge	2.9	3.3	2.2	1.8	ns
$SDRAMC_{12}$	Address Change after SDCK Rising Edge	4.3	4.3	4.7	4.7	ns
$SDRAMC_{13}$	Bank Change before SDCK Rising Edge	2.8	3.2	1.1	1.7	ns
$SDRAMC_{14}$	Bank Change after SDCK Rising Edge	4.6	4.6	5.0	5.0	ns
$SDRAMC_{15}$	CAS Low before SDCK Rising Edge	4.7	4.8	3.0	3.4	ns
$SDRAMC_{16}$	CAS High after SDCK Rising Edge	4.5	4.5	4.9	4.9	ns
$SDRAMC_{17}$	DQM Change before SDCK Rising Edge	2.8	3.2	1.1	1.7	ns
$SDRAMC_{18}$	DQM Change after SDCK Rising Edge	4.2	4.2	4.6	4.6	ns
$SDRAMC_{19}$	D0-D15 in Setup before SDCK Rising Edge	2.5	1.7	2.6	1.8	ns
$SDRAMC_{20}$	D0-D15 in Hold after SDCK Rising Edge	-0.1	-0.1	-0.1	-0.1	ns
$SDRAMC_{21}$	D16-D31 in Setup before SDCK Rising Edge	3.3	2.3	3.4	2.4	ns
$SDRAMC_{22}$	D16-D31 in Hold after SDCK Rising Edge	0.1	0.1	0.2	0.2	ns
$SDRAMC_{23}$	SDWE Low before SDCK Rising Edge	4.1	4.4	2.4	2.9	ns
$SDRAMC_{24}$	SDWE High after SDCK Rising Edge	4.8	4.8	5.2	5.2	ns
$SDRAMC_{25}$	D0-D15 Out Valid before SDCK Rising Edge	2.9	3.4	1.4	2.0	ns
$SDRAMC_{26}$	D0-D15 Out Valid after SDCK Rising Edge	3.8	3.8	4.5	4.5	ns
$SDRAMC_{27}$	D16-D31 Out Valid before SDCK Rising Edge	4.1	4.5	2.4	3.0	ns
$SDRAMC_{28}$	D16-D31 Out Valid after SDCK Rising Edge	3.4	3.4	4.4	4.4	ns

Note: 1. The derating factor is not to be applied to t_{CLMCK} or t_{CHMCK}.

47.8.4.2 External Bus Interface 1 Timings

Table 47-29. SDRAMC Clock Signal

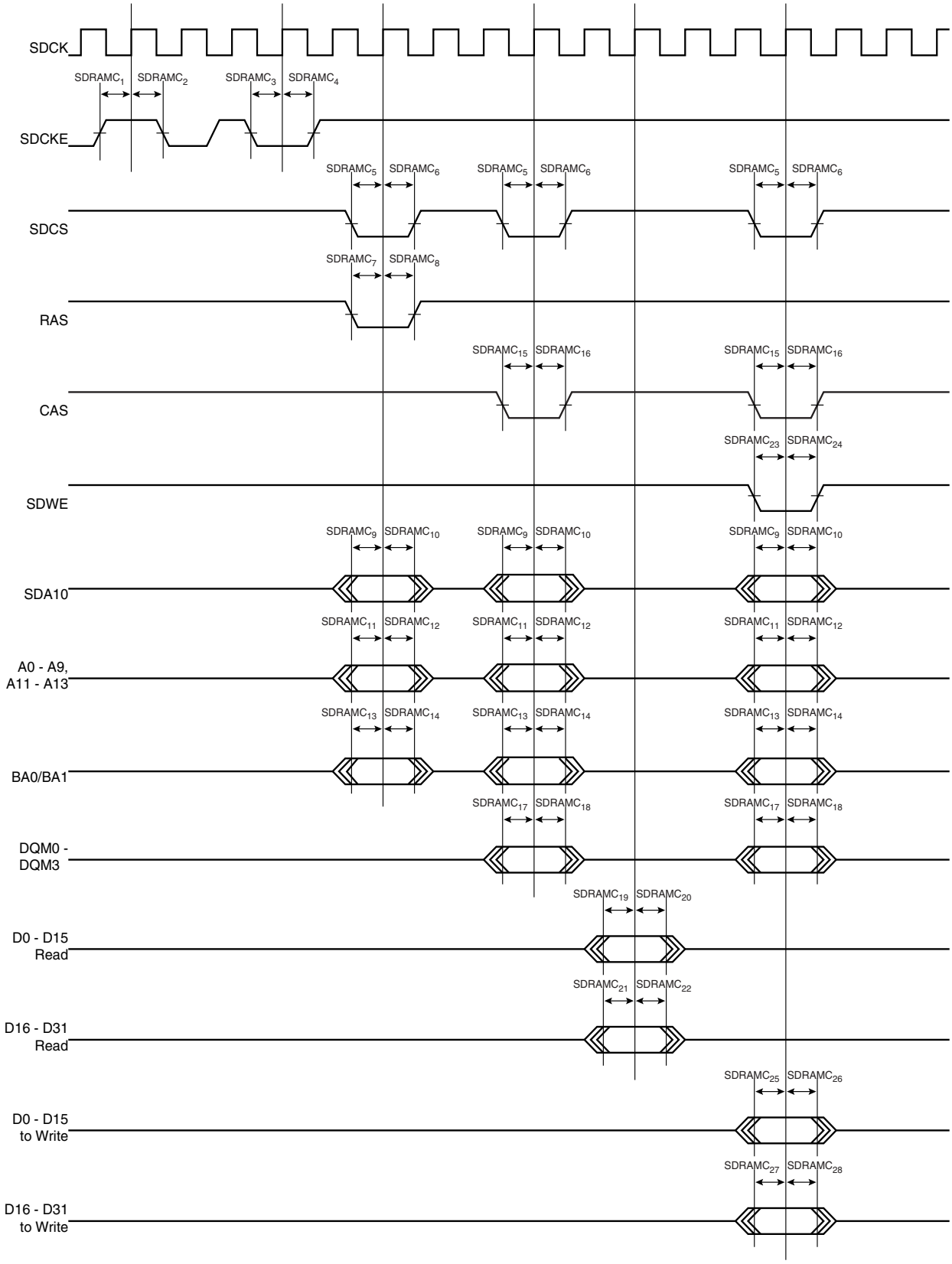
Symbol	Parameter	Max		Units
		1.8V Supply	3.3V Supply	
$1/(t_{CPSDCK})$	SDRAM Controller Clock Frequency	100	100	MHz

Table 47-30. SDRAMC Signals

Symbol	Parameter	Min				Units
	VDDIOM supply	1.8V		3.3V		
	VDDCORE supply	1.08V	1.2V	1.08V	1.2V	
$SDRAMC_1$	SDCKE High before SDCK Rising Edge	3.6	3.9	2.3	2.8	ns
$SDRAMC_2$	SDCKE Low after SDCK Rising Edge	4.3	4.4	5.1	5.1	ns
$SDRAMC_3$	SDCKE Low before SDCK Rising Edge	3.8	4.1	2.2	2.6	ns
$SDRAMC_4$	SDCKE High after SDCK Rising Edge	3.7	3.9	4.8	4.8	ns
$SDRAMC_5$	SDCS Low before SDCK Rising Edge	3.7	4.0	2.0	2.5	ns
$SDRAMC_6$	SDCS High after SDCK Rising Edge	4.2	4.2	5.0	5.0	ns
$SDRAMC_7$	RAS Low before SDCK Rising Edge	4.7	4.8	3.1	3.4	ns
$SDRAMC_8$	RAS High after SDCK Rising Edge	3.0	3.4	4.1	4.3	ns
$SDRAMC_9$	SDA10 Change before SDCK Rising Edge	4.0	4.3	2.7	3.1	ns
$SDRAMC_{10}$	SDA10 Change after SDCK Rising Edge	3.7	3.8	4.8	4.7	ns
$SDRAMC_{11}$	Address Change before SDCK Rising Edge	3.2	3.6	1.7	2.4	ns
$SDRAMC_{12}$	Address Change after SDCK Rising Edge	3.8	3.9	4.3	4.3	ns
$SDRAMC_{13}$	Bank Change before SDCK Rising Edge	3.4	3.8	1.8	2.4	ns
$SDRAMC_{14}$	Bank Change after SDCK Rising Edge	3.8	3.8	4.3	4.3	ns
$SDRAMC_{15}$	CAS Low before SDCK Rising Edge	4.7	4.8	3.1	3.4	ns
$SDRAMC_{16}$	CAS High after SDCK Rising Edge	3.0	3.4	4.1	4.3	ns
$SDRAMC_{17}$	DQM Change before SDCK Rising Edge	3.3	3.6	1.7	2.2	ns
$SDRAMC_{18}$	DQM Change after SDCK Rising Edge	3.8	3.8	4.3	4.3	ns
$SDRAMC_{19}$	D0-D15 in Setup before SDCK Rising Edge	0.7	0.5	0.8	0.5	ns
$SDRAMC_{20}$	D0-D15 in Hold after SDCK Rising Edge	0.6	0.5	0.7	0.6	ns
$SDRAMC_{21}$	D16-D31 in Setup before SDCK Rising Edge	2.1	1.5	2.2	1.6	ns
$SDRAMC_{22}$	D16-D31 in Hold after SDCK Rising Edge	-0.2	-0.2	-0.1	-0.1	ns
$SDRAMC_{23}$	SDWE Low before SDCK Rising Edge	4.2	3.4	2.5	2.9	ns
$SDRAMC_{24}$	SDWE High after SDCK Rising Edge	3.6	3.7	4.7	4.7	ns
$SDRAMC_{25}$	D0-D15 Out Valid before SDCK Rising Edge	3.2	3.7	1.7	2.3	ns
$SDRAMC_{26}$	D0-D15 Out Valid after SDCK Rising Edge	3.1	3.4	4.1	4.1	ns
$SDRAMC_{27}$	D16-D31 Out Valid before SDCK Rising Edge	3.9	4.2	2.4	2.8	ns
$SDRAMC_{28}$	D16-D31 Out Valid after SDCK Rising Edge	3.4	3.6	4.5	4.5	ns

Note: 1. The derating factor is not to be applied to t_{CLMCK} or t_{CHMCK}.

Figure 47-5. SDRAMC Signals Relative to SDCK



47.9 EMAC Timings

Table 47-31. EMAC Signals Relative to EMDC

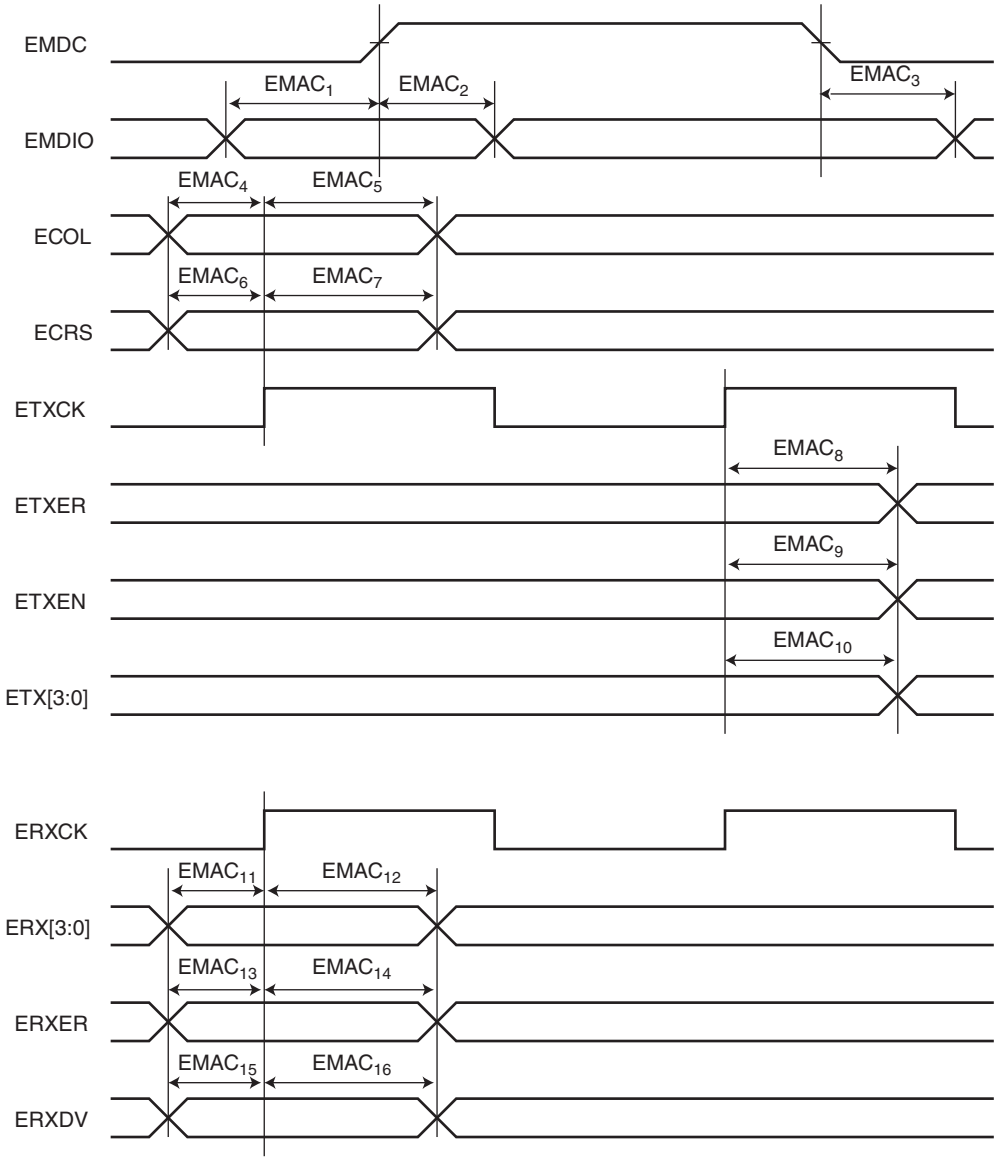
Symbol	Parameter	Min (ns)	Max (ns)
$EMAC_1$	Setup for EMDIO from EMDC rising	15.5	
$EMAC_2$	Hold for EMDIO from EMDC rising	-4.4	
$EMAC_3$	EMDIO toggling from EMDC rising	-3.9	3.1

47.9.1 MII Mode

Table 47-32. EMAC MII Specific Signals

Symbol	Parameter	Min (ns)	Max (ns)
$EMAC_4$	Setup for ECOL from ETXCK rising	-0.4	
$EMAC_5$	Hold for ECOL from ETXCK rising	2.0	
$EMAC_6$	Setup for ECRS from ETXCK rising	1.3	
$EMAC_7$	Hold for ECRS from ETXCK rising	0.1	
$EMAC_8$	ETXER toggling from ETXCK rising	4.4	11.9
$EMAC_9$	ETXEN toggling from ETXCK rising	7.0	19.2
$EMAC_{10}$	ETX toggling from ETXCK rising	4.4	12.8
$EMAC_{11}$	Setup for ERX from ERXCK	9.1	
$EMAC_{12}$	Hold for ERX from ERXCK	-1.8	
$EMAC_{13}$	Setup for ERXER from ERXCK	1.5	
$EMAC_{14}$	Hold for ERXER from ERXCK	-0.4	
$EMAC_{15}$	Setup for ERXDV from ERXCK	5.1	
$EMAC_{16}$	Hold for ERXDV from ERXCK	-1.7	

Figure 47-6. EMAC MII Mode

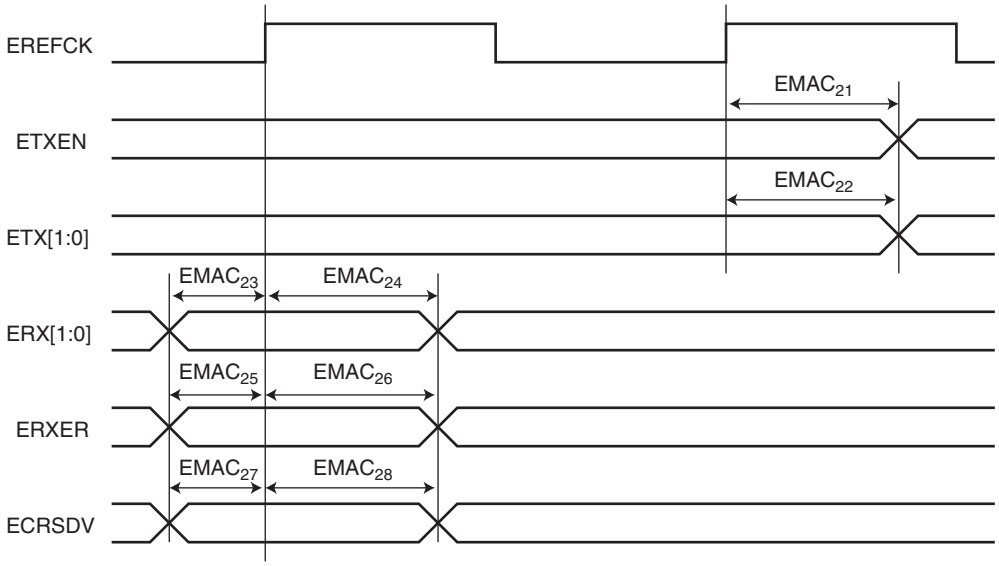


47.9.2 RMII Mode

Table 47-33. EMAC RMII Specific Signals

Symbol	Parameter	Min (ns)	Max (ns)
$EMAC_{21}$	ETXEN toggling from EREFCK rising	5.0	14.6
$EMAC_{22}$	ETX toggling from EREFCK rising	4.8	12.9
$EMAC_{23}$	Setup for ERX from EREFCK	3.9	
$EMAC_{24}$	Hold for ERX from EREFCK	-0.1	
$EMAC_{25}$	Setup for ERXER from EREFCK	-0.2	
$EMAC_{26}$	Hold for ERXER from EREFCK	1.0	
$EMAC_{27}$	Setup for ECRSDV from EREFCK	3.7	
$EMAC_{28}$	Hold for ECRSDV from EREFCK	-0.1	

Figure 47-7. EMAC RMII Mode



47.10 Peripheral Timings

47.10.1 SPI

Figure 47-8. SPI Master Mode with (CPOL = NCPHA = 1) or (CPOL= 1 and NCPHA = 0)

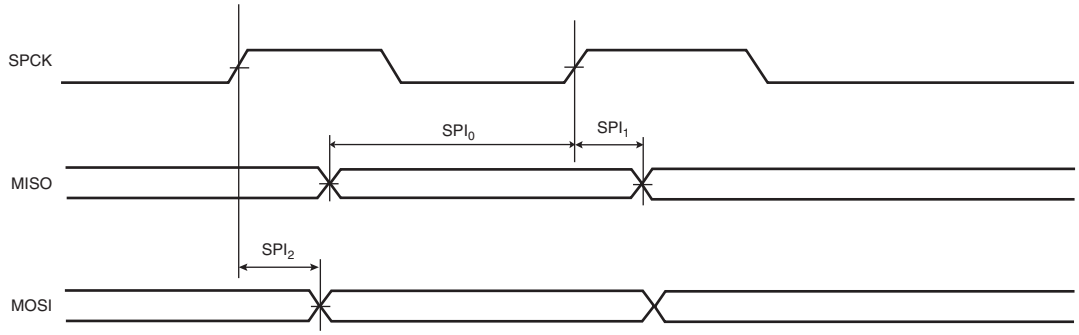


Figure 47-9. SPI Master Mode with (CPOL = NCPHA = 0) or (CPOL = NCPHA = 1)

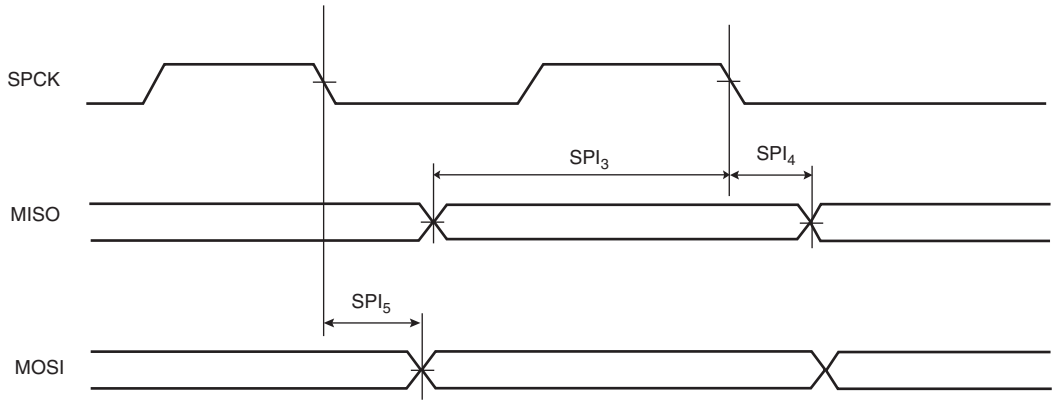


Figure 47-10. SPI Slave Mode with (CPOL=0 and NCPHA=1) or (CPOL=1 and NCPHA=0)

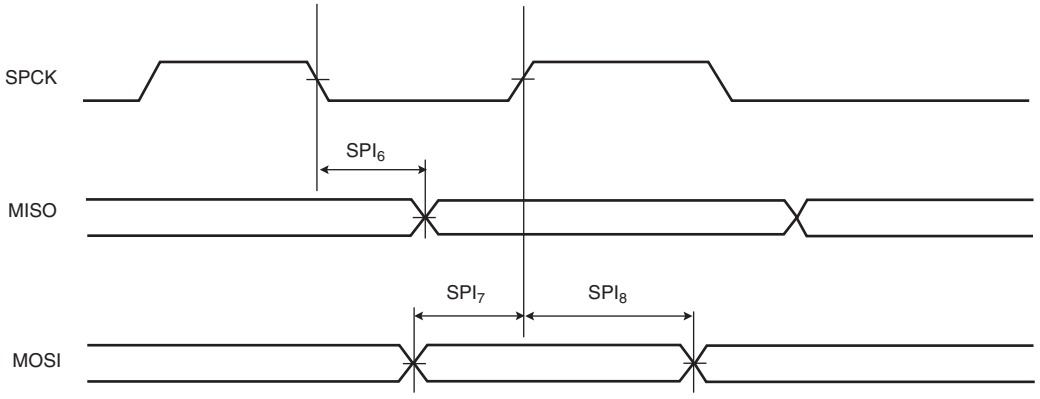


Figure 47-11. SPI Slave Mode with (CPOL = NCPHA = 0) or (CPOL= NCPHA= 1)

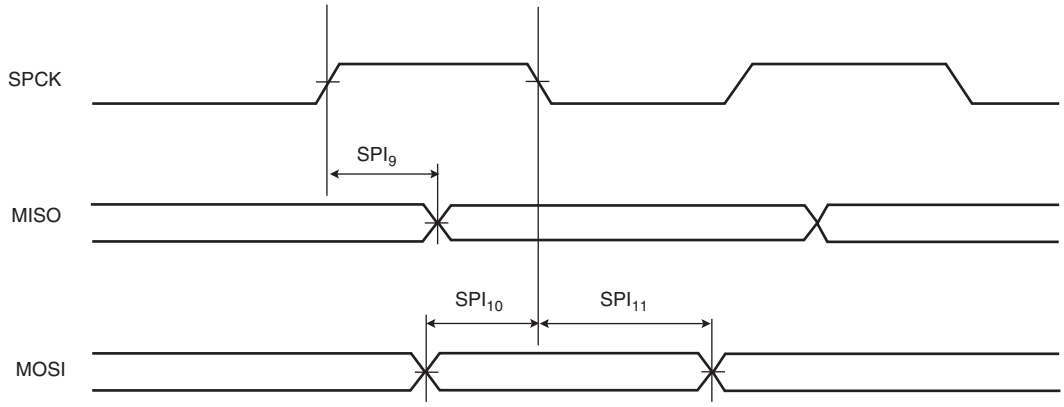


Table 47-34. SPI Timings

Symbol	Parameter	Conditions	Min	Max	Units
SPI_0	MISO Setup time before SPCK rises (master)	(1)	$t_{CPMCK}/2$ +10.3		ns
SPI_1	MISO Hold time after SPCK rises (master)	(1)	$-t_{CPMCK}/2$ -3.5		ns
SPI_2	SPCK rising to MOSI Delay (master)	(1)		1.0	ns
SPI_3	MISO Setup time before SPCK falls (master)	(1)	$t_{CPMCK}/2$ +10.9		ns
SPI_4	MISO Hold time after SPCK falls (master)	(1)	$-t_{CPMCK}/2$ -4.3		ns
SPI_5	SPCK falling to MOSI Delay (master)	(1)		0.4	ns
SPI_6	SPCK falling to MISO Delay (slave)	(1)		9.6	ns
SPI_7	MOSI Setup time before SPCK rises (slave)	(1)	3.5		ns
SPI_8	MOSI Hold time after SPCK rises (slave)	(1)	-0.6		ns
SPI_9	SPCK rising to MISO Delay (slave)	(1)		9.6	ns
SPI_{10}	MOSI Setup time before SPCK falls (slave)	(1)	3.4		ns
SPI_{11}	MOSI Hold time after SPCK falls (slave)	(1)	-0.5		ns

Note: 1. Cload is 8 pF for MISO and 6 pF for SPCK and MOSI.

47.10.2 ISI

Figure 47-12. ISI Timing Diagram

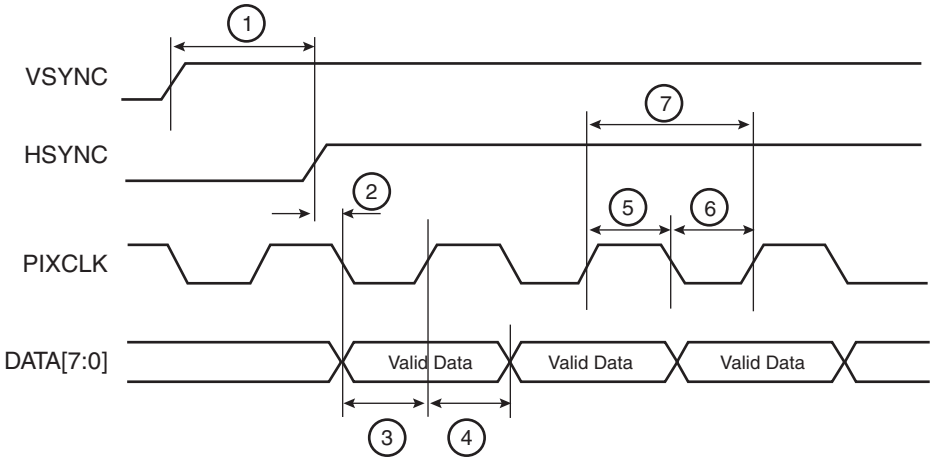


Table 47-35. ISI Timings

Symbol	Parameter	Min	Max	Units
1	VSYNC to HSYNC	2.75		ns
2	HSYNC to PIXCLK	-1.49		ns
3	DATA setup time	5.87		ns
4	DATA hold time	-1.28		ns
5	PIXCLK high time	0		ns
6	PIXCLK low time	0		ns
7	PIXCLK frequency		I/O Max freq See Section 47.5.4 “I/Os” on page 1031	MHz

47.10.3 MCI

The PDC interface block controls all data routing between the external data bus, internal MMC/SD module data bus, and internal system FIFO access through a dedicated state machine that monitors the status of FIFO content (empty or full), FIFO address, and byte/block counters for the MMC/SD module (inner system) and the application (user programming).

These timings are given for a 25 pF load, corresponding to 1 MMC/SD Card.

Figure 47-13. MCI Timing Diagram

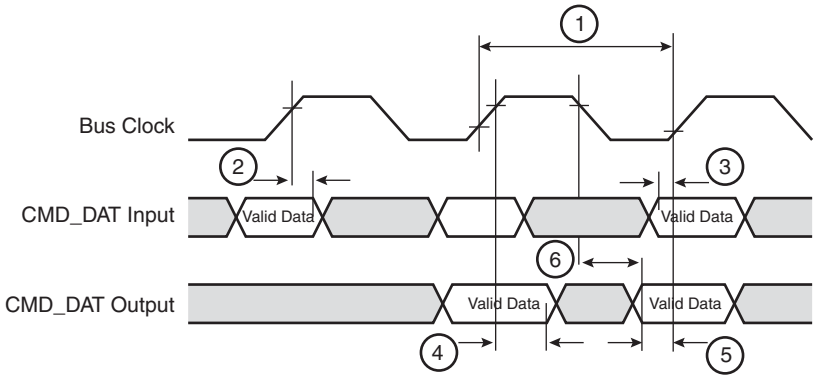


Table 47-36. MCI Timings

Symbol	Parameter	Min	Max	Units
1	$1/t_{CLK}$ = CLK frequency at Data transfer Mode (PP)	0	51	MHz
2	Input hold time	1.5		ns
3	Input setup time	-0.1		ns
4	Output hold time	-1.0		ns
5	Output setup time	$1/t_{CLK}$ - 2.8		ns

47.10.4 UDP and UHP Switching Characteristics

Figure 47-14. USB Data Signal Rise and Fall TImes

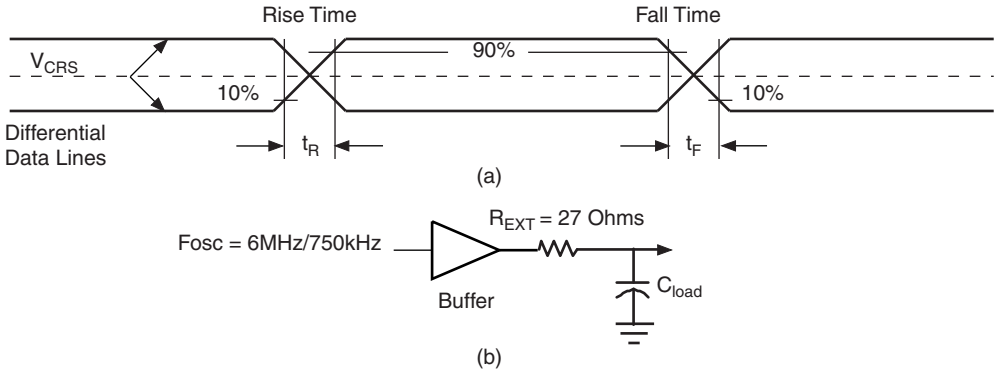


Table 47-37. In Low Speed

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{FR}	Transition Rise Time	C_{LOAD} = 400 pF	75		300	ns
t_{FE}	Transition Fall Time	C_{LOAD} = 400 pF	75		300	ns
t_{FRFM}	Rise/Fall time Matching	C_{LOAD} = 400 pF	80		125	%

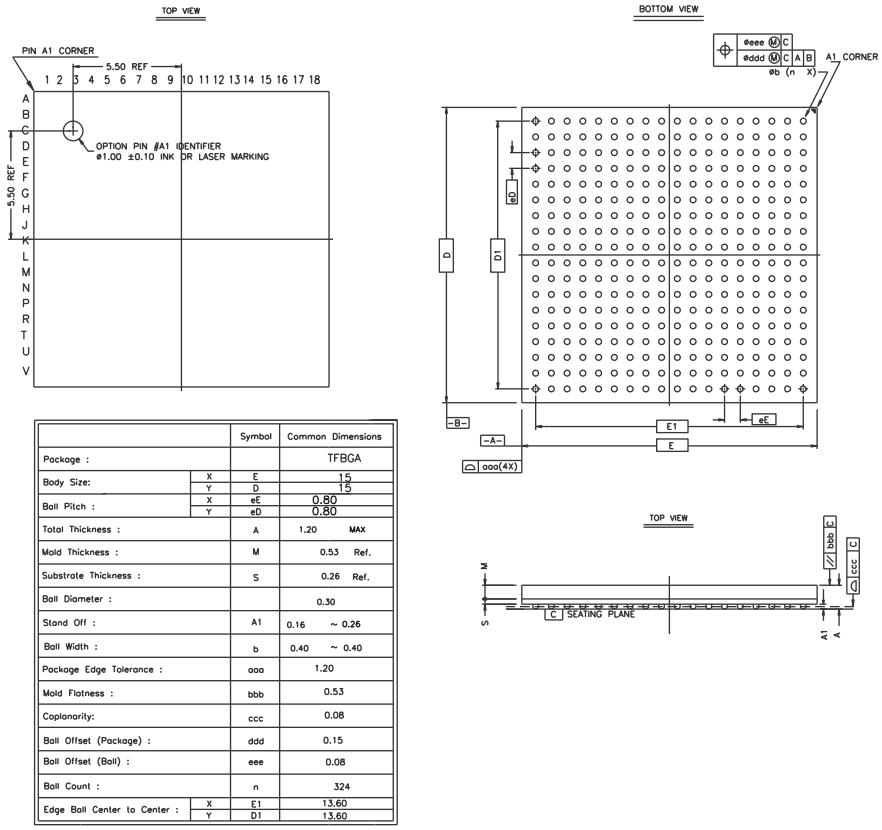
Table 47-38. In Full Speed

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
t_{FR}	Transition Rise Time	C_{LOAD} = 50 pF	4		20	ns
t_{FE}	Transition Fall Time	C_{LOAD} = 50 pF	4		20	ns
t_{FRFM}	Rise/Fall time Matching		90		111.11	%

48. AT91SAM9263 Mechanical Characteristics

48.1 Package Drawing

Figure 48-1. 324-ball TFBGA Package Drawing



		Symbol	Common Dimensions
Package :			TFBGA
Body Size:	X	E	15
	Y	D	15
Ball Pitch :	X	eE	0.80
	Y	eD	0.80
Total Thickness :		A	1.20 MAX
Mold Thickness :		M	0.53 Ref.
Substrate Thickness :		S	0.26 Ref.
Ball Diameter :			0.30
Stand Off :		A1	0.16 ~ 0.26
Ball Width :		b	0.40 ~ 0.40
Package Edge Tolerance :		aaa	1.20
Mold Flatness :		bbb	0.53
Coplanarity:		ccc	0.08
Ball Offset (Package) :		ddd	0.15
Ball Offset (Ball) :		eee	0.08
Ball Count :		n	324
Edge Ball Center to Center :	X	E1	13.60
	Y	D1	13.60

Table 48-1. Soldering Information

Ball Land	0.4 mm +/- 0.05
Soldering Mask Opening	0.275 mm +/- 0.03

Table 48-2. Device and 324-ball TFBGA Package Maximum Weight

572	mg

Table 48-3. 324-ball TFBGA Package Characteristics

Moisture Sensitivity Level	3

Table 48-4. Package Reference

JEDEC Drawing Reference	MO-210
JESD97 Classification	e1

This package respects the recommendations of the NEMI User Group.

48.2 Soldering Profile

Table 48-5 gives the recommended soldering profile from J-STD-020C.

Table 48-5. 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.

49. AT91SAM9263 Ordering Information

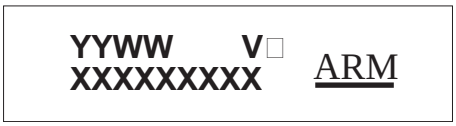
Table 49-1. AT91SAM9263 Ordering Information

Marketing Revision Level A Ordering Code	Marketing Revision Level B Ordering Code	Package	Package Type	Temperature Operating Range
AT91SAM9263-CU	AT91SAM9263B-CU	TFBGA 324	Green	Industrial -40°C to 85°C

50. AT91SAM9263 Errata

50.1 Marking

All devices are marked with the Atmel logo and the ordering code.
Additional marking has the following format:



where

- “YY”: manufactory year
- “WW”: manufactory week
- “V”: revision
- “XXXXXXXXX”: lot number

50.2 AT91SAM9263 Errata - Revision “A” Parts

Refer to Section 50.1 “Marking” on page 1059.

50.2.1 Main Oscillator

50.2.1.1 Main Oscillator: Spurious Malfunction of Main Oscillator

The main oscillator can exhibit spurious malfunction as exhibited by the duty-cycle not performing according to specification, missing main clock periods, or in the worst case, no main clock at all. The behavior does not alter over time. A device fully tested over the operating range (including power supply and temperature) in an application without exhibiting any malfunction will continue to function according to specifications.

Problem Fix/Workaround

Placing a series resistor of 470 Ω between XOUT and the crystal will fix the issue.

The additional series resistor will not alter the device specification as documented in the datasheet, including the startup time and power consumption.

50.2.2 Two D Graphic Controller (TDGC)

50.2.2.1 Polygon Fill Function

Polygon fill function is not functional.

Problem Fix/Workaround

None.

50.2.2.2 TDGC Clipping Function

The Clipping function is not functional.

Problem Fix/Workaround

None.

50.2.3 AC97

50.2.3.1 Bad Management of Endianness Conversion

When the transfer size is not a multiple of bytes, the Endianness is incorrect.

Problem Fix/Workaround

None.

50.2.4 BMS

50.2.4.1 BMS Does Not Have Correct State

The AT91SAM9263 device samples the BMS signal on the rising edge of proc_nreset (refer to RSTC) and so one slow clock cycle after the NRST rising edge. NRST cannot be used to isolate BMS (multiplexed with AC97_RX) at reset.

Problem Fix/Workaround

None.

50.2.5 Bus Matrix

50.2.5.1 Problem with Locked Transfers

Locked transfers are not correctly handled by the Bus Matrix and can lead to a system freeze-up.

This does not concern ARM locked transfers.

Problem Fix/Workaround

Avoid other Bus Matrix masters locked transfers.

50.2.6 CAN

50.2.6.1 CAN: Low Power Mode and Error Frame

If the Low Power Mode is activated while the CAN is generating an error frame, this error frame may be shortened.

Problem Fix/Workaround

None

50.2.6.2 CAN: Low Power Mode and Pending Transmit Messages

No pending transmit messages may be sent once the CAN Controller enters Low-power Mode.

Problem Fix/Workaround

Check that all messages have been sent by reading the related Flags before entering Low-power Mode.

50.2.6.3 CAN: Contents of Mailbox 0 can be sent Even if Mailbox is Disabled

When a transmit request (on a mailbox other than mailbox 0) is done simultaneously with the RX line going low, the CAN may send the contents of mailbox 0 even if this mailbox is disabled or has been set in reception.

The contents of mailbox 0 are sent, followed by the mailbox which is supposed to be sent.

Note that this is unlikely to happen and only occurs with a high bus load.

In this case, when the CAN attempts to send mailbox 0, the mailbox 0 ID is used for arbitration. If arbitration is lost, the send request for mailbox 0 is canceled, else the contents of mailbox 0 are sent. The original transmit request is correctly handled afterwards.

Problem Fix / Workaround:

Two workarounds:

1. Do not use mailbox 0 and program an unused identifier (example: 0x7FF) which will not affect the CAN bus. The spurious emission may happen but the impact on the bus will be minimized.

   Or:

2. Use only mailbox 0 to transmit and no other mailboxes. In this case, there will be no spurious emission on the CAN bus.

50.2.7 ECC

50.2.7.1 ECC status may be wrong with external SRAM

When the data bus width is different for an SRAM on any EBI NCS and the NAND Flash, the ECC status is wrong. A single error is seen as a multiple error and is not corrected.This does not occur with SDRAM.

Problem Fix/Workaround

None.

50.2.8 EMACB

50.2.8.1 Transmit Underrun Errors

EMACB FIFO internal arbitration scheme is:

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

EMACB master interface releases the AHB bus between two transfers.

EMACB has the highest priority.

If we are in a state where EMACB RX and TX FIFOs have requests pending, the following sequence occurs:

1. EMACB RX FIFO write (burst 4)
2. EMACB release the AHB bus
3. The AHB matrix can grant an another master (ARM I or D for example)
4. AHB matrix re-arbitration (finish at least the current word/halfword/byte)
5. The AHB matrix grants the EMACB
6. The EMACB TX FIFO read (burst 4)

In a case of a slow memory and/or a special operation such as SDRAM refresh or SDRAM bank opening /closing, there may be TX underrun (latency min 960 ns).

Problem Fix/Workaround

Reduce re-arbitration time between RX and TX EMACB transfer by using internal SRAM (or another slave with a short access time) for transmit buffers and descriptors.

50.2.9 LCD

50.2.9.1 LCD Screen Shifting After a Reset

When a FIFO underflow occurs, a reset of the DMA and FIFO pointers is necessary. Performing the following sequence:

- DMA disable
- Wait for DMABUSY
- DMA reset
- DMA enable

leads to reset DMA pointers but not FIFO pointers, the displayed image is shifted.

Problem Fix/Workaround

Apply the following sequence:

- LCD power off
- DMA disable
- Wait for DMABUSY
- DMA reset
- LCD power on
- DMA enable.

50.2.9.2 LCD Periodic Bad Pixels

LCD periodic bad pixels is due to mis-aligned DMA base address in frame buffer. LCD DMA performs bursts to read memory. These bursts must not cross 1 Kb AMBA boundary.

Problem Fix/Workaround

The burst size in 32-bit words is programmed by field BRSTLN in DMA_FRMCFG register.

The LCD DMA Base Address is programmed in DMA_BADDR1 register.

DMA Base Address must be programmed with a value aligned onto LCD DMA burst size.

e.g.:

BRSTLN = 15

For a 16-word burst, the LCD DMA Base Address must start on 16-word offset: 0x0, 0x40, 0x80 or 0xc0.

BRSTLN = 3

For a 4-word burst, the LCD DMA Base Address must start on 0x0, 0x10, ..., 0xf0.

50.2.9.3 24-bit Packed Mode

LCD DMA Base Address and LCD DMA burst size must be selected with care in 24-bit packed mode. A 32-bit word contains some bits of a pixel and some bits of the following. If LCD DMA Base Address is not aligned with a pixel start, the colors will be modified.

Respect "LCD periodic bad pixels" erratum constrains lead to select the LCD DMA Base Address regarding the LCD DMA burst size.

Problem Fix/Workaround

LCD DMA Base Address is to be set on a pixel start, every three 32-bit word.

The offset of the LCD DMA Base Address must be a multiple of 0x30 plus 0x0, 0xc, 0x18 or 0x24. (0x0, 0xc, 0x18, 0x24, 0x30, 0x3c, 0x48, 0x54, 0x60,0x6c, 0x78, 0x84, 0x90, 0x9c, 0xa8, 0xb4, 0xc0 ...)

e.g. regarding the bursts size:

1) BRSTLN = 3 implies the following LCD DMA Base Address offsets: 0x0, 0x30, 0x60, ...

2) BRSTLN = 15 implies the following LCD DMA Base Address offsets: 0x0 and 0xc0 only

50.2.10 MCI

50.2.10.1 Busy signal of R1b responses is not taken in account

The busy status of the card during the response (R1b) is ignored for the commands CMD7, CMD28, CMD29, CMD38, CMD42, CMD56.

Additionally, for commands CMD42 and CMD56, a conflict can occur on data line 0 if the MCI sends data to the card while the card is still busy.

The behavior is correct for CMD12 command (STOP_TRANSFER).

Problem Fix/Workaround

None

50.2.10.2 SDIO interrupt does not work with slots other than A

If there is a 1-bit data bus width on other slots than Slot A, the SDIO interrupt cannot be captured. The sample is made on the wrong data line.

Problem Fix/Workaround

None

50.2.10.3 Data Timeout Error Flag

As the data Timeout error flag checking the Naac timing cannot rise, the MCI can be stalled waiting indefinitely the Data start bit.

Problem Fix/Workaround

A STOP command must be sent with a software timeout.

50.2.10.4 Data Write Operation and Number of Bytes

The Data Write operation with a number of bytes less than 12 is impossible.

Problem Fix/Workaround

The PDC counters must always be equal to 12 bytes for data transfers lower than 12 bytes. The BLKLEN or BCNT field are used to specify the real count number.

50.2.10.5 Flag Reset is not correct in half duplex mode

In half duplex mode, the reset of the flags ENDRX, RXBUFF, ENDTX and TXBUFE can be incorrect.

These flags are reset correctly after a PDC channel enable.

Problem Fix/Workaround

Enable the interrupts related to ENDRX, ENDTX, RXBUFF and TXBUFE only after enabling the PDC channel by writing PDC_TXTEN or PDC_RXTEN.

50.2.11 NTRST

50.2.11.1 NTRST: Device does not boot correctly due to power-up sequencing issue

The NTRST signal is powered by VDDIOP power supply (3.3V) and the ARM processor is powered by VDDCORE power supply (1.2V).

During the power-up sequence, if VDDIOP power supply is not established whereas the VDDCORE Power On Reset output is released, the NTRST signal is not correctly asserted. This

leads to a bad reset of the Embedded Trace Macrocell (ETM9). The ARM processor then enters debug state and the device does not boot correctly.

Problem Fix/Workaround

1. Connect NTRST pin to NRST pin to ensure that a correct powering sequence takes place in all cases.
2. Connect NTRST to GND if no debug capabilities are required.

50.2.12 ROM Code

50.2.12.1 SDCard Boot is Not Functional

SDCard Boot is not functional in this revision.

Problem Fix/Workaround

None.

50.2.12.2 NAND Flash Boot is Not Functional

NAND Flash Boot is not functional in this revision.

Problem Fix/Workaround

None.

50.2.13 SDRAM Controller

50.2.13.1 SDCLK Clock Active after Reset

After a reset, the SDRAM clock is always active leading to overconsumption in the pad.

Problem Fix/Workaround

The following sequence stops the SDRAM clock:

1. Set the bit LPCB in the SDRAMC Low Power Register.
2. Write 0 in the SDRAMC Mode Register and perform a dummy write in SDRAM to complete.

50.2.13.2 Mobile SDRAM Device Initialization Constraint

Using Mobile SDRAM devices that need to have their DQMx level HIGH during Mobile SDRAM device initialization may lead to data bus contention and thus external memories on the same EBI must not be accessed.

This does not apply to Mobile SDRAM devices whose DQMx level is “Don’t care” during this phase.

Problem Fix/Workaround

Mobile SDRAM initialization must be performed in internal SRAM.

50.2.13.3 JEDEC Standard Compatibility

In the current configuration, SDCKE rises at the same time as SDCK while exiting self-refresh mode. To be fully compliant with the JEDEC standard, SDCK must be STABLE before the rising edge of SDCKE.

It is not the case in this product.

Problem Fix/Workaround

Use a fully JEDEC compliant SDRAM module.

50.2.14 Static Memory Controller (SMC)

50.2.14.1 SMC Chip Select Parameters Modification

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.

For example, the modification of the Chip Select 0 (CS0) parameters while fetching the code from a memory on CS0, may lead to unpredictable behavior.

Problem Fix/Workaround:

The code used to modify the parameters of an SMC Chip Select can be executed from the internal RAM or from a memory connected to another Chip Select.

50.2.15 Serial Peripheral Interface (SPI)

50.2.15.1 SPI: PDC Data Loss

One byte data can be lost when PDC transmits. This occurs when write accesses are performed on the base address of any peripheral, during the PDC transfer.

Problem Fix/Workaround:

– Add a timeout for the PDC transfer and check the value of the PDC transmit counter when the timeout elapsed.

– Check the data integrity by a checksum.

– Avoid write access on the base address of peripherals during a PDC transfer.

50.2.15.2 SPI: Pulse Generation on SPCK

In Master Mode, there is an additional pulse generated on SPCK when the SPI is configured as follows:

– The Baudrate is odd and different from 1.

– The Polarity is set to 1.

– The Phase is set to 0.

Problem Fix/Workaround

None.

50.2.15.3 SPI: Bad PDC Behavior when CSAAT=1 and SCBR = 1

If the SPI2 is programmed with CSAAT = 1, SCBR (baudrate) = 1 and two transfers are performed consecutively on the same slave with an IDLE state between them, the second data is sent twice.

Problem Fix/Workaround

None. Do not use the combination CSAAT=1 and SCBR =1.

50.2.15.4 SPI: LASTXFER (Last Transfer) Behavior

In FIXED Mode with CSAAT bit set and in PDC Mode, the Chip Select can rise depending on the data written in the SPI_TDR when the TX_EMPTY flag is set. For example, if the PDC writes a "1" in bit 24 (LASTXFER bit) of the SPI_TDR, the Chip Select rises as soon as the TXEMPTY flag is set.

Problem Fix/Workaround

Use the CS in PIO mode when PDC Mode is required and CS has to be maintained between transfers.

50.2.15.5 SPI: Baudrate set to 1

When Baudrate is set to 1 (i.e. when serial clock frequency equals the system clock frequency), and when the fields BITS (number of bits to be transmitted) equals an ODD value (in this case 9,11,13 or 15), an additional pulse is generated on output SPCK. No such pulse occurs if BITS field equals 8,10,12,14 or 16 and Baudrate = 1.

Problem Fix/Workaround

None.

50.2.15.6 SPI: Software Reset

If the Software reset command is performed during the same clock cycle as an event for TXRDY, there is no reset.

Problem Fix/Workaround

Perform another a software reset.

50.2.15.7 SPI: Chip Select and Fixed Mode

In fixed Mode, if a transfer is performed through a PDC on a Chip Select different from the Chip Select 0, the output spi_size sampled by the PDC depends on the field BITS of SPI_CSR0 register, whatever the selected Chip select may be. For example, if CSR0 is configured for a 10-bit transfer, whereas the CSR1 is configured for an 8-bit transfer, when a transfer is performed in Fixed mode through the PDC on Chip Select 1, the transfer is considered to be a half-word transfer.

Problem Fix/Workaround

If a PDC transfer has to be performed in 8 bits on a Chip select y (y different from 0), the field BITS of the CSR0 must be configured in 8 bits in the same way as the field BITS of the CSRy Register.

50.2.15.8 SPI: Bad Serial Clock Generation on 2nd Chip Select

Bad Serial clock generation on the 2nd chip select when SCBR = 1, CPOL = 1 and NCPHA = 0.

This occurs using SPI with the following conditions:

- Master Mode
- CPOL = 1 and NCPHA = 0
- Multiple chip selects are used with one transfer with Baud rate (SCBR) equal to 1 (i.e., when serial clock frequency equals the system clock frequency) and the other transfers set with SCBR are not equal to 1
- Transmitting with the slowest chip select and then with the fastest one, then an additional pulse is generated on output SPCK during the second transfer.

Problem Fix/Workaround

Do not use a multiple Chip Select configuration where at least one SCRx register is configured with SCBR = 1 and the others differ from 1 if NCPHA = 0 and CPOL = 1.

If all chip selects are configured with Baudrate = 1, the issue does not appear.

50.2.15.9 *SPI: Software Reset Must be Written Twice*

If a software reset (SWRST in the control register) is performed, the SPI may not work properly (the clock is enabled before the chip select).

Problem Fix/Workaround

The SPI Control Register field, SWRST (Software Reset) needs to be written twice to be correctly set.

50.2.16 Serial Synchronous Controller (SSC)

50.2.16.1 *Transmitter Limitations in Slave Mode*

If TK is programmed as output and TF is programmed as input, it is impossible to emit data when start of edge (rising or falling) of synchro with a Start Delay equal to zero.

Problem Fix/Workaround

None.

50.2.16.2 *Periodic Transmission Limitations in Master Mode*

If Last Significant Bit is sent first (MSBF = 0), the first TAG during the frame synchro is not sent.

Problem Fix/Workaround

None.

50.2.16.3 *SSC: Last RK Clock Cycle when RK Outputs a Clock During Data Transfer*

When the SSC receiver is used with the following conditions:

- the internal clock divider is used (CKS = 0 and DIV different from 0)
- RK pin set as output and provides the clock during data transfer (CKO = 2)
- data sampled on RK falling edge (CKI = 0),

At the end of the data, the RK pin is set in high impedance which might be seen as an unexpected clock cycle.

Problem Fix/Workaround

Enable the pull-up on RK pin.

50.2.16.4 *SSC: First RK Clock Cycle when Rk Outputs a Clock During Data Transfer*

When the SSC receiver is used with the following conditions:

- RX clock is divided clock (CKS = 0 and DIV different from 0)
- RK pin set as output and provides the clock during data transfer (CKO = 2)
- data sampled on RK falling edge (CKI = 0),

The first clock cycle time generated by the RK pin is equal to MCK/(2 x (value +1)).

Problem Fix/Workaround

None.

50.2.17 System Controller

50.2.17.1 *Possible Event Loss when Reading RTT_SR*

If an event (RTTINC or ALMS) occurs within the same slow clock cycle as when RTT_SR is read, the corresponding bit may be cleared. This may lead to the loss of this event.

Problem Fix/Workaround

The software must handle the RTT event as an interrupt and should not poll RTT_SR.

50.2.18 Two-wire Interface (TWI)

50.2.18.1 Clock Divider

The value of CLDIV x 2^{CKDIV} must be less than or equal to 8191, the value of CHDIV x 2^{CKDIV} must be less than or equal to 8191·

Problem Fix/Workaround

None.

50.2.18.2 Disabling Does not Operate Correctly

Any transfer in progress is immediately frozen if the Control Register (TWI_CR) is written with the bit MSDIS at 1. Furthermore, the status bits TXCOMP and TXRDY in the Status Register (TWI_SR) are not reset.

Problem Fix/Workaround

The user must wait for the end of transfer before disabling the TWI. In addition, the interrupts must be disabled before disabling the TWI.

50.2.18.3 Software Reset

When a software reset is performed during a frame and when TWCK is low, it is impossible to initiate a new transfer in READ or WRITE mode.

Problem Fix/Workaround

None.

50.2.18.4 STOP not Generated

If the sequence described as follows occurs:

1. WRITE 1 or more bytes at a given address.
2. Send a STOP.
3. Wait for TXCOMP flag.
4. READ (or WRITE) 1 or more bytes at the same address.

The STOP is not generated.

The line shows: DADR BYTE 1, ..., BYTE n, NO STOP generated, BYTE 1, ..., BYTE n.

Problem Fix/Workaround

Insert a delay of one TWI clock period before step 4.

50.2.19 UDP

50.2.19.1 Bad Data in the First IN Data Stage

All or part of the data of the first IN data Stage are not transmitted. It may then be a Zero Length Packet. The CRC is correct. Thus the HOST may only see that the size of the received data does not match the requested length. But even if performed again, the control transfer probably fails.

Problem Fix/Workaround

Control transfers are mainly used at device configuration. After clearing RXSETUP, the software needs to compute the setup transaction request before writing data into the FIFO if needed. This time is generally greater than the minimum safe delay required above. If not, a software wait loop after RXSETUP clear may be added at minimum cost.

50.2.20 UHP

50.2.20.1 Non-ISO IN Transfers

Conditions:

Consider the following scenario:

1. The Host controller issues an IN token.
2. The Device provides the IN data in a short packet.
3. The Host controller writes the received data to the system memory.
4. The Host controller is now supposed to do two Write transactions (TD status write and TD retirement write) to the system memory in order to complete the status update.
5. Host controller raises the request for the first write transaction. By the time the transaction is completed, a frame boundary is crossed.
6. After completing the first write transaction, the Host controller skips the second write transaction.

Consequence: When this error occurs, the Host controller tries the same IN token again.

Problem Fix/Workaround

This problem can be avoided if the system guarantees that the status update can be completed within the same frame.

50.2.20.2 ISO OUT Transfers

Conditions:

Consider the following scenario:

1. The Host controller sends an ISO OUT token after fetching 16 bytes of data from the system memory.
2. When the Host controller is sending the ISO OUT data, because of system latencies, remaining bytes of the packet are not available. This results in a buffer underrun condition.
3. While there is an underrun condition, if the Host controller is in the process of bit-stuffing, it causes the Host controller to hang.

Consequence: After the failure condition, the Host controller stops sending the SOF. This causes the connected device to go into suspend state.

Problem Fix/Workaround

This problem can be avoided if the system can guarantee that no buffer underrun occurs during the transfer.

50.2.20.3 Remote Wakeup Event

Conditions:

When a Remote Wakeup event occurs on a downstream port, the OHCI Host controller begins to send resume signaling to the device. The Host controller should send this resume signaling for 20 ms. However, if the driver sets the HcControl.HCFS into USBOPERATIONAL state during

the resume event, then the Host controller terminates sending the resume signal with an EOP to the device.

Consequence: If the Device does not recognize the resume (<20 ms) event then the Device, it will remain in the suspend state.

Problem Fix/Workaround

Host stack can do a port resume after it sets the HcControl.HCFS to USBOPERATIONAL.

50.2.21 USART

50.2.21.1 SCK1 and SCK2 are Inverted

SCK1 and SCK2 clocks are inverted on the PIO controller, but the enable of the clocks is correct.

This makes it impossible to use USART1 and USART2 in Synchronous Mode.

Problem Fix/ Workaround

To use USART1 in Synchronous Mode, the user must program USART1 and USART2 with exactly the same configuration. SCK2 clock will output on PD10.

Note: EBI0_CFCE2 usage on "periph A" is not forbidden because PD9 is SCK1 which is not used.

50.2.21.2 RXBRK Flag Error in Asynchronous Mode

When timeguard is 0, RXBRK is not set when the break character is located just after the Stop Bit. FRAME (Frame Error) is set instead.

Problem Fix/Workaround

Timeguard should be > 0.

50.2.21.3 RTS not Expected Behavior

1. Setting the receiver to hardware handshaking mode drops RTS line to low level even if the receiver is still turned off. USART needs to be completely configured and started before setting the receiver to hardware handshaking mode.
2. Disabling the receiver during a PDC transfer while RXBUFF flag is '0' has no effect on RTS. The only way to get the RTS line to rise to high level is to reset both PDMA buffers by writing the value '0' in both counter registers.

Problem Fix/Workaround

None.

50.2.21.4 Two characters sent if CTS rises during emission

If CTS rises to 1 during a character transmit, the Transmit Holding Register is also transmitted if not empty.

Problem Fix/Workaround

None.

50.2.21.5 TXD signal is floating in Modem and Hardware Handshaking mod

TXD signal should be pulled up in Modem and Hardware Handshaking mode.

Problem Fix/Workaround

TXD is multiplexed with PIO which integrates a pull up resistor. This internal pull-up must be enabled.

50.2.21.6 *DCD is Active High instead of Low*

The DCD signal is active at High level in the USART Modem Mode.

DCD should be active at Low level.

Problem Fix/Workaround

Add an inverter.

50.3 AT91SAM9263 Errata - Revision “B” Parts

Refer to Section 50.1 “Marking” on page 1059.

50.3.1 Main Oscillator

50.3.1.1 Main Oscillator: Spurious Malfunction of Main Oscillator

The main oscillator can exhibit spurious malfunction as exhibited by the duty-cycle not performing according to specification, missing main clock periods, or in the worst case, no main clock at all. The behavior does not alter over time. A device fully tested over the operating range (including power supply and temperature) in an application without exhibiting any malfunction will continue to function according to specifications.

Problem Fix/Workaround

Placing a series resistor of 470 Ω between XOUT and the crystal will fix the issue.

The additional series resistor will not alter the device specification as documented in the datasheet, including the startup time and power consumption.

50.3.2 Two D Graphic Controller (TDGC)

50.3.2.1 Polygon Fill Function

Polygon fill function is not functional.

Problem Fix/Workaround

None.

50.3.2.2 TDGC Clipping Function

The Clipping function is not functional.

Problem Fix/Workaround

None.

50.3.3 AC97

50.3.3.1 Bad Management of Endianness Conversion

When the transfer size is not a multiple of bytes, the Endianness is incorrect.

Problem Fix/Workaround

None.

50.3.4 Bus Matrix

50.3.4.1 Problem with Locked Transfers

Locked transfers are not correctly handled by the Bus Matrix and can lead to a system freeze-up.

This does not concern ARM locked transfers.

Problem Fix/Workaround

Avoid other Bus Matrix masters locked transfers.

50.3.5 CAN

50.3.5.1 CAN: Low Power Mode and Error Frame

If the Low Power Mode is activated while the CAN is generating an error frame, this error frame may be shortened.

Problem Fix/Workaround

None

50.3.5.2 CAN: Low Power Mode and Pending Transmit Messages

No pending transmit messages may be sent once the CAN Controller enters Low-power Mode.

Problem Fix/Workaround

Check that all messages have been sent by reading the related Flags before entering Low-power Mode.

50.3.5.3 CAN: Contents of Mailbox 0 can be sent Even if Mailbox is Disabled

When a transmit request (on a mailbox other than mailbox 0) is done simultaneously with the RX line going low, the CAN may send the contents of mailbox 0 even if this mailbox is disabled or has been set in reception.

The contents of mailbox 0 are sent, followed by the mailbox which is supposed to be sent.

Note that this is unlikely to happen and only occurs with a high bus load.

In this case, when the CAN attempts to send mailbox 0, the mailbox 0 ID is used for arbitration. If arbitration is lost, the send request for mailbox 0 is canceled, else the contents of mailbox 0 are sent. The original transmit request is correctly handled afterwards.

Problem Fix / Workaround:

Two workarounds:

1. Do not use mailbox 0 and program an unused identifier (example: 0x7FF) which will not affect the CAN bus. The spurious emission may happen but the impact on the bus will be minimized.

   Or:

2. Use only mailbox 0 to transmit and no other mailboxes. In this case, there will be no spurious emission on the CAN bus.

50.3.6 ECC

50.3.6.1 ECC status may be wrong with external SRAM

When the data bus width is different for an SRAM on any EBI NCS and the NAND Flash, the ECC status is wrong. A single error is seen as a multiple error and is not corrected.This does not occur with SDRAM.

Problem Fix/Workaround

None.

50.3.7 EMACB

50.3.7.1 Transmit Underrun Errors

EMACB FIFO internal arbitration scheme is:

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

EMACB master interface releases the AHB bus between two transfers.

EMACB has the highest priority.

If we are in a state where EMACB RX and TX FIFOs have requests pending, the following sequence occurs:

1. EMACB RX FIFO write (burst 4)
2. EMACB release the AHB bus
3. The AHB matrix can grant an another master (ARM I or D for example)
4. AHB matrix re-arbitration (finish at least the current word/halfword/byte)
5. The AHB matrix grants the EMACB
6. The EMACB TX FIFO read (burst 4)

In a case of a slow memory and/or a special operation such as SDRAM refresh or SDRAM bank opening /closing, there may be TX underrun (latency min 960 ns).

Problem Fix/Workaround

Reduce re-arbitration time between RX and TX EMACB transfer by using internal SRAM (or another slave with a short access time) for transmit buffers and descriptors.

50.3.8 LCD

50.3.8.1 LCD Screen Shifting After a Reset

When a FIFO underflow occurs, a reset of the DMA and FIFO pointers is necessary. Performing the following sequence:

- DMA disable
- Wait for DMABUSY
- DMA reset
- DMA enable

lead to well reset DMA pointers but not FIFO pointers, the displayed image is shifted.

Problem Fix/Workaround

Apply the following sequence:

- LCD power off
- DMA disable
- Wait for DMABUSY
- DMA reset
- LCD power on
- DMA enable.

50.3.8.2 *LCD Periodic Bad Pixels*

LCD periodic bad pixels is due to mis-aligned DMA base address in frame buffer. LCD DMA performs bursts to read memory. These bursts must not cross 1Kb AMBA boundary.

Problem Fix/Workaround

The burst size in 32-bit words is programmed by field BRSTLN in DMAFRMCFG register.

The LCD DMA Base Address is programmed in DMABADDR1 register.

DMA Base Address must be programmed with a value aligned onto LCD DMA burst size.

e.g.:

BRSTLN = 15

For a 16-word burst, the LCD DMA Base Address must start on 16-word offset: 0x0, 0x40, 0x80 or 0xc0.

BRSTLN = 3

For a 4-word burst, the LCD DMA Base Address must start on 0x0, 0x10, ..., 0xf0.

50.3.8.3 *24-bit Packed Mode*

LCD DMA Base Address and LCD DMA burst size must be selected with care in 24-bit packed mode. A 32-bit word contains some bits of a pixel and some bits of the following. If LCD DMA Base Address is not aligned with a pixel start, the colors will be modified.

Respect "LCD periodic bad pixels" erratum constraints lead to select the LCD DMA Base Address regarding the LCD DMA burst size.

Problem Fix/Workaround

LCD DMA Base Address is to be set on a pixel start, every three 32-bit word.

The offset of the LCD DMA Base Address must be a multiple of 0x30 plus 0x0, 0xc, 0x18 or 0x24. (0x0, 0xc, 0x18, 0x24, 0x30, 0x3c, 0x48, 0x54, 0x60,0x6c, 0x78, 0x84, 0x90, 0x9c, 0xa8, 0xb4, 0xc0 ...)

e.g. regarding the bursts size:

1) BRSTLN = 3 implies the following LCD DMA Base Address offsets: 0x0, 0x30, 0x60, ...

2) BRSTLN = 15 implies the following LCD DMA Base Address offsets: 0x0 and 0xc0 only

50.3.9 MCI

50.3.9.1 *Busy signal of R1b responses is not taken in account*

The busy status of the card during the response (R1b) is ignored for the commands CMD7, CMD28, CMD29, CMD38, CMD42, CMD56.

Additionally, for commands CMD42 and CMD56, a conflict can occur on data line 0 if the MCI sends data to the card while the card is still busy.

The behavior is correct for CMD12 command (STOP_TRANSFER).

Problem Fix/Workaround

None

50.3.9.2 SDIO interrupt does not work with slots other than A

If there is a 1-bit data bus width on other slots than Slot A, the SDIO interrupt cannot be captured. The sample is made on the wrong data line.

Problem Fix/Workaround

None

50.3.9.3 Data Timeout Error Flag

As the data Timeout error flag checking the Naac timing cannot rise, the MCI can be stalled waiting indefinitely the Data start bit.

Problem Fix/Workaround

A STOP command must be sent with a software timeout.

50.3.9.4 Data Write Operation and Number of Bytes

The Data Write operation with a number of bytes less than 12 is impossible.

Problem Fix/Workaround

The PDC counters must always be equal to 12 bytes for data transfers lower than 12 bytes. The BLKLEN or BCNT field are used to specify the real count number.

50.3.9.5 Flag Reset is not correct in half duplex mode

In half duplex mode, the reset of the flags ENDRX, RXBUFF, ENDTX and TXBUFE can be incorrect.

These flags are reset correctly after a PDC channel enable.

Problem Fix/Workaround

Enable the interrupts related to ENDRX, ENDTX, RXBUFF and TXBUFE only after enabling the PDC channel by writing PDC_TXTEN or PDC_RXTEN.

50.3.10 NTRST

50.3.10.1 NTRST: Device does not boot correctly due to power-up sequencing issue

The NTRST signal is powered by VDDIOP power supply (3.3V) and the ARM processor is powered by VDDCORE power supply (1.2V).

During the power-up sequence, if VDDIOP power supply is not established whereas the VDDCORE Power On Reset output is released, the NTRST signal is not correctly asserted. This leads to a bad reset of the Embedded Trace Macrocell (ETM9). The ARM processor then enters debug state and the device does not boot correctly.

Problem Fix/Workaround

1. Connect NTRST pin to NRST pin to ensure that a correct powering sequence takes place in all cases.
2. Connect NTRST to GND if no debug capabilities are required.

50.3.11 Reset Controller (RSTC)

50.3.11.1 RSTC: ERSTL Default Value is 1

The default value of ERSTL field in RSTC_MR register has been changed from 0x0 to 0x1. This means that the NRST line rises 4 cycles after backup_nreset

Problem Fix/Workaround

None

50.3.11.2 RSTC: Reset during SDRAM Accesses

When a User Reset occurs during SDRAM read access, the SDRAM clock is turned off while data is ready to be read on the data bus. The SDRAM maintains the data until the clock restarts.

If the User Reset is programmed to assert a general reset, the data maintained by the SDRAM leads to a data bus conflict and adversely affects the boot memories connected on the EBI:

- NAND Flash boot functionality, if the system boots out of internal ROM.
- NOR Flash boot, if the system boots on an external memory connected on the EBI CS0.

Problem Fix/Workaround

1. Avoid User Reset to generate a system reset.
2. Trap the User Reset with an interrupt.

In the interrupt routine, power down the SDRAM properly and perform Peripheral and Processor Reset with software in assembler.

Example with libV3.

- The main code:

```
//user reset interrupt setting
  // Configure AIC controller to handle SSC interrupts
  AT91F_AIC_ConfigureIt (
    AT91C_BASE_AIC,                         // AIC base address
    AT91C_ID_SYS,                          // System peripheral ID
    AT91C_AIC_PRIOR_HIGHEST,                   // Max priority
    AT91C_AIC_SRCTYPE_INT_EDGE_TRIGGERED, // Level sensitive
    sysc_handler );

  // Enable SYSC interrupt in AIC
  AT91F_AIC_EnableIt(AT91C_BASE_AIC, AT91C_ID_SYS);

*AT91C_RSTC_RMR = (0xA5<<24) | (0x4<<8) | AT91C_RSTC_URSTIEN;
```

- The C SYS handler:

```
extern void soft_user_reset(void);

void sysc_handler(void){
//check if interrupt comes from RSTC
  if( (*AT91C_RSTC_RSR & AT91C_RSTC_URSTS ) == AT91C_RSTC_URSTS){
  soft_user_reset();
  //never reached
  while(1);

  }
}
```

- The assembler routine:

```
AREA       TEST, CODE
```

```
  INCLUDEAT91SAM9xxx.inc
  EXPORTsoft_user_reset

soft_user_reset
;disable IRQs
  MRS r0, CPSR
  ORR r0, r0, #0x80
  MSR CPSR_c, r0

;change refresh rate to block all data accesses
  LDR r0, =AT91C_SDRAMC_TR
  LDR r1, =1
  STR r1, [r0]

;prepare power down command
  LDR r0, =AT91C_SDRAMC_LPR
  LDR r1, =2

;prepare proc_reset and periph_reset
  LDR r2, =AT91C_RSTC_RCR
  LDR r3, =0xA5000005

;perform power down command
  STR r1, [r0]
;perform proc_reset and periph_reset (in the ARM pipeline)
  STR r3, [r2]

  END
```

50.3.12 SDRAM Controller

50.3.12.1 Mobile SDRAM Device Initialization Constraint

Using Mobile SDRAM devices that need to have their DQMx level HIGH during Mobile SDRAM device initialization may lead to data bus contention and thus external memories on the same EBI must not be accessed.

This does not apply to Mobile SDRAM devices whose DQMx level is “Don’t care” during this phase.

Problem Fix/Workaround

Mobile SDRAM initialization must be performed in internal SRAM.

50.3.13 Static Memory Controller (SMC)

50.3.13.1 SMC Chip Select Parameters Modification

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.

For example, the modification of the Chip Select 0 (CS0) parameters while fetching the code from a memory on CS0, may lead to unpredictable behavior.

Problem Fix/Workaround:

The code used to modify the parameters of an SMC Chip Select can be executed from the internal RAM or from a memory connected to another Chip Select.

50.3.14 Serial Peripheral Interface (SPI)

50.3.14.1 SPI: Pulse Generation on SPCK

In Master Mode, there is an additional pulse generated on SPCK when the SPI is configured as follows:

- The Baudrate is odd and different from 1.
- The Polarity is set to 1.
- The Phase is set to 0.

Problem Fix/Workaround

None.

50.3.14.2 SPI: Bad PDC Behavior when CSAAT=1 and SCBR = 1

If the SPI2 is programmed with CSAAT = 1, SCBR (baudrate) = 1 and two transfers are performed consecutively on the same slave with an IDLE state between them, the second data is sent twice.

Problem Fix/Workaround

None. Do not use the combination CSAAT=1 and SCBR =1.

50.3.14.3 SPI: LASTXFER (Last Transfer) Behavior

In FIXED Mode with CSAAT bit set and in PDC Mode, the Chip Select can rise depending on the data written in the SPI_TDR when the TX_EMPTY flag is set. For example, if the PDC writes a "1" in bit 24 (LASTXFER bit) of the SPI_TDR, the Chip Select rises as soon as the TXEMPTY flag is set.

Problem Fix/Workaround

Use the CS in PIO mode when PDC Mode is required and CS has to be maintained between transfers.

50.3.14.4 SPI: Baudrate Set to 1

When Baudrate is set to 1 (i.e. when serial clock frequency equals the system clock frequency), and when the fields BITS (number of bits to be transmitted) equals an ODD value (in this case 9,11,13 or 15), an additional pulse is generated on output SPCK. No such pulse occurs if BITS field equals 8,10,12,14 or 16 and Baudrate = 1.

Problem Fix/Workaround

None.

50.3.14.5 SPI: Software Reset

If the Software reset command is performed during the same clock cycle as an event for TXRDY, there is no reset.

Problem Fix/Workaround

Perform another a software reset.

50.3.14.6 SPI: SPI Software Reset Must Be Written Twice

The SPI Control Register field **SWRST** (Software Reset) needs to be written twice to be correctly set.

Problem Fix/Workaround

None.

50.3.14.7 SPI: Chip Select and Fixed Mode

In fixed Mode, if a transfer is performed through a PDC on a Chip Select different from the Chip Select 0, the output spi_size sampled by the PDC depends on the field BITS of SPI_CSR0 register, whatever the selected Chip select may be. For example, if CSR0 is configured for a 10-bit transfer, whereas the CSR1 is configured for an 8-bit transfer, when a transfer is performed in Fixed mode through the PDC on Chip Select 1, the transfer is considered to be a half-word transfer.

Problem Fix/Workaround

If a PDC transfer has to be performed in 8 bits on a Chip select y (y different from 0), the field BITS of the CSR0 must be configured in 8 bits in the same way as the field BITS of the CSRy Register.

50.3.14.8 SPI: Software Reset Must be Written Twice

If a software reset (SWRST in the control register) is performed, the SPI may not work properly (the clock is enabled before the chip select).

Problem Fix/Workaround

The SPI Control Register field, SWRST (Software Reset) needs to be written twice to be correctly set.

50.3.15 Serial Synchronous Controller (SSC)

50.3.15.1 Transmitter Limitations in Slave Mode

If TK is programmed as output and TF is programmed as input, it is impossible to emit data when start of edge (rising or falling) of synchro with a Start Delay equal to zero.

Problem Fix/Workaround

None.

50.3.15.2 Periodic Transmission Limitations in Master Mode

If Last Significant Bit is sent first (MSBF = 0), the first TAG during the frame synchro is not sent.

Problem Fix/Workaround

None.

50.3.15.3 Unexpected Delay on TD output

When SSC is configured with the following conditions:

- TCMR.STTDLY more than 0
- RCMR.START = Start on falling edge / Start on Rising edge / Start on any edge
- RFMR.FSOS = None (input)
- TCMR.START = Receive Start

An unexpected delay of 2 or 3 system clock cycles is added to TD output.

Problem Fix/Workaround

None.

50.3.16 Pulse Width Modulation (PWM)

50.3.16.1 Zero Period

It is impossible to update a period equal to 0 by using the PWM_CUPD register.

Problem Fix/Workaround

None.

50.3.17 System Controller

50.3.17.1 Possible Event Loss when Reading RTT_SR

If an event (RTTINC or ALMS) occurs within the same slow clock cycle as when RTT_SR is read, the corresponding bit may be cleared. This may lead to the loss of this event.

Problem Fix/Workaround

The software must handle the RTT event as an interrupt and should not poll RTT_SR.

50.3.18 Two-wire Interface (TWI)

50.3.18.1 Clock Divider

The value of CLDIV x 2^{CKDIV} must be less than or equal to 8191, the value of CHDIV x 2^{CKDIV} must be less than or equal to 8191·

Problem Fix/Workaround

None.

50.3.18.2 Disabling Does Not Operate Correctly

Any transfer in progress is immediately frozen if the Control Register (TWI_CR) is written with the bit MSDIS at 1. Furthermore, the status bits TXCOMP and TXRDY in the Status Register (TWI_SR) are not reset.

Problem Fix/Workaround

The user must wait for the end of transfer before disabling the TWI. In addition, the interrupts must be disabled before disabling the TWI.

50.3.18.3 Software Reset

When a software reset is performed during a frame and when TWCK is low, it is impossible to initiate a new transfer in READ or WRITE mode.

Problem Fix/Workaround

None.

50.3.18.4 STOP not Generated

If the sequence described as follows occurs:

1. WRITE 1 or more bytes at a given address.
2. Send a STOP.
3. Wait for TXCOMP flag.
4. READ (or WRITE) 1 or more bytes at the same address.

The STOP is not generated.

The line shows: DADR BYTE 1, ..., BYTE n, NO STOP generated, BYTE 1, ..., BYTE n.

Problem Fix/Workaround

Insert a delay of one TWI clock period before step 4.

50.3.19 UDP

50.3.19.1 Bad Data in the First IN Data Stage

All or part of the data of the first IN data Stage are not transmitted. It may then be a Zero Length Packet. The CRC is correct. Thus the HOST may only see that the size of the received data does not match the requested length. But even if performed again, the control transfer probably fails.

Problem Fix/Workaround

Control transfers are mainly used at device configuration. After clearing RXSETUP, the software needs to compute the setup transaction request before writing data into the FIFO if needed. This time is generally greater than the minimum safe delay required above. If not, a software wait loop after RXSETUP clear may be added at minimum cost.

50.3.20 UHP

50.3.20.1 Non-ISO IN Transfers

Conditions:

Consider the following scenario:

1. The Host controller issues an IN token.
2. The Device provides the IN data in a short packet.
3. The Host controller writes the received data to the system memory.
4. The Host controller is now supposed to do two Write transactions (TD status write and TD retirement write) to the system memory in order to complete the status update.
5. Host controller raises the request for the first write transaction. By the time the transaction is completed, a frame boundary is crossed.
6. After completing the first write transaction, the Host controller skips the second write transaction.

Consequence: When this error occurs, the Host controller tries the same IN token again.

Problem Fix/Workaround

This problem can be avoided if the system guarantees that the status update can be completed within the same frame.

50.3.20.2 ISO OUT Transfers

Conditions:

Consider the following scenario:

1. The Host controller sends an ISO OUT token after fetching 16 bytes of data from the system memory.
2. When the Host controller is sending the ISO OUT data, because of system latencies, remaining bytes of the packet are not available. This results in a buffer underrun condition.
3. While there is an underrun condition, if the Host controller is in the process of bit-stuffing, it causes the Host controller to hang.

Consequence: After the failure condition, the Host controller stops sending the SOF. This causes the connected device to go into suspend state.

Problem Fix/Workaround

This problem can be avoided if the system can guarantee that no buffer underrun occurs during the transfer.

50.3.20.3 Remote Wakeup Event

Conditions:

When a Remote Wakeup event occurs on a downstream port, the OHCI Host controller begins to send resume signaling to the device. The Host controller should send this resume signaling for 20 ms. However, if the driver sets the HcControl.HCFS into USBOPERATIONAL state during the resume event, then the Host controller terminates sending the resume signal with an EOP to the device.

Consequence: If the Device does not recognize the resume (<20 ms) event then the Device, it will remain in the suspend state.

Problem Fix/Workaround

Host stack can do a port resume after it sets the HcControl.HCFS to USBOPERATIONAL.

50.3.21 USART

50.3.21.1 RXBRK Flag Error in Asynchronous Mode

When timeguard is 0, RXBRK is not set when the break character is located just after the Stop Bit. FRAME (Frame Error) is set instead.

Problem Fix/Workaround

Timeguard should be > 0.

50.3.21.2 RTS not Expected Behavior

1. Setting the receiver to hardware handshaking mode drops RTS line to low level even if the receiver is still turned off. USART needs to be completely configured and started before setting the receiver to hardware handshaking mode.
2. Disabling the receiver during a PDC transfer while RXBUFF flag is '0' has no effect on RTS. The only way to get the RTS line to rise to high level is to reset both PDMA buffers by writing the value '0' in both counter registers.

Problem Fix/Workaround

None.

50.3.21.3 Two characters sent if CTS rises during emission

If CTS rises to 1 during a character transmit, the Transmit Holding Register is also transmitted if not empty.

Problem Fix/Workaround

None.

50.3.21.4 TXD signal is floating in Modem and Hardware Handshaking mod

TXD signal should be pulled up in Modem and Hardware Handshaking mode.

Problem Fix/Workaround

TXD is multiplexed with PIO which integrates a pull up resistor. This internal pull-up must be enabled.

50.3.21.5 DCD is Active High instead of Low

The DCD signal is active at High level in the USART Modem Mode.

DCD should be active at Low level.

Problem Fix/Workaround

Add an inverter.

50.3.21.6 Bad value in Number of Errors Register

The Number of Errors Register always returns 0 instead of the ISO7816 error number.

Problem/Fix Workaround

None.

51. Revision History

The most recent version appears first in the tables that follow.

Revision	Comments	Change Request Ref.
6249H	Erratum “RSTC: Reset during SDRAM Accesses” removed from Rev A Erratum “SDCLK Clock Active after Reset “removed from Rev B 47.3.2 “Power Sequence“ moved to a Section 47.4 on page 1029	RFO
	EBI0_NCS3 restriction added to Section 10.4.2 “ETM™” on page 41	6053
	In Section 20. ”AT91SAM9263 Bus Matrix”, slave number changed from 7 to 6 in ”Description”, and in ”Bus Matrix (MATRIX) User Interface”.	6055
	1st column header of Table 49-1, ”AT91SAM9263 Ordering Information”, on page 1058 edited.	6066
	Section 33. “Two-wire Interface (TWI)” on page 477 changed: TWI2_6212E.fm --> TWI_6212C.fm.	6081
	Erratum added to rev A as Section 50.2.13.1 “RSTC: Reset during SDRAM Accesses” on page 1065, and to rev B as Section 50.2.13.1 “RSTC: Reset during SDRAM Accesses” on page 1065	6083
	Voltage row added to Table 47-8, ”XIN Clock Electrical Characteristics”, on page 1031.	6165
	Rev A and B MCI errata ”SDIO interrupt does not work with slots other than A” edited.	6169
	Bit values changed in Section 28.9.16 “PMC Interrupt Mask Register” on page 375.	6311
	‘All peripheral clocks are deactivated’ added to Section 47.3.1 “Power Consumption versus Modes” on page 1026	6343
	Second paragraph in Section 5.3 “Programmable I/O Lines Power Supplies” on page 13 edited. Section 47.8.4 “SDRAMC Signals” on page 1044 edited, 2 tables inserted.	6395
	Table 47-27, ”SDRAMC Clock Signal”, on page 1044 edited.	6396
	‘Selectable by software...’ removed from Table 47-2, ”DC Characteristics”, on page 1025.	6402
	3 Rev B errata added: USART Section 50.3.21.6 “Bad value in Number of Errors Register” on page 1085, SSC Section 50.3.15.3 “Unexpected Delay on TD output” on page 1081, and PWM Section 50.3.16.1 “Zero Period” on page 1082.	6466
	Rev A USART errata Section 50.2.21.4 on page 1071 to Section 50.2.21.6 on page 1072 duplicated to Rev B.	6475

Revision	Comments	Change Request Ref.
6249G	**Overview:** ”Features” Debug Unit (DBGU) updated. Section 10.4.3 ”EBI1”, updated Section 10.4.4 ”Ethernet 10/100MAC”, added to datasheet Section 6.5 ”Shutdown Logic Pins”, updated, “SHDN pin is tri state output.....”	 5846 5903 rfo
	BootProgram: Section 13.5 ”SD Card Boot” Boot ROM does not support high capacity SDCards, added to datasheet Section 13.8 ”Hardware and Software Constraints”, added footnote (1) Boot ROM does not support high capacity SDCards.	 5935
	DBGU Section 30.1 ”Overview”added text: “The Debug Unit two-pin UART can be used stand-alone for general purpose serial communication.”	5846
	SMC: “Programmable IO Delays” removed from datasheet	 5926
	USART: Manchester Encoding/Decoding is available in this product	5929
	Errata: Section 50.2 ”AT91SAM9263 Errata - Revision “A” Parts” Section 50.2.1.1 ”Main Oscillator: Spurious Malfunction of Main Oscillator”, added to errata. Section 50.2.15.9 ”SPI: Software Reset Must be Written Twice”, added to errata. Section 50.3 ”AT91SAM9263 Errata - Revision “B” Parts” Section 50.3.1.1 ”Main Oscillator: Spurious Malfunction of Main Oscillator”, added to errata. Section 50.3.14.8 ”SPI: Software Reset Must be Written Twice”, added to errata.	 5960 5957 5960 5957
6249G	Section 50.2 ”AT91SAM9263 Errata - Revision “A” Parts” Section 50.2.6.3 ”CAN: Contents of Mailbox 0 can be sent Even if Mailbox is Disabled” Section 50.3 ”AT91SAM9263 Errata - Revision “B” Parts” Section 50.3.5.3 ”CAN: Contents of Mailbox 0 can be sent Even if Mailbox is Disabled”	 6048
	Section 50.3 ”AT91SAM9263 Errata - Revision “B” Parts” Section 50.3.11 ”Reset Controller (RSTC)”, and Section 50.3.11.1 ”RSTC: ERSTL Default Value is 1” added to errata	 5882
	Electrical Characteristics: Section 47.4 ”Core Power Supply POR Characteristics”, added to datasheet.	 6042

Revision 6249F	Comments	Change Request Ref.
	Section 5.1 ”Power Supplies”, VDDCORE and VDDBU updated. Section 5.2, “Power Sequence Requirements removed from datasheet.	5791/5793
	USART: Section 34.5.1 ”I/O Lines”, reference to modem mode removed. Section 34.3 ”Application Block Diagram”, updated and modem removed	5743

Revision 6249E	Comments	Change Request Ref.
	Overview:	
⚠	New Ordering Code: **AT91SAM9263B-CU** added to Table 49-1, ”AT91SAM9263 Ordering Information”.	5560
	Section 8.1.2.1 ”BMS = 1, Boot on Embedded ROM”, changes to list under “Bootloader on a non-volatile memory”.	5425
	Section 5.2 ”Power Sequence Requirements”, section added to datasheet.	5643
	Section 10.4.3 ”EBI1”, System Resource Multiplexing, Ethernet 10/100 MAC limitation on EBI1 updated.	5713
	Section 10.5.8 ”Multimedia Card Interface”, protocol specification compatibilities updated.	5282
	Table 3-1, ”Signal Description List”, Image Sensor Interface, ISI_MCK is provided by PCK3. Table 10-6, ”Multiplexing on PIO Controller E”, ISI_MCK removed from PE11 line of the table.	5329
⚠	**New**!!! Address Offsets added to all register tables in User Interface sections of the datasheet.	rfo
	AIC: “AIC Spurious Interrupt Vector Register” on page 398 fixed typo in bitfield. Section 29.6.3 ”Interrupt Sources”, Interrupt Source 1 OR-wiring description updated. Section 29.7.5 ”Protect Mode”, “writing PROT enables Debug Control Protect Mode in AIC_DCR register, updated.	4749: 5191 5193
	Boot Program: Section 13.1 ”Overview”, 3rd and 4th paragraphs updated. Section 13.6 ”NAND Flash Boot”, 1st and 2nd paragraphs removed, 3rd paragraph updated. Section 13.6.1 ”Supported NAND Flash Devices”, must be connected on EBI0. Section 13.3 ”Device Initialization”, additional supported crystals added in Table 13-1.	5244 5266 5280
	CAN: Section 37.6.4.6 “Error Interrupt Handler” on page 661, section added to datasheet.	4736

Revision 6249E	Comments (Continued)	Change Request Ref.
	DMAC:	
	Figure 25-4, “External DMA Request Timing,” on page 275, updated.	5503
	Section 25.3.3.5 ”External DMA Request Definition”, 3rd paragraph CTLxL.xxx typos fixed. “The DMA ends...” added to 4th paragraph.	rfo
	Multiple bits represented by “x” in Bit names from Section 25.4.11 on page 315 to Section 25.4.23 on page 327	5504
	Section 25.4.7 ”Configuration Register for Channel x High”, SRC_PER bitfield description unscrambled from previous paragraph.	5524
	Section 25.4 ”DMA Controller (DMAC) User Interface”, DMA_IdReg, DMAC_DmaTestReg removed. address 0x3a4, 0x3b8 is reserved.	5644
	Debug and Test	
	Figure 12-6, “AMP Mictor Connector Orientation”, replaced internal product reference in figure.	5385
	Table 12-3, ”AT91SAM9263 JTAG Boundary Scan Register”, pin names assigned to bits 517, 205. pin name updated for bit 203.	5607
	EMAC:	
	Section 41.3.1 ”Clock” added to Functional Description	3328
	GPBR:	
	The General Purpose Backup Registers section has been added.	rfo
	ISI:	
	Section 46.4.7 ”ISI Preview Register”: PREV_HSIZE and PREV_VSIZE updated with RGB only comments.	5430
	LCDC:	
	Section 44.10.2 ”TFT Mode Example”, “HFP...” line, typo corrected: “HPW = (64-2)”	5619
	MCI:	
	Section 40.1 ”Overview”, MCI supports Multimedia Card (MMC) Specification V3.31	5282
	PMC and CKGR**:**	
	Section 28.1 ”Overview”: “PCK must be switched off.....”	4322
	Section 28.3 ”Processor Clock Controller”, new information added explaining Idle Mode.	
	Figure 27-1, “Typical Slow Clock Crystal Oscillator Connection”, corrected GNDPLL to GNDBU	4470
	Section 27.4.2 ”Divider and Phase Lock Loop Programming”, added the last line, specific to PLLA and PLLB initialization.	5046
	Section 28.7 ”Programming Sequence”; in step 3: “Setting PLLA and divider A”, ICPPLA requirement added to 1st paragraph. in step 4: “Setting PLLB and divider B”, ICPPLB requirement added to 1st paragraph.	
	Section 28.7 ”Programming Sequence”; in step 6, “Selection of Master CLock and Processor Clock”, 4th paragraph, this text has been added: “By default, the PRES parameter is set to 0 which means that master clock is equal to slow clock.”	5596
	PWM:	
	Section 38.6 ”Pulse Width Modulation Controller (PWM) User Interface”, Channel-dependent registers are indexed. (see Table 38-2 on page 719)	4486
	Section 38.6.13 ”PWM Channel Update Register” Fixed typos in table: CPD (PWM_CMRx Register)	5185
	RSTC:	
	Section 14.3.4.4 ”Software Reset”, PERRST must be used with PROCRST, except for debug purposes.	5436

Revision 6249E	Comments (Continued)	Change Request Ref.
	SDRAMC: Section 23.6.1 "SDRAMC Mode Register"; changed MODE bit description. SDRAMC Configuration Register "CAS: CAS Latency" on page 250, changed CAS bit description	 4593 4623
	SHDWC: Table 18-1, "Shutdown Controller Block Diagram"; corrected register names in the block diagram.	 4734
	SMC: Section 22.14 "Programmable IO Delays", added to datasheet Section 22.8.5 "Coding Timing Parameters", "Effective Value" column under "Permitted Range" updated in Table 22-4 on page 205. Section 22.9.3.1 "User Procedure", Instructions regarding configuration parameters of SMC Chip Select added.	 rfo/4951 5604 5621
	SPI: Section 32.7.9 "SPI Chip Select Register", note pertaining to BITS field added. This note is referenced in the BITS bitfield description and Section 32.6.4 "SPI Slave Mode"	 5588
	SSC: Section 35.8.3 "SSC Receive Clock Mode Register", corrected bit name to STTDLY.	 4778
	TC: Section 39.6 "Timer Counter (TC) User Interface", Register mapping consolidated into one table, offsets of channel registers indexed and referenced in the register tables that follow. Functional value of WAVE is given. Section 39.6.2 "TC Block Mode Register", typo fixed in bit fields 2 and 3. Section 39.6.4 "TC Channel Mode Register: Capture Mode", bit field 15 updated.	4583
	TDGC: All mention of Polygon Fill feature removed. Fill Control Register, Vertex X, Vertex Y registers removed from User Interface, addresses reassigned as "Reserved" in, Table 45-2, "TDGC Register Mapping," on page 971. User Interface: 2D Graphics Controller (2DGC) replaced by Two D Graphics Controller (TDGC). Section 45.5.23 "VRAM Size Register", VSIZE field (updated with explicit hex radix). Section 45.5.4 "Source/Begin Y Register", typo "YCOR" fixed, = "YSRC". Section 45.5.15 "Extended End Y Register", typo EXT_BY fixed, = "EXT_EY". Table title number removed from tables in bitfield descriptions.	5205 5627 rfo
	TWI: Section 33. "Two-wire Interface (TWI)", the implementation of the TWI has been updated in this product.	 rfo

Revision 6249E	Comments (Continued)	Change Request Ref.
	UDP:	
	Section 43.6.10 "UDP Endpoint Control and Status Register", update to code and added instructions regarding USB clock and system clock rate.	4462
	Updated note under code with text"....a wait time of 3 UDPCK clock cycles and 3 peripheral clock cycles is required..." Ditto for RX_DATA and TXPKTRDY bit fields	4487
	Section 43.2 "Block Diagram", in 2nd paragraph under the block diagram, peripheral clock requirements updated.	4508
	Table 43-4, "Register Mapping", UDP_CSR, UDP_FDR updated with indexed offsets. Footnote added to UDP_ISR reset	4802
	Section 43.6.6 "UDP Interrupt Mask Register", bit 12 defined as BIT12, cannot be masked.	
	Section 43.6 "USB Device Port (UDP) User Interface", reset value for UDP_RST_EP is 0x0000_0000 TXCV typo fixed to TXVC.	5049
	Table 43-1, "USB Endpoint Description", footnote added to Dual-Bank heading Section 43.5.2.9 "Transmit Data Cancellation", added to datasheet. Section 43.6.9 "UDP Reset Endpoint Register", added steps to clear endpoints below warning. "TXPKTRDY: Transmit Packet Ready" bit field description in USB_CSR, updated "Write: 0 =..."	5150
	UHP:	
	Section 38.1 "Overview", created hyperlink to "Open HCI Rev1.0 Specification".	4361
	USART	
	Section 34.6.3.1 "Transmitter Operations", last paragraph updated.	rfo
	Section 34.6.5 "IrDA Mode", updated with instructions to receive IrDA signals.	4912
	Figure 31-1, "USART Block Diagram", updated signal directions from to PIO.	4905
	"AT91SAM9263 Electrical Characteristics"	
	Table 47-2, "DC Characteristics", V_{OL} and V_{OH} lines updated.	5288
	Section 47.10.1 "SPI": the Figure 47-8 on page 1051 and Figure 47-9 on page 1051 renamed.	5260
	Figure 47-14, "USB Data Signal Rise and Fall TImes", R_{EXT} = 27Ω	5610
	"AT91SAM9263 Ordering Information"	
	New Ordering Code: AT91SAM9263B-CU, added to Table 49-1, "AT91SAM9263 Ordering Information".	5560
	"AT91SAM9263 Errata - Revision "A" Parts"	
	Section 50.2.2.2 "TDGC Clipping Function", added to errata.	5384
	Section 50.2.9.2 "LCD Periodic Bad Pixels", 16-word and 4-word offset requirements text updated.	5265
	Section 50.2.14 "Static Memory Controller (SMC)", "SMC Chip Select Parameters Modification", added	5642
	"AT91SAM9263 Errata - Revision "B" Parts"	
	Section 50.3.2.2 "TDGC Clipping Function", added to errata.	5384
	Section 50.3.8.2 "LCD Periodic Bad Pixels", 16-word and 4-word offset requirements text updated.	5265
	Section 50.3.14.6 "SPI: SPI Software Reset Must Be Written Twice", added to errata.	5597
	Section 50.3.13 "Static Memory Controller (SMC)", "SMC Chip Select Parameters Modification", added	5642

Revision	Comments	Change Request Ref.
6249D	**Overview:**	
	"Features" SPI: Synchronous Communications feature removed.	4910
	Section 5.1 "Power Supplies", VDDIO and VDDBU slope alignment described.	4967
	Section 5.2 "Power Consumption", paragraph beginning with "On VDDBU..." updated.	4505
	Section 10.5.8 "Multimedia Card Interface","When REMAP = 1....." removed from 2nd paragraph.	5029
	Section 8.2.1.1 "External Bus Interface 0", feature added.	4146
	Section 8.2.1.2 "External Bus Interface 1" feature added.	
	"Package and Pinout", references to package are "324-ball TFBGA"	4664
	Figure 9-3, "AT91SAM9263 Power Management Controller Block Diagram," on page 30, /3 divider removed.	4834
	Section 10.5.8 "Multimedia Card Interface", MMC and SDMC compatibility updated.	4945
	Bus Matrix:	
	Section 20.5 "Bus Matrix (MATRIX) User Interface", MATRIX_PRASx and MATRIX_PRBSx are Write-only.	4633
	Section 20.5.5 "Bus Matrix Master Remap Control Register", remap control bits and masters are described.	4632
	Boot: Section 13.8 "Hardware and Software Constraints", internal SRAM constraints and user area are defined.	4587
	EBI: Table 21-4, "EBI Pins and External Static Devices Connections", NCS2/NANDCS signals defined.	4587
	"Power Management Controller (PMC) User Interface" PLL CHarge Pump Current Register (PMC_PLLICPR) removed.	4587
	"AT91SAM9263 Electrical Characteristics" and "AT91SAM9263 Mechanical Characteristics"	
	Table 47-2, "DC Characteristics" Reference to Junction Temperature removed.	4730
	and "AT91SAM9263 Mechanical Characteristics", Thermal Considerations removed.	
	Table 47-2, "DC Characteristics", VIL, VIH, VOL, VOH, conditions updated	4927
	Table 47-10, "32 kHz Oscillator Characteristics", on page 1033, removed VDDOSC = 1.2V	4838
	Section 47.6.2 "32 kHz Crystal Characteristics" on page 1033, added.	
	Section 47.8 "EBI Timings", tables and figures pertinent to the SMC in this section have been updated.	5108
	Figure 48-1, "324-ball TFBGA Package Drawing," on page 1056, updated.	4668
	"AT91SAM9263 Errata"	
	Section 50.2 "AT91SAM9263 Errata - Revision "A" Parts"	
	"CAN", Section 50.2.6.2 "CAN: Low Power Mode and Pending Transmit Messages" updated.	
	Mailbox error on TX, removed from errata.	
	Section 50.2.9 "LCD" added to errata	
	"MCI", Section 50.2.10.3 "Data Timeout Error Flag", Section 50.2.10.4 "Data Write Operation and Number of Bytes", Section 50.2.10.5 "Flag Reset is not correct in half duplex mode" added	4692
	Section 50.2.13.2 "Mobile SDRAM Device Initialization Constraint", updated.	4638
	"Serial Peripheral Interface (SPI)" Section 50.2.15.1 "SPI: PDC Data Loss" updated.	rfo
	Section 50.2.15.8 "SPI: Bad Serial Clock Generation on 2nd Chip Select", added.	4771

Revision	Comments	Change Request Ref.
6249D	”AT91SAM9263 Errata” (continued) ”Serial Synchronous Controller (SSC)”, Section 50.2.16.3 ”SSC: Last RK Clock Cycle when RK Outputs a Clock During Data Transfer” and Section 50.2.16.4 ”SSC: First RK Clock Cycle when Rk Outputs a Clock During Data Transfer”, added.	4771
	”USART”, Section 50.2.21.4 ”Two characters sent if CTS rises during emission”, added. Section 50.2.21.5 ”TXD signal is floating in Modem and Hardware Handshaking mod” and Section 50.2.21.6 ”DCD is Active High instead of Low”, added	4692 4721
	Section 50.3 ”AT91SAM9263 Errata - Revision “B” Parts” has been added to errata.	5168

Revision	Comments	Change Request Ref.
6249C		
	In Section 4.1 “324-ball TFBGA Package Outline” on page 10 corrected package top view.	4463
	All new information for Table 7-1, “List of Bus Matrix Masters,” on page 16, Table 7-2, “List of Bus Matrix Slaves,” on page 17 and Table 7-3, “Masters to Slaves Access,” on page 18.	4466
	In Section 9.3 “Shutdown Controller” on page 29, corrected reference to shutdown pin.	3870
	In Section 5.2 “Power Consumption” on page 13, specified static current consumption as worst case. Corrected Section 10.4.7 “NAND Flash” on page 41, with information on EMAC.	3825
	In Section 10.4.3 “EBI1” on page 41, added Ethernet 10/100 MAC to the System Resource Multiplexing list of EBI1.	4064
	In Section 10.4.11 “Image Sensor Interface” on page 42 and Section 10.4.12 “Timers” on page 42, removed mention of keyboard interfaces.	4407
	In Table 12-3, ”AT91SAM9263 JTAG Boundary Scan Register” on page 86, changed pin 246 to NC.	
	Boot: Updated supported crystals in Table 13-1, “Crystals Supported by Software Auto-detection (MHz),” on page 97. Corrected Figure 13-3, “LDR Opcode,” on page 98. Updated supported DataFlash device references in Table 13-2, “DataFlash Device,” on page 99.	4230 4450 4186
	EBI: Updated Figure 21-1, “Organization of the External Bus Interface 0,” on page 164 and Figure 21-2, “Organization of the External Bus Interface 1,” on page 165 and Table 21-5, “EBI Pins and External Devices Connections,” on page 170 with NANDALE and NANDCLE pins. Removed note on CLE and ALE NAND Flash signals. Removed reference to EBI0 for NANDOE and NANDWE. In Table 21-2, “EBI1 I/O Lines Description,” on page 167, corrected EBI address bus width. In Table 21-5, “EBI Pins and External Devices Connections,” on page 170, corrected NAND Flash AD to I/O. Added Note(5) on CE and NAND Flash.	4149 3850 3905
	SHDW: In Table 18-2, “Register Mapping,” on page 145, corrected offset value for SHDW_SR register.	4224
	SMC: In Section 22.7.2.1 “Byte Write Access” on page 195, added information that boot is not allowed in Byte Write Access mode.	3252
	ECC: Section 24.3 “Functional Description” on page 258 updated. Section 24.3.1 “Write Access” on page 258 and Section 24.3.2 “Read Access” on page 258 updated. Section 24.4.4 “ECC Parity Register” on page 266 and Section 24.4.5 “ECC NParity Register” on page 267 updated. In Table 24-1, “Register Mapping,” on page 262, corrected offset value for ECC_SR.	3970 4306
	PMC: In Section 28.3 “Processor Clock Controller” on page 349, new details on PCK disable.	3835
	PIO: Notes (1), (2) and (3) updated in Table 31-2, “Register Mapping,” on page 434.	3974
	SPI: In Section 32.6.4 “SPI Slave Mode” on page 461, updated information on OVRES bit.	3943
	TWI: New Section 33. “Two-wire Interface (TWI)” on page 481.	

Revision	Comments	Change Request Ref.
6249C	USART: In Section 34.5.1 “I/O Lines” on page 521 added information on TXD enabled. In Section 34.6.2 “Receiver and Transmitter Control” on page 527, corrected information on software reset.	4825 4367
	CAN: In Figure 37-7 on page 660 corrected mode switch conditions.	4089
	UDP: Table 43-2 on page 863, “Supported Endpoint” column updated in the USB Communication Flow. Updated description of bit EPEDS in the USB_CSR register on page 894 for details on control endpoints not affected. Updated: write 1=.....”RX_DATA_BK0: Receive Data Bank 0”bitfield in USB_CSR Updated: write 0 =...”TXPKTRDY: Transmit Packet Ready” bitfield in USB_CSR	3476 4063 4099
	LCDC: In Table 44-1, “I/O Lines Description,” on page 899, updated description of LCDDEN. Updated Table 44-4, ”Little Endian Memory Organization”, on page 903, with Pixel 24 bpp unpacked format. Updated bit description ”PIXELSIZE: Bits per pixel” on page 938 in LCDCON2 register, updated bit configuration table with new value for 24 bpp unpacked. In Section 44.11.24 “Power Control Register” on page 947, LCD_PWR bit description, changed all occurrences of “pin” to “signal”.	3587
	ISI: Added information to CODEC_ON bit description in Section 46.4.1 “ISI Control 1 Register” on page 1006. Added Bit 3 CDC_PND to Section 46.4.3 “ISI Status Register” on page 1010. Correction to name of Section 46.4.8 “ISI Preview Decimation Factor Register” on page 1017. Added note on ISI_PCK to Table 46-9, “Register Mapping,” on page 1005. Updated ISI_RST bit description in Section 46.4.1 “ISI Control 1 Register” on page 1006.	3518 3519 3250 3524
	In Table 47-3, “Power Consumption for Different Modes,” on page 1027, added note for SRAM access.	3904
	Added Table 47-7, “Master Clock Waveform Parameters,” on page 1031.	4304
	Updated Section 47.6 “Crystal Oscillator Characteristics” on page 1033.	4092, 3862
	Corrected VDDOSC value in Table 47-10, “32 kHz Oscillator Characteristics,” on page 1033.	4244
	Errata changes: Inserted “Two D Graphic Controller (TDGC)” , Section 50.2.2.1 “Polygon Fill Function” on page 1060. Inserted “EMACB” , Section 50.2.8.1 “Transmit Underrun Errors” on page 1062. Updated “USART” , Section 49.2.19.3 “CTS Signal in Hardware Handshake” on page 1038. Inserted “USART” , Section 50.2.21.1 “SCK1 and SCK2 are Inverted” on page 1071. Inserted “SDRAM Controller” , Section 50.2.13.3 “JEDEC Standard Compatibility” on page 1065. Inserted Section 50.2.11.1 “NTRST: Device does not boot correctly due to power-up sequencing issue” on page 1064. Inserted Section 50.2.4.1 “BMS Does Not Have Correct State” on page 1060.	4093 4093 4093 4465 4221 3882

Revision	Comments	Change Request Ref.
6249B	Corrected typo to IDE hard disk in Section 1. ”Description”, on page 3.	3804
	In Section 13. “AT91SAM9263 Boot Program” on page 95, added information on NAND Flash and SDCard boot in the ROM.	3802
	Corrected typo in PB range in Table 46-2, “DC Characteristics,” on page 983. Updated Static Current conditions and values.	3804
	Corrected ordering code in Section 48. ”AT91SAM9263 Ordering Information”, on page 1017.	3805
	In “AT91SAM9263 Errata”, added Section 49.2.6.1 ”SDCard Boot is not functional”, on page 1021 and Section 49.2.6.2 ”NAND Flash Boot is not functional”, on page 1021.	3803
6249A	First issue.	

Table of Contents

Features *1*

1 Description *3*

2 AT91SAM9263 Block Diagram *4*

3 Signal Description *5*

4 Package and Pinout *10*

4.1 324-ball TFBGA Package Outline 10

4.2 324-ball TFBGA Package Pinout 11

5 Power Considerations *13*

5.1 Power Supplies 13

5.2 Power Consumption 13

5.3 Programmable I/O Lines Power Supplies 13

6 I/O Line Considerations *14*

6.1 JTAG Port Pins 14

6.2 Test Pin 14

6.3 Reset Pins 14

6.4 PIO Controllers 14

6.5 Shutdown Logic Pins 14

7 Processor and Architecture *15*

7.1 ARM926EJ-S Processor 15

7.2 Bus Matrix 16

7.3 Matrix Masters 16

7.4 Matrix Slaves 16

7.5 Master to Slave Access 18

7.6 Peripheral DMA Controller 18

7.7 DMA Controller 19

7.8 Debug and Test Features 20

8 Memories *21*

8.1 Embedded Memories 22

8.2 External Memories 25

9 System Controller *27*

9.1 System Controller Block Diagram 28

9.2Reset Controller 29
9.3Shutdown Controller 29
9.4Clock Generator 29
9.5Power Management Controller 30
9.6Periodic Interval Timer 30
9.7Watchdog Timer 30
9.8Real-time Timer 31
9.9General-purpose Backup Registers 31
9.10Backup Power Switch 31
9.11Advanced Interrupt Controller 31
9.12Debug Unit 31
9.13Chip Identification 32
9.14PIO Controllers 32

10 Peripherals 33

10.1User Interface 33
10.2Identifiers 33
10.3Peripherals Signals Multiplexing on I/O Lines 34
10.4System Resource Multiplexing 41
10.5Embedded Peripherals Overview 42

11 ARM926EJ-S Processor Overview 47

11.1Overview 47
11.2Block Diagram 48
11.3ARM9EJ-S Processor 48
11.4CP15 Coprocessor 56
11.5Memory Management Unit (MMU) 59
11.6Caches and Write Buffer 60
11.7Tightly-Coupled Memory Interface 62
11.8Bus Interface Unit 63

12 AT91SAM9263 Debug and Test 65

12.1Overview 65
12.2Block Diagram 66
12.3Application Examples 67
12.4Debug and Test Pin Description 68
12.5Functional Description 68

13 AT91SAM9263 Boot Program 95

13.1Overview 95
13.2Flow Diagram 95
13.3Device Initialization 97
13.4DataFlash Boot 98
13.5SD Card Boot 101
13.6NAND Flash Boot 101
13.7SAM-BA Boot 101
13.8Hardware and Software Constraints 105

14 Reset Controller (RSTC) 107

14.1Overview 107
14.2Block Diagram 107
14.3Functional Description 108
14.4Reset Controller (RSTC) User Interface 117

15 Real-Time Timer (RTT) 121

15.1Description 121
15.2Block Diagram 121
15.3Functional Description 121
15.4Real-time Timer (RTT) User Interface 123

16 Periodic Interval Timer (PIT) 127

16.1Overview 127
16.2Block Diagram 127
16.3Functional Description 128
16.4Periodic Interval Timer (PIT) User Interface 130

17 Watchdog Timer (WDT) 135

17.1Overview 135
17.2Block Diagram 135
17.3Functional Description 136
17.4Watchdog Timer (WDT) User Interface 138

18 Shutdown Controller (SHDWC) 143

18.1Overview 143
18.2Block Diagram 143
18.3I/O Lines Description 143
18.4Product Dependencies 144
18.5Functional Description 144

18.6Shutdown Controller (SHDWC) User Interface 145

19 General Purpose Backup Registers (GPBR) 149

19.1Overview 149

19.2General Purpose Backup Registers (GPBR) User Interface 149

20 AT91SAM9263 Bus Matrix 151

20.1Description 151

20.2Memory Mapping 151

20.3Special Bus Granting Techniques 151

20.4Arbitration 152

20.5Bus Matrix (MATRIX) User Interface 155

20.6Chip Configuration User Interface 160

21 External Bus Interface (EBI) 163

21.1Overview 163

21.2Block Diagram 164

21.3I/O Lines Description 166

21.4Product Dependencies 173

21.5Functional Description 173

21.6Implementation Examples 181

22 Static Memory Controller (SMC) 191

22.1Overview 191

22.2I/O Lines Description 191

22.3Multiplexed Signals 191

22.4Application Example 192

22.5Product Dependencies 193

22.6External Memory Mapping 193

22.7Connection to External Devices 194

22.8Standard Read and Write Protocols 198

22.9Automatic Wait States 206

22.10Data Float Wait States 211

22.11External Wait 215

22.12Slow Clock Mode 221

22.13Asynchronous Page Mode 224

22.14Static Memory Controller (SMC) User Interface 227

23 SDRAM Controller (SDRAMC) 233

23.1Overview 233
23.2I/O Lines Description 233
23.3Application Example 234
23.4Product Dependencies 235
23.5Functional Description 238
23.6SDRAM Controller (SDRAMC) User Interface 246

24 Error Corrected Code (ECC) Controller 257

24.1Overview 257
24.2Block Diagram 257
24.3Functional Description 258
24.4Error Corrected Code Controller (ECC) User Interface 262

25 DMA Controller (DMAC) 269

25.1Overview 269
25.2Block Diagram 269
25.3Functional Description 269
25.4DMA Controller (DMAC) User Interface 299

26 Peripheral DMA Controller (PDC) 329

26.1Overview 329
26.2Block Diagram 330
26.3Functional Description 330
26.4Peripheral DMA Controller (PDC) User Interface 333

27 Clock Generator 343

27.1Overview 343
27.2Slow Clock Crystal Oscillator 343
27.3Main Oscillator 343
27.4Divider and PLL Block 345

28 Power Management Controller (PMC) 348

28.1Overview 348
28.2Master Clock Controller 348
28.3Processor Clock Controller 349
28.4USB Clock Controller 349
28.5Peripheral Clock Controller 350
28.6Programmable Clock Output Controller 350
28.7Programming Sequence 351

28.8Clock Switching Details 355
28.9Power Management Controller (PMC) User Interface 359

29 Advanced Interrupt Controller (AIC) 377

29.1Overview 377
29.2Block Diagram 378
29.3Application Block Diagram 378
29.4AIC Detailed Block Diagram 378
29.5I/O Line Description 379
29.6Product Dependencies 379
29.7Functional Description 380
29.8Advanced Interrupt Controller (AIC) User Interface 390

30 Debug Unit (DBGU) 401

30.1Overview 401
30.2Block Diagram 402
30.3Product Dependencies 403
30.4UART Operations 403
30.5Debug Unit (DBGU) User Interface 410

31 Parallel Input/Output Controller (PIO) 425

31.1Overview 425
31.2Block Diagram 426
31.3Product Dependencies 427
31.4Functional Description 428
31.5I/O Lines Programming Example 433
31.6Parallel Input/Output Controller (PIO) User Interface 434

32 Serial Peripheral Interface (SPI) 451

32.1Overview 451
32.2Block Diagram 452
32.3Application Block Diagram 452
32.4Signal Description 453
32.5Product Dependencies 453
32.6Functional Description 454
32.7Serial Peripheral Interface (SPI) User Interface 463

33 Two-wire Interface (TWI) 477

33.1Description 477

33.2Embedded Characteristics 477
33.3List of Abbreviations 477
33.4Block Diagram 478
33.5Application Block Diagram 478
33.6Product Dependencies 479
33.7Functional Description 479
33.8Master Mode 481
33.9Multi-master Mode 492
33.10Slave Mode 495
33.11Two-wire Interface (TWI) User Interface 503

34 Universal Synchronous Asynchronous Receiver Transmitter (USART) 517

34.1Overview 517
34.2Block Diagram 518
34.3Application Block Diagram 519
34.4I/O Lines Description 520
34.5Product Dependencies 521
34.6Functional Description 522
34.7Universal Synchronous Asynchronous Receiver Transmitter (USART) User Interface 552

35 Synchronous Serial Controller (SSC) 573

35.1Overview 573
35.2Block Diagram 574
35.3Application Block Diagram 574
35.4Pin Name List 575
35.5Product Dependencies 575
35.6Functional Description 575
35.7SSC Application Examples 586
35.8Synchronous Serial Controller (SSC) User Interface 589

36 AC97 Controller (AC97C) 613

36.1Overview 613
36.2Block Diagram 614
36.3Pin Name List 615
36.4Application Block Diagram 615
36.5Product Dependencies 616
36.6Functional Description 617

36.7AC97 Controller (AC97C) User Interface 628

37 Controller Area Network (CAN) 649

37.1Overview 649

37.2Block Diagram 650

37.3Application Block Diagram 651

37.4I/O Lines Description 651

37.5Product Dependencies 651

37.6CAN Controller Features 652

37.7Functional Description 664

37.8Controller Area Network (CAN) User Interface 677

38 Pulse Width Modulation Controller (PWM) 709

38.1Overview 709

38.2Block Diagram 709

38.3I/O Lines Description 710

38.4Product Dependencies 710

38.5Functional Description 710

38.6Pulse Width Modulation Controller (PWM) User Interface 719

39 Timer Counter 733

39.1Overview 733

39.2Block Diagram 734

39.3Pin Name List 735

39.4Product Dependencies 735

39.5Functional Description 736

39.6Timer Counter (TC) User Interface 749

40 MultiMedia Card Interface (MCI) 767

40.1Overview 767

40.2Block Diagram 768

40.3Application Block Diagram 769

40.4Pin Name List 769

40.5Product Dependencies 769

40.6Bus Topology 770

40.7MultiMedia Card Operations 773

40.8SD/SDIO Card Operations 781

40.9MultiMedia Card Interface (MCI) User Interface 782

41 Ethernet MAC 10/100 (EMAC) 803

41.1Overview 803

41.2Block Diagram 803

41.3Functional Description 804

41.4Programming Interface 815

41.5Ethernet MAC 10/100 (EMAC) User Interface 818

42 USB Host Port (UHP) 855

42.1Description 855

42.2Block Diagram 855

42.3Product Dependencies 856

42.4Functional Description 856

42.5Typical Connection 858

43 USB Device Port (UDP) 859

43.1Overview 859

43.2Block Diagram 860

43.3Product Dependencies 860

43.4Typical Connection 862

43.5Functional Description 863

43.6USB Device Port (UDP) User Interface 878

44 LCD Controller (LCDC) 897

44.1Overview 897

44.2Block Diagram 898

44.3I/O Lines Description 899

44.4Product Dependencies 899

44.5Functional Description 899

44.6Interrupts 920

44.7Configuration Sequence 920

44.8Double-buffer Technique 921

44.92D Memory Addressing 922

44.10Register Configuration Guide 924

44.11LCD Controller (LCDC) User Interface 925

45 Two D Graphics Controller (TDGC) 957

45.1Description 957

45.2Block Diagram 958

45.3Functional Description 959

45.4Examples of Drawing Functions 967
45.5Two D Graphic Controller (TDGC) User Interface 971

46 Image Sensor Interface (ISI) 995

46.1Overview 995
46.2Block Diagram 996
46.3Functional Description 996
46.4Image Sensor Interface (ISI) User Interface 1005

47 AT91SAM9263 Electrical Characteristics 1025

47.1Absolute Maximum Ratings 1025
47.2DC Characteristics 1025
47.3Power Consumption 1026
47.4Core Power Supply POR Characteristics 1029
47.5Clock Characteristics 1031
47.6Crystal Oscillator Characteristics 1033
47.7USB Transceiver Characteristics 1036
47.8EBI Timings 1037
47.9EMAC Timings 1048
47.10Peripheral Timings 1051

48 AT91SAM9263 Mechanical Characteristics 1056

48.1Package Drawing 1056
48.2Soldering Profile 1057

49 AT91SAM9263 Ordering Information 1058

50 AT91SAM9263 Errata 1059

50.1Marking 1059
50.2AT91SAM9263 Errata - Revision “A” Parts 1060
50.3AT91SAM9263 Errata - Revision “B” Parts 1073

51 Revision History 1087

Table of Contents i

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/AT91SAM

Technical Support
AT91SAM Support
Atmel techincal support

Sales Contacts
www.atmel.com/contacts/

Literature Requests
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© 2008 Atmel Corporation. **All rights reserved.** Atmel®, Atmel logo and combinations thereof, DataFlash®, SAM-BA® and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. ARM®, Thumb®, the ARMPowered® logo and others are the registered trademarks or trademarks of ARM Ltd. Windows® and others are the registered trademarks or trademarks of Microsoft Corporation in the US and/or other countries. Other terms and product names may be the trademarks of others.